Virtual Basestation: Architecture for an Open Shared WiMAX Framework *

Gautam Bhanage, Ivan Seskar
WINLAB, Rutgers University
671 US1 South, North Brunswick, NJ 08902
{gautamb, seskar}@winlab.rutgers.edu

Rajesh Mahindra
NEC Labs America
4 Independence Way, Princeton, NJ 08540
rajesh@nec-labs.com

Dipankar Raychaudhuri
WINLAB, Rutgers University
671 US1 South, North Brunswick, NJ 08902
ray@winlab.rutgers.edu

ABSTRACT

This paper presents the architecture and performance evaluation of a virtualized wide-area "4G" cellular wireless network. Specifically, it addresses the challenges of virtualization of resources in a cellular base station to enable shared use by multiple independent slice users (experimenters or mobile virtual network operators), each with possibly distinct flow types and network layer protocols. The proposed virtual basestation architecture is based on an external substrate which uses a layer-2 switched datapath, and an arbitrated control path to the WiMAX base station. The framework implements virtualization of base station's radio resources to achieve isolation between multiple virtual networks. An algorithm for weighted fair sharing among multiple slices based on an airtime fairness metric has been implemented for the first release. Preliminary experimental results from the virtual basestation prototype are given, demonstrating mobile network performance, isolation across slices with different flow types, and custom flow scheduling capabilities.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks, Network Architecture and Design—*Wireless communication*

General Terms

Design, Experimentation, Verification

*Research supported in part by NSF Grant # CNS-0737890, under subcontract from the GENI project office at BBN Technologies.

VISA 2010, September 3, 2010, New Delhi, India.

Keywords

Virtual Basestation, Network Virtualization, Mobile WiMAX, MVNO, GENI, Testbeds, 802.16e

1. INTRODUCTION

Virtualization can be broadly defined as the task of splitting the entire system, including the network resources (wireless or wired), in such a way that every *slice* has the illusion of having the entire system to itself. Further, we can define a *slice* as a subset of the system resources that are allocated to a single user[1]. Hence, using virtualization, a testbed operator can support multiple simultaneous experiments which have a long execution time and improve the testbed's scalability and efficiency. These reasons have been the primary motivation for the consideration or application of virtualization for networking research testbeds such as ORBIT [15], PLANETLAB [14], and Emulab [13]. Virtualizing testbeds also allows for integration in the GENI [2] framework, which envisions the presence of an open heterogeneous experimental substrate. We present the system requirements and design decisions encountered in virtualizing a mobile WiMAX basestation for sharing it across multiple experimenters as a part of the GENI testbed infrastructure.

Though, we discuss deployment in the GENI framework as the main application in the paper, our solution can be applied to other problems in the same class. For example, our virtual basestation framework could be used without modification for supporting multiple mobile virtual network operators (MVNOs) on a single physical basestation supported by the mobile network operator (MNO). Features in our virtual basestation framework such as weighted slice allocation and isolation, opportunity for slice environment control, and frame switching at layer-2 provide multiple opportunities for MVNOs to customize their virtual basestations while still sharing the same physical hardware. Such a shared hardware design presents opportunities for MVNOs to reach the market with a relatively less investment, and provides the flexibility of possibly doing on-demand resource allocation based on agreements with the MNO. From an MNO perspective, the loose coupling of our virtual basestation design

[1]Through the rest of the study, we will use terms *slice* and *user* interchangeably.

from the hardware itself helps to make the design portable across multiple platforms. Finally, our design also allows individual MVNOs to customize flow scheduling within each virtual basestation for achieving service differentiation.

In this study, we propose the virtual basestation framework as a means by which a single physical basestation may be shared across multiple slice users i.e experimenters or MVNOs. Specifically, our contributions are-

1. We lay down guidelines for virtualizing and integrating a 4G wireless radio framework while being independent of the actual hardware chipset. The modified architecture allows simultaneous sharing of underlying resources and preserves repeatability.

2. By ensuring that all frame switching within the network architecture is based purely on layer-2 information, we support the use of arbitrary network, transport, sessions, and application protocols.

3. Through prototyping, we show the feasibility of our design and present preliminary results from the WiMAX Basestation.

4. Controlled experiments show that sufficient radio isolation can be achieved, irrespective of flow priorities (such as UGS or BE), in a shared mode of operation such that scientific conclusions can be drawn with sufficient accuracy.

5. We show how our virtual basestation setup could be used by slice users (experimenters or MVNOs) to customize flow scheduling within their slices.

Rest of the paper is organized as follows. Section 2 gives a brief overview of the system design. Section 3 presents a discussion on the changes in the system required for emulating multiple virtual basestations for individual slices. In section 4 we present the modifications to the control and datapath of the network architecture to support multiple simultaneous slices. Preliminary results and experiment scenarios are discussed in Section 5. Some related work is discussed in Section 6. Finally, Section 7 concludes the study and provides future directions.

2. SYSTEM OVERVIEW

We begin with a brief overview of the network components in a conventional WiMAX deployment. After describing the functions performed by each of these components, we highlight requirements of the virtualized infrastructure. This is followed by a high level overview of our proposed design.

2.1 Conventional WiMAX System Design

A standard Profile-A/C WiMAX system typically consists of three important components: (1) Basestation transceiver system (BTS), (2) Application service network gateway (ASN-GW or ASN), and (3) Content service network gateway (CSN)[2]. The BTS is the main component of the WiMAX system and consists of the air interface that includes the radio which communicates with the clients. Among other things, the BTS is capable of controlling RF features such as the transmission frequency, power, rate, symbol ratios, retransmission mechanisms, and other client management functionality. The BTS interacts with the wired world through the use of the ASN-GW. The ASN-GW is used to route traffic appropriately from the wired interface(s) to individual service flows on WiMAX clients. After describing the requirements for our virtualized framework, we will discuss the modifications and additions needed to these components for realizing our virtual basestation design.

[2]There are other components like the AAA authentication server, but we do not discuss these since they are not a part of the core prototype in our shared setup.

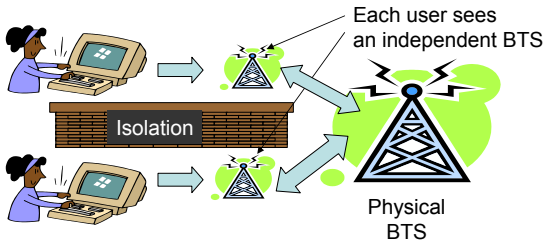

Figure 1: Conceptual illustration of our virtual basestation design. Virtualization consists of three main concepts: Abstraction and programmability, which provide every slice with the experience of having control over a virtual basestation, and slice isolation which ensures performance repeatability.

2.2 Requirements In A Shared BTS Setup

At a conceptual level, the system requirements are as described in Figure 1. Our system should provide every slice/user with the illusion that it has control of an entire basestation, though the radio capacity of this virtual BTS is possibly lesser than the original physical BTS. Such virtual basestations provided to individual slices should have a control framework which is similar to the original basestation. Finally, these virtual basestations should also be isolated from each other i.e, they should not affect each others performance.

Before we delve into the details of our design we will briefly discuss specific requirements which are important from two different perspectives. Though all requirements described in this section are specific to the deployment of a 802.16*e* Mobile WiMAX system, the requirements and corresponding design can be generalized to encompass the deployment of other 4*G* cellular radios such as LTE [4].

Slice User's Perspective.

We begin with a discussion on the design features important from a slice user's i.e an MVNO's or experimenter's perspective:

- Layer-2 Frame Switching: To allow the user to run a custom network stack, which is a key requirement for evaluating the clean slate protocols, the complete datapath for the WiMAX system should be run purely using layer-2 switching.

- Programmability: Each of the slices should have access to most of the features that can be controlled on the Basestation. These include and are not limited to the use of custom service flow definitions and flow scheduling mechanisms.

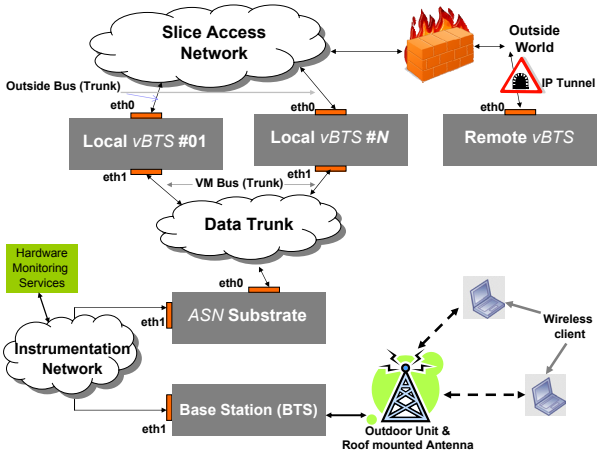

Figure 2: Generic architecture for the proposed WiMAX deployment. This architecture is capable of supporting mutiple *vBTS* substrates, which could be either local or remotely located.

- Slice feedback: Appropriate feedback should be provided to every slice for allowing measurements and performance monitoring. Detailed information such as RSSI, Modulation and coding scheme (MCS), throughput (Mbps), MAC retries, and the CINR ratio are among few of the metrics that are particularly useful for individual slices.

- Repeatability: Slices would want their results to be repeatable over time. Hence, even though the results are scaled down (due to sharing with other slices) they will still allow scientific inference to be drawn from experiments.

Administrator's Perspective.

The system administrator will either be a mobile network operator or a testbed operator. Features which are important from such a system administrator's perspective are as listed below-

- Slice management: Our setup should support creation, destruction, and restoration of slice instances in the system. Such a feature should also support dynamic creation, destruction or modification of the data path for individual slices. Other features under this category include quota management for slices such as CPU, disk or radio resources allocated per slice.

- Monitoring services: Resource tracking services that allow the system administrator to have a broad view of the system permits system usage control and diagnostics. Typically, features such as details on authenticated clients, slices, traffic, current and historical resource usage are useful.

Based on these design considerations, we present our virtual basestation framework. We begin with a broad overview of the network architecture followed by a detailed design description of individual components in the system.

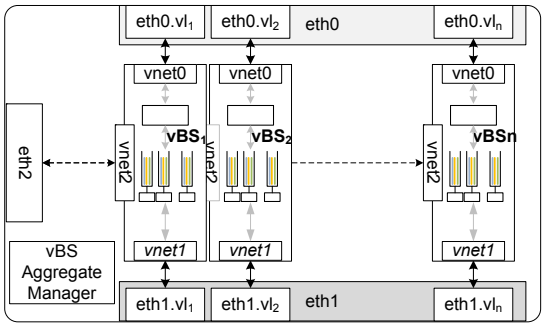

Figure 3: Proposed architecture for the *vBTS* substrate. VLANs are named as $ethX.vl_y$, where X is the number of the physical interface and y denotes the vlan id on that interface.

2.3 Top-Down Design Overview

An architectural overview of our virtual basestation design is as shown in Figure 2. In addition to the Basestation transceiver (BTS) and the Application service network (ASN-GW), which are standard components of the WiMAX framework, we add the virtual basestation (*vBTS*) substrate. Our idea of emulating *vBTS*s outside of the physical BTS is motivated from the fact that the BTS is a proprietary product, and in most cases we will not be able to modify its functionalities or features. Such a loosely coupled system design allows our virtual basestation framework to be portable across other BTS hardware platforms and improves its applicability.

Functionally, the *vBTS* substrate provides a platform for multiple slices to login and conduct experiments through the use of virtual machines. Each of the virtual basestations on the local[3] *vBTS* substrate has an appropriate control and layer-2 datapath configured to reach the physical BTS transparently through the customized ASN-GW. The control and customization of the slice are made feasible through appropriate grid service, while the layer-2 datapath is setup using features such as VLANs, tunnels at layer 2 and frame routing through the Click modular router. These *vBTS*s can now be leased by MVNOs from the operator or they can be directly included as a part of the GENI slices for performing end-to-end experimentation.

In the following sections, the paper will discuss the details and rationale for the design of the virtual basestation substrate as well as the modified ASN-GW substrate.

3. VIRTUAL BASESTATION SUBSTRATE

The main function of the *vBTS* substrate is to emulate an isolated private basestation transceiver for every slice. We begin with a generic discussion of the design methodology for the *vBTS* substrate, followed by a discussion on the components used for prototyping the substrate.

3.1 VM Technology Selection

The *vBTS* substrate fulfills the purpose of emulating a fully functional WiMAX Basestation Transceiver (BTS) to every slice user, while possibly supporting lesser (but fixed) radio capacity than the original BTS itself. To ensure that

[3]Remote mode could be supported by using tunneling.

all design requirements listed in the previous section 2 are fulfilled, we need to carefully select appropriate virtualization technology for running the virtual machines. Specifically, our virtual machine (VM) technology should: (1) Allow for quick creation and deletion of VM instances for management purposes, (2) Support a wide variety of OS distributions to allow customization of environment, (3) Allow the user to customize everything from the Kernel (and hence the network stack) to the drivers used in the OS, and finally, (4) Provide means for easy administration. As per qualitative comparisons provided in a previous study [11], *Full Virtualization* provides each of these features and it is thus preferred over other forms of system virtualization for our *vBTS* implementation.

We use Kernel Virtual Machines (KVM) [3] which are based on the QEMU [7] emulator. This is a full virtualization technique and relies on the CPU architecture to have virtualization extensions such as *Intel VT* or *AMD-V*. KVM can be easily administered from conventional Debian systems, which are also used with our other systems in the ORBIT framework. KVM also has support for a wide variety of distributions for the guest VMs, and supports virtual networking that allows creation of virtual interfaces.

3.2 Slice Access Design

A generalized design of the *vBTS* substrate is as shown in the Figure 3. Each of the VMs are designed to have three virtual interfaces, marked as *vnet0* through *2*. These virtual interfaces are bridged with appropriate VLANs on corresponding physical interfaces. Individual vlan-ids serve as slice identifiers. Functionality provided by each of the virtual interfaces is discussed below.

GENI/Outside Access: The first virtual interface *vnet0* is responsible for providing connectivity to the outside world from the VM. This is achieved by bridging traffic from each *vnet0* to the appropriate outbound VLAN interface. The slice user can also use SSH over this interface to enter into the VM. This interface is directly routable from the physical *eth0* interface on the *vBTS* host machine.

Client Access: The second interface on the virtual machine is *vnet1*, which is used by the slice user to reach the WiMAX clients via the BTS. Each of these *vnet1* interfaces are tunneled to individual *vlans* on the *eth1* interface of the *vBTS* host machine. These *vlans* travel over the trunk and are eventually connected to the wireless clients through the modified ASN substrate as seen in Figure 2. Using *vlans* permits us to easily separate traffic from a slice at layer-2. Such a setup also helps in making the system design scalable, since these *vlans* allow us to span the *vBTS* substrate over multiple physical machines connected over LANs. From a slice user's perspective, every MAC (layer-2) frame destined to the wireless clients should be sent over this local interface in the *vBTS* environment, and similarly receive frames from the clients through the same interface.

Control Access: The third interface on each of the VMs is *vnet2*[4], which is a control interface connected via the *eth2* interface on the node. This interface allows the experimenter to orchestrate the experiment, collect results, and eventually provide means of remote monitoring using the *vBTS* management services.

[4]If physical machine which runs the *vBTS* substrate does not have more than two ethernet interfaces, functionality provided by this control interface and the first interface for GENI access can be merged.

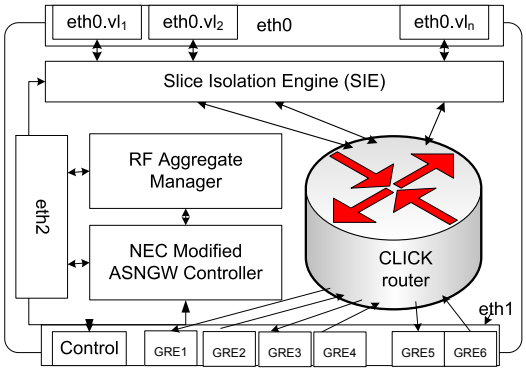

Figure 4: Modified ASN-GW substrate.

3.3 VM Grid Services

Programmability within each of the virtual machines is provided through a restful web interface which is a customized version of the OMF [5] grid service. OMF was chosen as an experiment control and management framework because it provides an easy, centralized approach to resource management. Most of the control and monitoring features supported by the OMF grid service are web-based, which can be accessed remotely through a browser interface or programmable web requests. The virtualization API exposed through the grid service allows every slice user to control virtual machine interfaces, add wireless clients to the slice, manage *vlans* and control local datapath creation, configure parameters for the clients, create custom service flows and finally also collect results from the experiment. Details of this grid service design will be addressed in a separate study. We will now discuss the modifications for the ASN-GW substrate, followed by preliminary evaluations on the platform.

4. MODIFIED ASN-GW FUNCTIONALITY

The access service network gateway (ASN-GW) performs the function of a gateway for the BTS. Depending on the profile of the WiMAX system, the ASN may also provide other functionality such as radio resource management. In this section, we will discuss modifications made to the ASN substrate to support a virtualized mode of operation with the *vBTS* substrate. The design of our modified ASN is as shown in the Figure 4. Changes are made both on the control and data path. On the data path, the ASN is modified for dynamically channeling traffic from individual *vlan* interfaces (from the *vBTS* substrate) to appropriate *gre* tunnels terminating in the BTS. Another important data path component of our virtual basestation framework is a flexible slice isolation engine that allows for control of radio resources used by individual slices. On the control path, the R6 interface is modified to change the way wireless clients are handled, thus allowing better control for data path mapping. We will now discuss the design of individual components of the modified ASN-GW in detail.

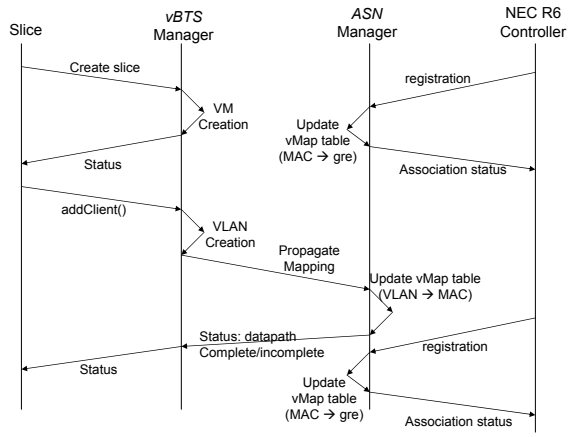

Figure 5: An example of message passing in the system. The figure depicts the interaction among components of the system when a slice adds a client, and a mobile client actually registers with the system.

4.1 Modified ASN Datapath

The ASN substrate is responsible for acting as a transparent gateway between the *vBTS*s and the actual air interface. Since all packet switching has to happen at Layer-2, we removed the IP routing from the conventional ASN-GW setup and use Click [12, 1] for frame redirection. All packet classification is based on slice identifiers, VLANids, and MAC addresses. The management architecture on the *vBTS* substrate sends the client MAC information for every slice as per individual requests to the RF aggregate manager. Typically, this information allows the ASN to bridge traffic from the individual VLAN devices (on eth0) to corresponding *gre* tunnels ending in the BTS, which transmits and receives information from the wireless clients. Using mapping information obtained from the *vBTS*, the ASN bridges appropriate *vlan* devices to the *gre* tunnels. Though the *gre* tunnels themselves are layer-3 devices, we do not use any layer-3 routing. Standard five tuple classifiers (Source IP, Destination IP, Source Port, Destination Port, TOS) can be used in addition to our switching mechanism to redirect client traffic to the correct *gre* tunnels, which represent different service classes across clients. To maintain radio isolation across clients and slices, we provide the slice isolation engine (SIE), which is briefly discussed later.

4.2 Modified R6 Control Path

The R6 Controller provided with the NEC Basestation is a proprietary software module that manages the NEC basestation and provides various funtionalities like mobility, authentication, user data handling, QoS Management. It communicates with the basestation using the R6 interface. Some part of the R6 interface is standardized in the WiMAX Network Forum. However, some interfaces and definitions are proprietary to NEC. For this reason, this module is available in a binary-only format. Extensions provided by this module are a socket based interface to notify client association and dissociation as client MAC-GRE pairs, and mechanisms for access control based on the client MAC identifiers. These extensions will help the RF Manager to setup the datapath and hence establish connectivity to the clients. We envision that a component providing this very basic functionality will be supported by most manufacturers, thereby facilitating deployment across diverse platforms.

4.3 Slice Isolation Engine

The slice isolation engine (SIE) is a traffic shaping mechanism that limits slice traffic irrespective of the clients and service classes used, such that the fraction of radio resource used by each slice are as per slice allocation policies. Though the NEC BTS supports time fairness of individual flows within the same priority class, for the sake of portability to other BTS hardware versions, and to provision weighted fairness across priority classes, our virtual basestation framework provides explicit isolation support. In addition to these features, our framework provides isolation across groups of flows (slices), which are a logical entity existing outside of the BTS.

Providing slice isolation is important since it helps to ensure the flexibility and repeatability of experiments. The slice isolation engine throttles throughput of individual slices (flow groups) based on allocated slice quotas, and estimated downlink transmission rate. In case of the slice having more than one client, we compute aggregate downlink physical rate for the slice using three strategies: (1) Simple/weighted average, (2) Median rate estimate, and a more complex (3) Feedback based slice rate estimate mechanism. For brevity, we do not discuss these in this section. The SIE runs as a closed loop control mechanism on the modified ASN that constantly estimates performance of clients belonging to individual slices, and uses this information to control downlink traffic. We use the element handlers supported by the Click [12] that allows re-configuration of preset parameters at runtime. A first generation implementation of the slice isolation engine is discussed in a previous study [10].

4.4 Basic Experiment Setup

An example of message passing between different entities in the system for slice creation, client registration, and addition of clients to slices is as shown in Figure 5. The experiment begins with the slice/user requesting for a *vBTS* creation. In response, the *vBTS* aggregate manager creates an appropriate VM instance, sets up networking from the VM to the *vlan*, and returns the status of the operation. If the *vBTS* creation is successful, the slice requests for addition of a wireless WiMAX client to its instance by specifying its MAC address. The *vBTS* aggregate manager sends this information to the RF aggregate manager that populates an entry in a mapping table. Typically, the fields in this table are: $vlan-id$, $vm-name$, $client-MAC$, $client-gre$, and *gre* direction. Out of these, the first three entries are provided by the *vBTS* aggregate manager. The $client-MAC$ to *gre* mapping information is made available by the NEC modified ASN controller to the RF aggregate manager on the ASN substrate. In the mapping table, the $client-MAC$ is used to uniquely identify every entry. When all entries are populated in a row of the mapping table, an end to end data path is automatically setup in Click [1] [5].

[5]Typically, these datapath modifications are envisioned through the use of hot swapping click kernel modules.

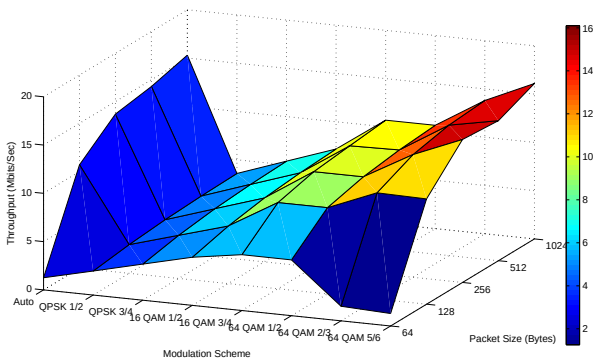

Figure 6: Baseline throughput measurements.

Parameter	*Value*
Basestation Manufacturer	NEC
Profile	A
Channel	2.5Ghz
Bandwidth	10Mhz
Default Service Flow	BE
DL/UL ratio	35:12
Traffic Type	UDP
Packet size	1024bytes
Link adaptation	Enabled
Client Chipset	Beceem

Figure 7: Default parameters used with all experiments unless mentioned otherwise in the experiment.

5. EXPERIMENTAL EVALUATION

To demonstrate the performance of our design we perform some baseline experiments. Initially, we will show the performance achievable with the basestation itself, followed by setups involving sample experiment scenarios. Experimental parameters for the NEC basestation are as shown in Figure 7, and are used as default parameters unless described explicitly in the experiment.

5.1 Raw UDP Throughput Performance

In this setup, we study the baseline downlink throughput performance of the WiMAX basestation transceiver. The client is located at a location which is very close to the basestation, though there is no line of sight link between the client and the BTS antenna. The observed CINR is 29dB and RSSI is -51dB. To evaluate the performance we measure UDP throughput for different frame sizes and different modulation and coding schemes (MCSs) used to reach the client.

Downlink throughput measurements obtained from the experiment are as shown in Figure 6. We observe that the throughput performance is the best for large frame sizes and improves with the use of higher bit rate MCSs. This behavior is justified since the client has a good connection to the BTS, which results in superior performance with higher rate MCSs. We also observe that the auto rate scheme used at the Basestation is capable of matching the performance

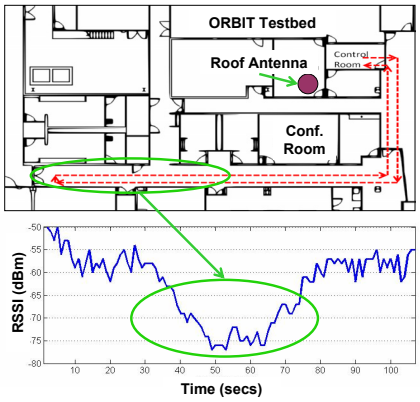

Figure 8: Experiment topology for indoor Femtocell emulation experiment. Position of stationary client is fixed in the control room, and the mobile client moves along the marked trajectory.

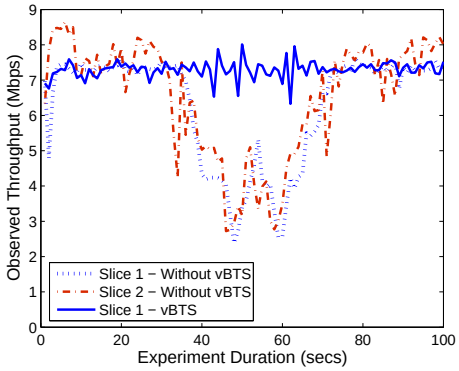

Figure 9: Throughput performance of individual slices as a function of time. Shown in the results is performance with BE service flows in each class.

achieved by static rates. This serves as a baseline test to validate performance of the autorate algorithm on the BTS.

5.2 BE-BE Slice Isolation

This experiment emulates mobility in a Femtocell deployment, and is repeated in all further indoor measurements. We consider two slices which are sending traffic from $vBTS1$ and $vBTS2$ to their corresponding clients. Client for the flow from $vBTS1$ (slice1) is stationary, and the client for the flow from $vBTS2$ (slice2) is mobile. The link from each VM to the corresponding client constitutes of a slice. The stationary client is located such that it has a CINR greater than 30 which allows the basestation to send traffic comfortably at 64QAM$\frac{5}{6}$. An experimenter walks with the mobile client as per the coverage map shown in the Figure 8. As per the RSSI trace for the walk, the link degrades in a corner of the corridor and improves as the experimenter returns to the starting position. Each slice is configured to saturate the link to its client with UDP traffic.

The observed downlink throughput for both the clients without any shaping is as shown in Figure 9. We observe

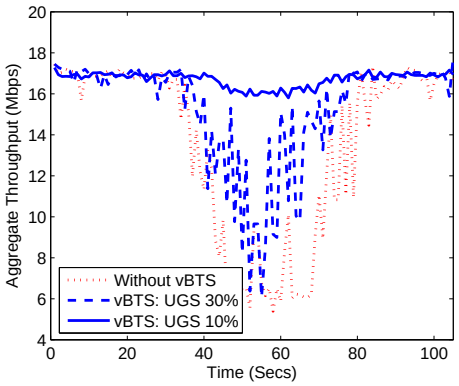

Figure 10: Throughput performance of individual slices as a function of time. Shown in the results is performance with BE service flow for the static client and UGS service flow for the mobile client.

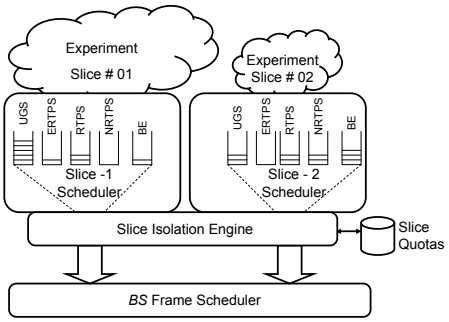

Figure 11: Hierarchical slice scheduling for allowing every slice to have a custom frame scheduler.

that as the mobile client reaches areas where the RSSI drops below a certain threshold, the rate adaptation scheme at the basestation selects a more robust modulation and coding scheme(MCS). However, in the process the link with the mobile client ends up consuming a lot more radio resource at the basestation, which affects performance of the stationary client. Thus we observe that while the BS scheduler is capable of providing QoS, it does not ensure radio resource fairness across links. The NEC BTS features a time-fair mode for providing fairness across flows of the same class, that could alleviate this condition. However, we use this case as a baseline to demonstrate the performance of our isolation engine (SIE) in the virtual basestation framework, and we will eventually show that our scheme can work across service classes and flow-groups. Results from the experiment repeated with our isolation engine are as shown in the Figure 9. Even as the channel for the mobile client deteriorates, our framework is able to appropriately limit the basestation utilization for the mobile client (slice) thereby providing fairness to the stationary client.

5.3 UGS-BE Slice Isolation

We will now extend results from the previous experiment by comparing the performance of slice isolation scheme across different traffic classes. To justify the performance we will consider an extreme condition where the slice with the mobile client is using unsolicited grant service (UGS) flows. UGS flows are typically used for voice services (VOIP) and are given priority over all other service flow types. The static client uses best effort (BE) flows, which have no reservation.

We repeat the previously discussed experiment, and plot the total throughput for both clients in Figure 10. An initial run of the experiment is performed without our framework, followed by varying reservation for the UGS flow. When we do not have any slice scheduling through our framework, the total throughput suffers when the UGS client traverses through the region with poor coverage. This is because the basestation frame scheduler tries to ensure all the traffic to the mobile client is delivered, while resulting in a very poor throughput performance for the static client. However, we notice that by using our virtual basestation framework, we are able to limit airtime allocated to the UGS client, thus improving overall throughput performance when the client traverses through the region with poor coverage.

5.4 Customized Scheduler Evaluation

Understanding and optimizing the MAC scheduling framework on the wireless edge presents an important avenue for improving end-to-end performance of specialized services such as voice, video or bulk file transfers. Customizing the MAC scheduling framework also provides a means of service differentiation which could help MVNOs to attract customers. As a part of our framework, one of the design goals was to allow the emulation of multiple MAC schedulers within different *vBTS*s. Since our mechanism allows isolation of a fixed percentage of the BTSs radio resources, every slice can use a custom scheduler to allocate these resources to their clients. The effective conceptual setup of the system is as shown in Figure 11. The design is based on a hierarchical scheduling mechanism where we have custom flow scheduling within the slice which is run as a part of the VM, possibly by an MVNO (which leases the slice) or an experimenter. At a lower layer, the slice scheduler (SIE) provided by our framework, will limit traffic from individual slices, thus preventing any inter-slice interference.

As an example, we show the performance of two flow schedulers implemented within the first slice: (1) Round robin: which alternately sends a packet for each of its two clients, and a simple proportional scheduler. (2) The proportional scheduler sends 85% of the allocated traffic to one client and the remainder packets for the other client. The first slice SLC1 has two clients: a static client, and a mobile client that follows the trajectory described in Figure 8. The second slice SLC2 has a single static client that has similar channel conditions to the static client in SLC1. Since SLC2 has a single client, it does not need a flow scheduling mechanism. We send downlink UDP traffic at saturation to each of these clients. Measured fraction of airtime used by the clients during the course of the experiment are as shown in Figure 12. We observe that when the first slice uses a simple round-robin scheduler, both clients in SLC 1 get similar airtime when channel quality is good for the mobile client. However, as the mobile client passes through the area with poor coverage, the airtime consumed by the mobile client increases, leading to a corresponding decrease in airtime available to the static client in SLC1. We observe that our slice isolation engine which is a part of the virtual

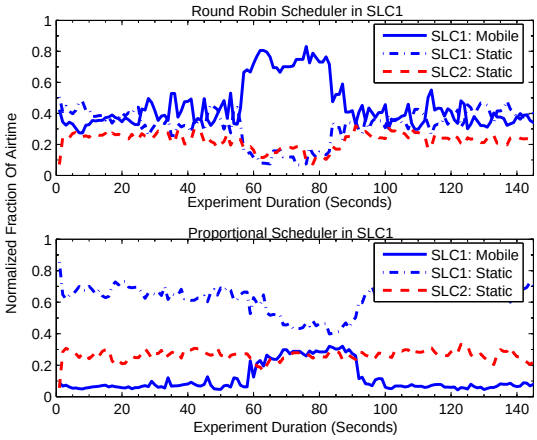

Figure 12: Airtime utilization with two custom scheduling schemes for the scheduler running within the first slice.

basestation framework succeeds in preventing SLC1 from using airtime allocated for SLC2. A similar performance isolation is seen when the experiment is repeated with a proportional flow scheduler in SLC1. Hence, this experiment shows that each of individual slices could run a custom flow scheduler without affecting performance of other slices, thus making experiments repeatable and isolated. This flexibility provided by our virtual basestation framework makes it an attractive candidate for deployment in both experimental testbeds and for hosting MVNOs.

6. RELATED WORK

Previous testbed virtualization efforts like the Planetlab [14] have primarily focussed on wired testbed virtualization. In the case of wireless devices, one of the first approaches to virtualize was proposed for short range WiFi radios in the form of virtual access points(VAPs) [9]. VAPs were proposed as abstractions on a physical AP, such that the functionality provided by the VAP was similar to that of an AP. Our approach to virtualizing the wireless radio differs from the VAP approach since the VAP is usually emulated by making changes to the device driver on the access point, thus making the solution dependent on the AP hardware (chipset) as well as the drivers needed for controlling the chipset. By building the virtual basestation as a logical entity outside of the actual commercial BTS, we make our design hardware independent.

One of the few virtual basestation designs is the VANU MultiRAN virtual basestation design [8]. The VANU design aims at running the entire virtualized BTS radio in software while our design approach relies on leveraging commercial carrier grade BTSs, and builds around them for providing virtualization. A similar approach is followed by the Open Basestation project [6] which relies on implementing a 2G BTS in software, though it does not support virtualization. Another study [16] discusses approaches in which the network architecture may be emulated using virtual machines, but does not deal with the emulation of the radio itself.

7. CONCLUSIONS

Network virtualization provides a convenient means of sharing resources across a wide set of users while allowing integration with other virtualized substrates. We present our virtual basestation framework for seamless sharing of a single physical basestation which could be implemented both by testbed operators and mobile network operators. Primarily, we discuss the approach used for emulating virtual wireless basestations to multiple slices, while also providing isolation to ensure repeatability of results. Representative results from over-the-air experiments are provided for baseline performance validation, isolation testing, and demonstration of custom flow scheduling using our framework. Future work involves a more detailed analysis of custom flow scheduling mechanisms used in conjunction with different underlying slice isolation strategies.

8. REFERENCES

[1] Click modular router. *http://read.cs.ucla.edu/click/*.
[2] GENI. *http://www.geni.net/*.
[3] KVM. *http://www.linux-kvm.org/*.
[4] Long term evolution. *http://www.3gpp.org/LTE*.
[5] Omf framework. *http://omf.mytestbed.net/*.
[6] The openbts project. *http://openbts.sourceforge.net/*.
[7] QEMU emulator. *http://www.qemu.org/*.
[8] Vanu multiran virtual basestation. *http://tinyurl.com/22lka2f*.
[9] B. Aboba. Virtual access points, ieee document, ieee 802.11-03/154r1. *http://tinyurl.com/yjjkwpv*.
[10] G. Bhanage, R. Daya, I. Seskar, and D. Raychaudhuri. VNTS: a virtual network traffic shaper for air time fairness in 802:16e slices. In *IEEE ICC - Wireless and Mobile Networking Symposium*, South Africa, 5 2010.
[11] G. Bhanage, I. Seskar, Y. Zhang, D. Raychaudhuri, and S. Jain. Experimental evaluation of openvz from a testbed deployment perspective. In *ICST Tridentcom*, Berlin, Germany, 5 2010.
[12] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3), 2000.
[13] Mike Hibler and Robert Ricci and Leigh Stoller and Jonathon Duerig and Shashi Guruprasad and Tim Stacky and Kirk Webby and Jay Lepreau. Large-scale Virtualization in the Emulab Network Testbed. In *USENIX*, 2008.
[14] L. Peterson, S. Muir, T. Roscoe, and A. Klingaman. PlanetLab Architecture: An Overview. Technical Report PDN–06–031, May 2006.
[15] D. Raychaudhuri, M. Ott, and I. Seskar. Orbit radio grid tested for evaluation of next-generation wireless network protocols. In *In proceedings of IEEE TRIDENTCOM*, pages 308–309, 2005.
[16] M. G. Rodríguez, F. G. Márquez, and E. J. Torres Mateos. A 3gpp system architecture evolution virtualized experimentation infrastructure for mobility prototyping. In *Proceedings of TridentCom*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008.