NETWORK VIRTUALIZATION ON THE WIRELESS EDGE

by

GAUTAM DILIP BHANAGE

A Dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

written under the direction of

Dr. Dipankar Raychaudhuri and Dr. Yanyong Zhang

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2011

ABSTRACT OF THE DISSERTATION

Network virtualization on the wireless edge

By Gautam Dilip Bhanage

Dissertation Directors: Dr. Dipankar Raychaudhuri and Dr. Yanyong Zhang

This thesis presents a comprehensive investigation of wireless network virtualization, a technique for creating multiple independent software-definable networks on a single set of hardware resources. Network virtualization has previously been applied to wired networking scenarios, but the general problems of wireless virtualization represents an important open problem that we address in this work. The main aspects include technical challenges, system concepts and architectures, as well as specific protocols and algorithms for implementing wireless network virtualization. In particular, this thesis addresses the following aspects of wireless network virtualization: (1) Basic mechanisms for link (spectrum) sharing with virtualized WiFi networks, (2) Virtualization techniques and traffic isolation algorithms for virtualized WiFi networks, (3) Virtualization of cellular basestations including an experimental evaluation for a prototype 4G/WiMAX virtual network, and finally, (4) an analytical evaluation of virtualization algorithms for more general multi-hop wireless topologies.

The first part of the thesis presents an exploratory discussion on the co-existence of multiple virtual networks. A comparison is presented for understanding the tradeoffs between sharing the radio through spatial and temporal separation on the ORBIT wireless testbed. Experimental evaluations reveal that while virtual networks sharing channel resources by space separation achieve better efficiency than those relying on time, the isolation between experiments in both cases is comparable. Further, we propose and implement a policy manager to alleviate the

isolation problem.

Supporting virtualized WiFi access point based networks allows for a convenient sharing of a physical access point across multiple ISPs or network operators. The second part of the thesis discusses our SplitAP architecture, which builds on the virtual access point (VAP) mechanism by extending it to support fair-sharing of airtime across multiple wireless networks. This is done by implementing a dynamically controlled isolation framework across competing slices. The framework also allows the user to deploy custom algorithms for enforcing uplink airtime fairness across client groups within the SplitAP framework. The thesis shows up to 40% improvement in isolation measured through a modified Jain fairness index with LPFC and LPFC+, two sample algorithms implemented on the framework.

The third part of the thesis addresses the challenge of virtualization of resources in a cellular basestation (BTS) while allowing operators to use distinct flow types, quota allocations, slice schedulers, 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 virtual network traffic shaping (VNTS) slice isolation mechanism allows the virtual basestation users to obtain at least an allocated percentage of the BTS resources in the presence of saturation and link degradation, helping make the performance repeatable. The fairness index and coupling coefficient show an improvement of up to 42%, and 73% respectively with preliminary indoor walking mobility experiments. Outdoor vehicular measurements show an improvement of up to 27%, and 70% with the fairness index and coupling coefficient respectively.

Finally, a theoretical formulation describes how a mapping mechanism can be used for provisioning and allocating resources on wireless networks that are supported by wireless virtualization schemes such as the virtual basestation and the SplitAP framework. Results show that the wireless mapping problem can be reduced to solving a combinatorial optimization problem at nodes selected greedily based on their capabilities to generate revenue. Results show that the proposed algorithm and infrastructure can be used to map networks with (1) both wired and wireless nodes, (2) networks with multiple sinks.

Acknowledgements

I would like to thank my advisors Dr. Dipankar Raychaudhari and Dr. Yanyong Zhang for their inspiration, encouragement and enthusiasm. The insight provided by both of them has played a big role in shaping this dissertation, and my thinking. I have been extremely lucky to have not one, but two great advisors who are equally kind, patient, and understanding. They took great efforts and provided sound advice during my research years at WINLAB. I am also thankful to Ivan Seskar, who provided invaluable advice, support and motivation, during tough times in my Ph.D. pursuit. Ivan is one of the most brilliant systems researchers I have come across, and his approach of dealing with high-pressure deadlines, and ability to single handedly execute any project will forever inspire me.

I would like to also thank Dr. Wade Trappe, Dr. Marco Gruteser, and Dr. Sampath Rangarajan for being on my thesis committee, and for their advice and suggestions regarding the thesis. Dr. Trappe has also been a collaborator on a lot of my masters thesis work, and I learnt a lot about patience and the value of taking efforts to teach from him. I have taken courses under both Dr. Trappe (queueing theory) and Dr. Gruteser (software engineering), which have both been very helpful in building my understanding of systems. Dr. Gruteser has also been very helpful in giving advice on our Hotspot virtualization project in the initial phases. Dr. Rangarajan has been very nice in hosting me at NEC Labs Princeton while collaborating on the initial phase of the virtual basestation project. I would like to acknowledge the academic and technical support provided by the Rutgers University staff, the staff at WINLAB, and Department of Electrical and Computer Engineering.

I would also like to thank my colleagues, specially my cubicle-partner Sanjit Kaul and other friends for their help which made my studies at Rutgers enjoyable and fruitful. It would not have been possible to write this doctoral thesis without support of all these people. The research presented in this thesis was supported by US National Science Foundation (NSF)

grants# CNS-072505, and CNS-0737890. I gratefully acknowledge these funding sources that made my Ph.D. work possible.

Finally, I wish to thank my parents Dilip and Pratima Bhanage, who taught me the importance of hard work and sincerity. My sister Dr.Rima Chaudhari and her husband Dr.Mayuresh Chaudhari have always instilled the courage in me to finish my studies. I would also like to thank my uncle Dr. Pradeep Shirodkar and aunt Ellen for motivating me, taking the efforts to meet me, and guiding me whenever I asked for his advice.

Above all of these, I would thank my wife and the love of my life, Smruti Lele, for her endless encouragement, without whose selfless support this work would have been impossible.

Dedication

To my wife, Smruti,

my family,

and

my friends.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 What is Network Virtualization?

The term *virtualization* originates from the area of server systems virtualization, where multiple virtual machines are emulated on a single physical machine. It refers to a mechanism for seamlessly sharing an underlying physical resource while providing complete isolation across the user base of the system. Network virtualization technology is used for extending these ideas of virtualization to networks. This dissertation specifically addresses the design issues and technical challenges of *virtualization* of wireless networks. Before we delve into the advantages, and details of wireless virtualization, we present a brief background on the field of network virtualization.

1.1.1 Background

Wired Virtual Networks: Among the very first applications in networks, virtual local area networks (VLANs) [1] were created in switches, which behave like independent ethernet networks connected to the same hardware. Such a mechanism allows the network operator to administer multiple independent LANs or broadcast domains on the same physical network. Such an approach helps to improve the manageability of the underlying network. Virtual private networks (VPNs) [2] use tunneling as a mechanism to emulate MAC frames on a remote network as if coming from a client on the local area network. This technique is used by enterprises to allow employees to log in to the company networks remotely.

In terms of network architectures, network virtualization was introduced to the wired world with the primary purpose of sharing the underlying physical network across virtual topologies

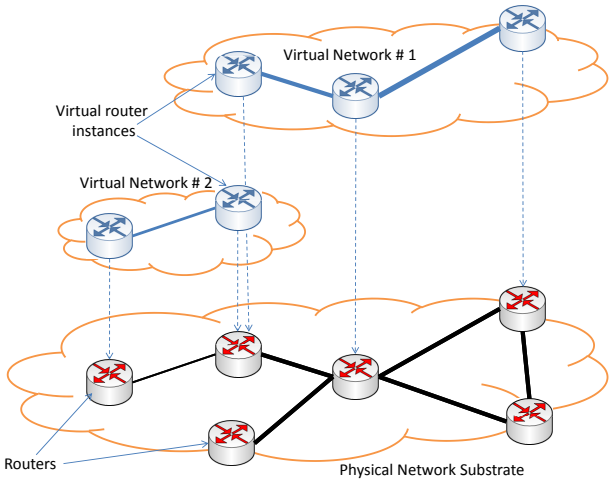


Figure 1.1: A wired virtualized network architecture. This figure shows how independent virtual networks can share the underlying wired infrastructure.

while permitting significant degree of customization across an independent set of users. A virtualized wired network architecture looks like the example shown in Figure 1.1. The underlying physical network is shared by the virtual networks, such that all the components in the virtual networks are some subset of the physical network, and the sum of resource requirements (such as link bandwidth, node computation capacity) across all virtual networks do not exceed the value that can be supported by the physical network.

Some of the previous studies discussing applications and deployments of virtualized frameworks are as follows. The X-Bone [3] architecture was proposed for rapid deployment of overlays which was used for the proposed virtual internet design [4]. The Tempest architecture [5] is a control framework that describes how multiple control architectures can be used to run over a single ATM architecture. Similar ideas have been proposed in the CABO project [6] which relies on virtualization for supporting multiple concurrent architectures. These ideas have also been put to use for conventional wired testbeds such as the Planetlab [7] and VINI [8] frameworks. Independent algorithms have also been proposed for mapping such virtual networks to wired substrates [9]. Architecture and mechanisms for making a framework based on virtual routers fault tolerant is discussed as a part of VROOM [10]. All of these ideas have been pre-dominantly considered for wired networks. We will present a discussion on the impact of extending virtualization to wireless networks.

Wireless Virtual Networks This dissertation systematically extends the ideas from wired network virtualization to wireless networks while addressing the underlying challenges arising due to the nature of the wireless medium. We define *virtual wireless networks* as logical entities that are provisioned on an underlying shared physical substrate, which can provide seamless and transparent connectivity to their clients. In this case, the term transparent is used to indicate that the clients of the system are oblivious to the fact that both the virtual networks being mapped and their wireless clients are oblivious to the fact that the radio hardware is shared.

Some previous studies have tried to address specific aspects of wireless virtualization. They are as follows. A study on virtualizing commodity wireless devices [11] proposes enhancement to the ORBIT radio grid, through the use of a time division multiplexing scheme for scheduling networks. This study provides a motivation for challenges in scheduling and context swapping virtual networks. Our approach is orthogonal to this study, in that we share the inherent time sharing nature of the MAC, and build mechanisms on top to ensure isolation across slices. The Multinet [12] project discusses an architecture for supporting a virtualized client connection to multiple networks from a single wireless client. Our setup discusses a solution for virtualized networks by splitting the AP.

As can be seen from these, the field of wireless network virtualization is very new. In the next section we will define use cases to justify the need for virtualized wireless networks.

1.2 Why do we need to virtualize Wireless Networks?

We begin with a presentation of three strong motivation case studies for supporting wireless network virtualization, followed by a detailed discussion on our contributions.

1.2.1 Case 1: Shared Commercial Wireless Networks

Recent years have seen a dramatic surge in the number of mobile subscribers with the number crossing 4.6billion in Feb 2010, which is approximately 67% of the world's population [13]. Further, some research studies see this number crossing 5.9 billion by 2013 [14]. With large number of users from various backgrounds using cellular networks, there is a tremendous potential for launching niche services through mobile virtual network operators (MVNOs). These

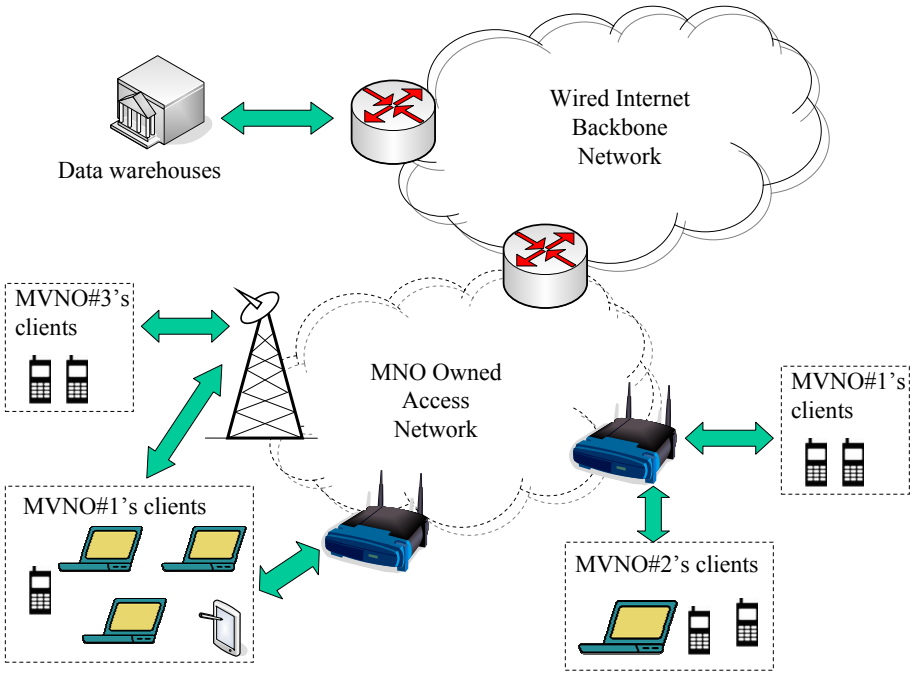


Figure 1.2: A shared heterogeneous network architecture. This figure shows how independent MVNOs and ISPs could possibly lease the wireless infrastructure for providing service to their clients.

MVNOs lease the network from the mobile network operator to provide their services. Currently, the network operators achieve this by leasing basestations and spectrum to these operators. However, we observe that it would be much nicer if the basestations itself, and the wireless spectrum on each basestation could be shared among multiple MVNOs. Doing this allows the mobile network operator to maximize profit by reducing the amount of hardware deployed (where possible) and also lowers costs for the MVNOs leasing the hardware.

Similarly, recent studies have shown that WiFi is used by over 700 million people, and there are over 750,000 hotspots (places with WiFi internet connectivity) around the world, and about 800 million new WiFi devices every year [15]. In such scenarios, it will be very useful if we are able to share some of these public WiFi hotspots among multiple internet service providers. Specially, in areas such as airports, hospitals, train stations, where the number of deployed devices would possibly be limited by space or by the availability of spectrum, sharing hardware provides a good solution. Additionally, these hotspots could possibly be used by mobile network operators or MVNOs to provide supplementary data service to their mobile

subscribers with smart phones.

Once we are able to share these WiFi hotspots and cellular basestations across MVNOs and ISPs, we envision the wireless architecture to look similar to that shown in the Figure 1.2. As shown, depending on the agreements between the MVNOs and the network operator who owns individual points in the infrastructure, wireless clients belonging to different MVNOs can connect to appropriate shared points on the infrastructure.

1.2.2 Case 2: Large Shared Wireless Testbeds

Apart from this application of sharing basestations and hotspots across MVNOs and ISPs, appropriate sharing mechanisms could also be used for implementing large shared testbeds such as GENI [16], and also for provisioning and providing leased services (such as video on demand), through the core network. The GENI initiative in particular envisions a nation-wide wired experimental facility with numerous shared wireless edges. The initial ideas in this design are drawn from existing testbeds like Planetlab [7]. However, the issue of sharing wireless devices for experimentation purposes is still largely a topic of research. It will be beneficial if the wireless components of the testbed are able to support this shared model of usage along with their wired counterparts.

1.2.3 Case 3: End - to - End Protocol Stack Customization

The current architecture of the internet supports different functionalities through the use of overlay services. However, as the number of users and the diversity of their applications keep increasing, at some point the architecture will not be able to address all the requirements at scale through the use of a single protocol stack [17, 18]. This problem is further aggravated when we also wish to cater to diverse need of wireless networks through a common protocol stack. For example, a common incremental solution on IP is not ideal for supporting diverse wireless systems such as mesh networks [19], peer - to - peer systems [20], delay tolerant networks [21], vehicular networks [22], and sensor networks [23][1]. Rather in such a case, it is envisioned that both the wired and the wireless parts of the network will eventually provide support for running

[1]These emerging wireless scenarios are also responsible for motivating research projects on clean slate protocols like the NSF FIND program, and the European FP7 program.

multiple simultaneous protocol stacks (which are possibly different from layer-3 and up), and these independent stacks will cater to different network requirements.

1.2.4 Design Requirements For Shared Networks

Based on the motivation from the three leading application use cases: (1) sharing commercial wireless networks and (2) wireless testbeds, and (3) supporting end - to - end customized protocol stacks, we can summarize a set of core features in the infrastructure that would make the sharing simpler, and seamless:

- Transparency: The slicing of the radio to be shared has to be transparent to the user of the system. From the client perspective this means that the clients need not be aware of the sharing of the infrastructure. From an MVNO perspective, this means that each of the MVNOs should not be concerned with the fact that the radio is shared i.e. they should have an interface that is similar to that enjoyed by them when they lease the entire radio. The transparency feature is also a big plus from the testbed deployment perspective since it allows for easy porting of prototypes from the wireless testbed to actual network hardware.

- Service differentiation through customization: The architecture of the shared radio interface should allow the slices (groups of users[2], in this case the MVNOs or the ISPs) of the system, to customize their interaction within their slice.

- Performance Assurance: The sharing should be such that the users of the system are able to use the system oblivious to each other. Such a mechanism ensures that some service level agreements (SLAs) are satisfied between the network provider and MVNOs. Similarly, in case of testbeds, performance assurance across multiple experimental slices ensures repeatability of results.

Through the remainder of this chapter, we will discuss how these requirements are satisfied through wireless network virtualization.

[2]The terms *slices* and user groups will be used interchangeably through the rest of this dissertation.

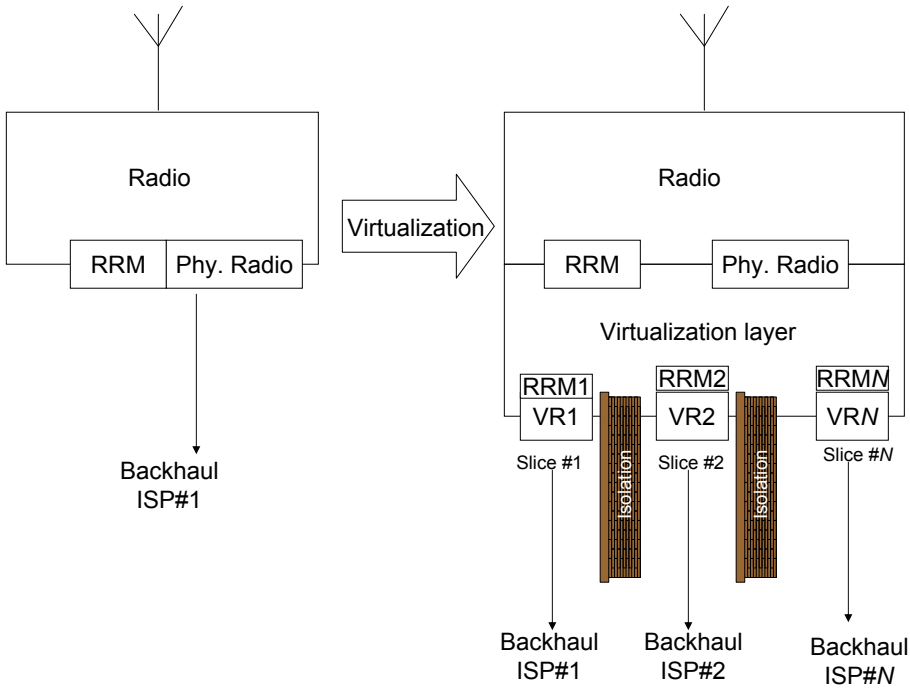


Figure 1.3: Virtualization applied to a general purpose radio. After virtualization, we observe that every slice will see independent virtual radio (VRs) interfaces where the radio resource management can be done by individual slices.

1.3 Wireless Virtualization Based Solution

Wireless network virtualization is a design paradigm that allows the network operator to support multiple virtual wireless networks on wireless infrastructure. All the requirements described in the previous sections can be satisfied through the wireless virtualization methodology. The transformation required to conventional wireless networks for supporting virtualization is as shown in the Figure 1.3. In order to virtualize a component of the network, we need to extend the basic principles of (1) abstraction, (2) programmability, and (3) isolation to both the radio devices and the network as a whole. Thus for virtualization, we add an extra layer over the MAC layer of the radio, which abstracts the single physical radio device, and emulates multiple virtual interfaces to the upper layers of the network stack. This allows every MVNO (or ISP or slice) to customize its interaction with the interface. The virtualization layer also provides isolation between each of the virtual interfaces, thus allowing them to operate and use the physical radio independent of each other. Programmability is provided by exposing part of the radio resource

management (RRM) functionality to each of the slices and allows them to customize their share of the radio.

1.3.1 Technical Challenges

Enforcing virtualization at the wireless edge poses several challenges. Through the remainder of the thesis we will discuss how the underlying design challenges are met by our proposed virtualized architectures:

- Generality of design: Beginning with the determination of which wireless network components need to be virtualized, the eventual prototype's design has to be simple and generic enough to permit portability to other hardware devices. E.g. virtualization guidelines and design laid down for a mobile WiMAX basestation have to be generic enough to permit easy portability across multiple manufacturers adhering to the 802.16e standard.

- Wireless Infrastructure Abstraction issues: While abstracting the underlying resource, if any extra virtual machines are used, they should export suitable features for supporting control features similar to those supported by the underlying resource. E.g. In case of a virtualized WiMAX basestation, the framework should be capable of emulating virtualized instances of basestations that support some subset of functionality supported on the physical basestation. Such a feature allows for more detailed experimentation setup in case of a testbed usage scenario, and permits better customization in the MVNO use case. This problem specifically requires more care in wireless networks due to widely varying nature of layer-2 wireless devices, as compared to those on the wired side.

- Isolation issues: This thesis also presents detailed design and prototyping of mechanisms for facilitating slice isolation. This problem is specifically harder in wireless networks due to the rapidly changing nature of the wireless medium. E.g. When we virtualize a wireless access point (discussed in Chapter 3), changing client channel conditions either due to mobility should not impact other co-existing slices at the access point.

- Protocol independence: One of the design goals which make the virtualization methodology particularly important is that it allows us to slice the networks in such a way that

they can be made protocol independent. In our case, we show that by virtualizing wireless devices at layer-2 we can make the system protocol independent from layer-3 and up. E.g. In our virtualized basestation design, we show that special mechanisms are needed for supporting L2 frame forwarding on virtualized basestations.

- Co-existence/Spectrum reuse: Even if we are able to virtualize and share individual network devices, we want multiple virtualized and non-virtualized networks to be able to co-exist in certain environments. E.g. While virtualizing the ORBIT testbed, we would want two or more virtual networks mapped on the testbed to share wireless spectrum (discussed more in Chapter 2). This problem is also harder in wireless networks since it is easier to provide isolation across networks in the wired case.

- Virtual Wireless Resource Mapping: Finally, after the issues in virtualizing wireless devices, and co-existence of multiple virtualized networks has been discussed, it is important to understand how a network operator will be able to map virtual network to the physical substrate. This problem is also harder in case of wireless networks due to complex interactions arising due to interference and hidden nodes in the network.

1.3.2 Methodology

To address the technical challenges described above, this dissertation describes how the concepts from systems virtualization can be extended to wireless networks for providing a mechanism for seamless sharing across diverse user sets. Specifically, it will begin with a discussion on the impact of multiple networks sharing the spectrum by comparing the time division and space division approaches. Using this insight, we propose the SplitAP framework for sharing public access points among groups of users (belonging to different ISPs). In a similar space, the virtual basestation framework discusses the efforts in design and prototyping. Attention is focussed on mechanisms used for platform independent, and indirect means for radio resource provisioning, using traffic shaping. Finally, a theoretical model and mapper are shown for provisioning virtual networks on top of existing physical networks.

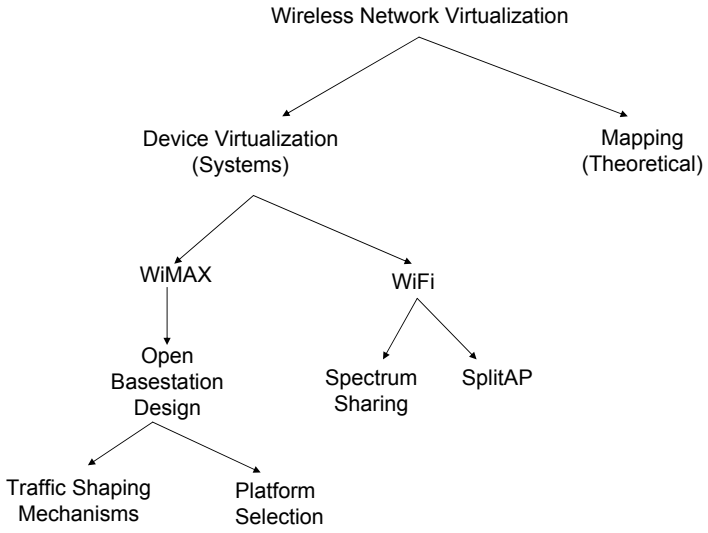


Figure 1.4: Dissertation layout.

1.4 Contributions And Dissertation Layout

This thesis is organized as shown in Figure 1.4. Chapter 2 describes the performance achieved when we have multiple co-existing wireless mechanisms separated in space and time, with no radio resource provisioning mechanisms. This part will focus on measuring performance when virtualization is used for sharing of the wireless spectrum rather than the nodes themselves. Chapter 3 will describe the design and implementation of a virtualized 802.11 access point on the ORBIT indoor wireless grid. Chapters 5, 4 discuss the different aspects in the design of a virtualized WiMAX basestation. Chapter 6 describes mechanisms for mapping virtual wireless points of presence (PoP) requests to the underlying physical wireless mesh network. Finally, Chapter 7 discusses conclusions and future directions. Chapter 8 described in the Appendix shows different aspects in the selection of platforms for virtualized wireless systems.

1.5 Previously Published Material

Chapter 2 revises a previous publication [24]: R. Mahindra, **G. Bhanage**, G. Hadjichristofi, I. Seskar, D. Raychaudhuri, and Y. Zhang. Space versus time separation for wireless virtualization on an indoor grid. In proceedings of NGI, pages $215 - 222$, April 2008.

Chapter 3 revises a previous publication [25]: **G. Bhanage**, D. Vete, I. Seskar, and D. Raychaudhuri. SplitAP: leveraging wireless network virtualization for flexible sharing of WLANs. In IEEE Globecom 2010 - Next Generation Networking Symposium (GC10 - NGN), Miami, Florida, USA, 12 2010.

Chapter 4 revises a previous publication [26]: **G. Bhanage**, I. Seskar, R. Mahindra, and D. Raychaudhuri. Virtual Basestation: architecture for an open shared wimax framework. In ACM Sigcomm conference, VISA workshop, New Delhi, India, 09 2010.

Chapter 5 revises a previous publication [27]: **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.

Chapter 2

Wireless Medium Separation and Integration

2.1 Summary

The decreasing cost of wireless hardware and ever increasing number of wireless testbeds has led to a shift in the protocol evaluation paradigm from simulations towards emulation. In addition, with a large number of users demanding experimental resources and lack of space and time for deploying more hardware, fair resource sharing among independent coexisting experiments is considered important. We study the proposed approaches to wireless virtualization with a focus on schemes conserving channels rather than nodes. Our detailed comparison reveals that while experiments sharing channel resources by space separation achieve better efficiency than those relying on time, the isolation between experiments in both cases is comparable. We propose and implement a policy manager to alleviate the isolation problem and suggest scenarios in which either of the schemes would provide a suitable virtualization solution.

2.2 Introduction

The GENI Project [16], supported by NSF, aims to provide a flexible, programmable, shared experimental infrastructure for investigation of future Internet protocols and software. GENI will consist of a global-scale wired network with programmable and virtualizable network elements (routers, switches, servers) along with several wireless access network deployments intended to support experimentation with mobile computing devices, embedded sensors, radio routers, etc. This project is aimed at finding solutions to the virtualization of wireless network resources to provide capabilities for simultaneous support of multiple concurrent experiments ("slices") on the same set of radio devices.

The ORBIT [28] is a 400 node wireless testbed sponsored by NSF for indoor wireless

experimentation. It is a multi-user radio grid that allows “sequential” short term access to the radio resources for repeatable experiments. Time scheduling is done so that users have exclusive access to the grid during their slot. Excessive usage leading to lack of available time slots in the light of the GENI initiative has further motivated efforts for ORBIT virtualization. Thus, virtualization in the ORBIT context refers to the ability of splitting the wireless testbed resources among multiple concurrent experiments with each experimenter controlling a ”slice” of the entire radio grid.

Suitableness of a wireless virtualization scheme is decided by:

- Resource Constraints: Different virtualization schemes can help conserve different resources (number of nodes, available orthogonal channels, ability of the experiment control mechanism to handle parallel experiments).

- Efficiency: Sharing a resource by virtualization introduces additional overheads for the management framework. For example, in case of a UML [29] based approach to virtualization excessive resource utilization may be seen in the form of context switching. The virtualization scheme should be efficient such that there is minimal management overhead, since it eventually decides the maximum number of parallel experiments.

- Inter-experiment interference: Virtualization of any resource almost always results in some form of compromised performance for co-existing experiments. While mapping virtualization to scientific experiments it is necessary to quantify any performance degradation of experiments.

- Experiment Repeatability: An important aspect with performing indoor controlled experiments is to ensure the repeatability of results. Improper resource sharing may result in unpredictable performance across multiple experiment runs.

A wide range of wireless virtualization schemes have been proposed [30]. We select and compare two approaches to wireless virtualization: space separation and time separation; based on their suitableness for deployment on the ORBIT testbed. Empirical evaluation of sample scenarios are used for comparison and deduction of overheads with wireless virtualization. Despite having minimal overhead in terms of CPU utilization in both approaches, we show the

usefulness for a policy management mechanism that dynamically allocates channel resources for experiments.

The contributions are as follows:

1. Compare strengths and drawbacks of space and time based virtualization among other schemes and determine their suitableness for deployment on a type of wireless testbed.

2. Provide empirical measurements from a systematic setup and use them to determine efficiency of the virtualization schemes.

3. Propose metrics to compare interference between experiments and provide an implementation of a Click based policy manager. We show that this policy manager makes it is possible to select a higher efficiency virtualization scheme irrespective of the level of inter-experiment interference.

This research is based on devices that use the same MAC and physical layers. All devices in the virtualization schemes use MAC and PHY layers that are compatible with the 802.11 standard. Our study does not aim to provide a comprehensive virtualization solution across heterogeneous wireless devices and drivers, but can serve as a reference to show the trends in performance that may be observed with the use of the two aforementioned virtualization approaches. This study lays out the criteria, which could be used for deciding virtualization schemes on testbeds. We believe that our study lays the foundation for some of the key design issues and deployment strategies for wireless virtualization on large-scale network testbeds like GENI [16].

The rest of the chapter is organized as follows. Section 2.3 describes some of the important approaches of wireless virtualization. Section 2.4 and 2.5 compare the performance difference between SDMA and VAP-based virtualization schemes. Section 2.6 discusses inter-experiment effects that may be seen in channel multiplexed wireless virtualized schemes. Finally, in Section 2.7, we propose a policy manager for ensuring isolation between virtualized experiments.

2.3 Virtualization Schemes

Wireless virtualization approaches may be conveniently classified along the space, time, and frequency axes as:

- Frequency separation channel sharing
- Space separation channel sharing
- Time separation channel sharing

Before providing an overview of these approaches, we briefly describe our experimental testbed.

2.3.1 Virtualization Platform

ORBIT is a two-tier laboratory emulator/field trial network testbed designed to achieve reproducibility of experimentation, while also supporting evaluation of protocols and applications in real-world settings. The laboratory-based wireless network emulator uses a novel approach involving a large two-dimensional grid of 400 802.11 radio nodes which can be dynamically interconnected into specified topologies with reproducible wireless channel models. The majority of the ORBIT nodes are fitted with Atheros 5212 based cards while the remaining have Intel cards. We used Atheros cards for our experiments.

2.3.2 Outline Of Virtualization Approaches

The author in [30] presents virtualization techniques that are intended to share a set of wireless resources amongst multiple users.

Frequency Separation Channel Multiplexing: Frequency separation implies partitioning of the experiments in the frequency domain with different experiments assigned orthogonal channels to prevent interference. Multiple experiments are executed on the same physical nodes, with each experiment being executed in an instance of an OS or virtual OS. Thus, the resources of a physical node are split into multiple virtual nodes. This virtualization would introduce a finite channel switching delay when switching from one virtual node to another. In most facilities, there is a provision for multiple wireless interfaces. Hence, individual experiments could

be mapped to different wireless interfaces on the same physical node eliminating switching delays.

Space Separation Channel Multiplexing: The space separation approach splits the testbed resources to provide sufficient spatial separation between the transmitting nodes and avoid interference across individual experiments. During this allocation, a subset of the physical resources is assigned to a specific experiment. This means that space separation provides virtualization across multiple nodes eliminating the need for experimenters to share experiment nodes. Space separation will be broadly referred to as space division multiple access (SDMA) scheme.

Time Separation Channel Multiplexing: Time separation or Time division multiple access (TDMA) virtualization partitions the network in time domain across multiple experiments. In [11], the authors propose the assignment of unique time slots during which all virtual nodes corresponding to a specific experiment use the wireless channel. Time sharing of the channel is discussed in further detail in subsection *E*.

2.3.3 Most Suited Approaches

Selection of a virtualization scheme primarily depends on the resource being conserved. Wireless virtualization can be targeted at either the conservation of nodes (hardware) or channels (frequency). Frequency multiplexing of the wireless channel, allows for node conservation where the same node could be shared using a UML [29] like mechanism on multiple channels to emulate different experiments. Keeping in mind Moore's Law, the concern for deployment of a virtualized testbed would be more on channel conservation with availability of relatively cheaper commodities. For instance, with access to 800 wireless interfaces on the ORBIT grid the focus was more on channel conservation rather than node conservation. Frequency multiplexing may not scale well with provision of only three orthogonal channels in 802.11b mode of experimentation. Since time and space separation allow for channel conservation we will compare and contrast these approaches for selecting an apt virtualization setting.

2.3.4 Space Separation on ORBIT

The ORBIT wireless testbed is located in a 20 meter x 20 meter space and hence the nodes are in close physical proximity of one another. Under these conditions, partitioning the resources in space to avoid interference would not be practically possible. This holds true for most of the emulator testbeds. Artificial stretching of the distance between the nodes is achieved by controlling transmission power of the nodes and using "noise injection" to emulate barriers between the nodes of different experiments. Observation from previous studies [31] reveals that it is considerably difficult to create and limit the effect of noise locally with the current noise injection subsystem on ORBIT. Our experiments explore the possibility for virtualizing the ORBIT grid using SDMA by controlling power in addition to providing spatial separation. In this artificially stretched SDMA scheme, the individual experiments are multiplexed on the same channel.

2.3.5 Time Sharing On ORBIT

Time sharing on the ORBIT wireless grid can be achieved using two approaches:

- Explicit TDMA implementation
- Virtual access points (VAP)

TDMA: TDMA has been implemented and tested on the ORBIT grid in [29]. This approach runs multiple UML instances on the same node, which use the same wireless device. They ensure through tight synchronization, that at any time all the nodes are running the same experiment slice across the network of nodes.

Efficiency of implementation and overall performance seen with a TDMA scheme will greatly depend on:

- Experiment Synchronization: In TDMA, there is a stringent need for high degree of time synchronization between all the experiment nodes. Moreover, wireless experiments can potentially involve a high number of experimental nodes. Though tools like the network timing protocol daemon (NTPD) [32] can provide distributed time synchronization, accuracies achieved may not be sufficient.

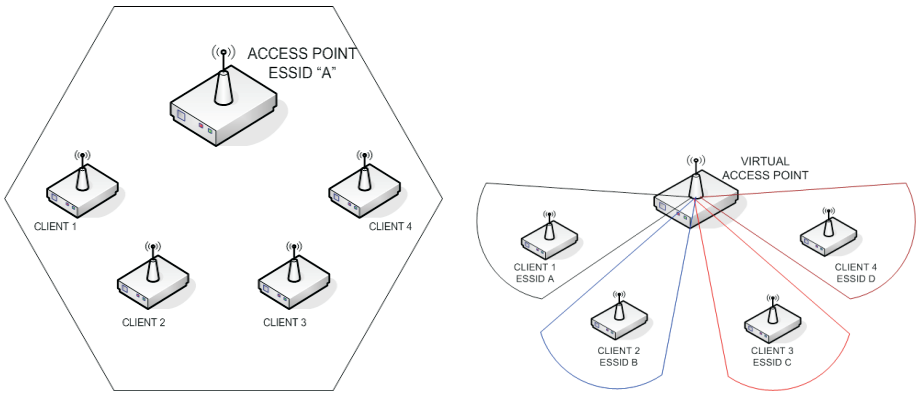
- Design Dilemma: The choice of time slot allotted to the different experiments is another design issue for the TDMA approach. A small value may not be possible due to practical limitations of wireless hardware like switching time and a large value would adversely affect the performance in delay sensitive experiments. Since, in this approach, several concurrent experiments share one or more physical nodes, there is also a need to provide isolation on every node between the experiments.

The TDMA approach requires design and deployment of a complicated infrastructure on current testbeds like ORBIT which does not seem plausible. To offset these disadvantages we consider the use of virtual access points as a mechanism for channel multiplexing.

Virtual Access Points (VAPs) A VAP is defined as a logical abstraction that runs on a physical access point while emulating the behavior of a conventional access point to all the stations in the network [33]. Using a VAP allows for two or more AP mechanisms to share the same channel thereby helping channel and energy conservation. In contrast to the TDMA approach for space separation channel multiplexing, VAPs are more suitable for running short-term experiments with less stringent constraints on the current testbed resources.

The concept of VAPs is incorporated in the 802.11 driver, which operates just above the MAC layer and below the IP layer. The driver provides the multiple AP abstraction to the higher layers though it is operating on a single lower layer. Hence all the protocols operating on the machine are agnostic to the presence of the abstraction. As we will show in the coming sections this setup can be extremely useful for minimizing down link interference with multiple infrastructure mode setups. We plan to use this mechanism to provide virtualization of fixed star topology wireless networks.

Since channel conservation is of prime importance, we choose to evaluate the space and time separation approaches for virtualization on the ORBIT grid. As the VAP approach provides a more plausible solution to time multiplexing over conventional TDMA approaches intended for long term experimentation, we will use if for further quantitative evaluation with space separation.



(a) A physical access point and four clients

(b) Four virtual access points and their individual clients

Figure 2.1: Experimental setup for performance evaluation with physical and virtual access points.

2.4 Throughput Comparison

Throughput, latency and jitter are usually the three main parameters, which determine a users utilization and experience on a network device. Throughput for individual experiments in a virtualized environment is expected to be lesser than those under single user conditions. However, performance under these conditions is largely contingent to how fairly the resources are shared.

A virtualized channel is likely to see multiple experimenters running simultaneous experiments and the end performance can largely be a function of individual experiment parameters rather than just a fair share between users.

Prior to investigating and comparing VAP and SDMA based virtualization schemes, we discuss briefly about the implementation and working of a VAP, which is a relatively new concept. This study should be insightful in determining if a VAP buys us significant advantages over a conventional physical access point setup.

2.4.1 Virtual Access Point Overhead

A VAP creates an abstraction of multiple physical access points running from the same hardware for the clients associating with it. Creation of these logical entities requires state maintenance and independent management signaling for each of the networks managed by each VAP.

Parameter	*Value*
Channel Rate	36Mb/sec
Aggregate Offered Load	50Mb/sec
Experiment Duration	5 Minutes
Averaging Duration	Per Second
Operation Mode	802.11a
Traffic type	Uplink
Chipset	ATHEROS
Driver	Madwifi(0.9.3.1)

Figure 2.2: Experimental Parameters Used With ORBIT Nodes

Before we evaluate the benefits of using VAPs, we consider it important to determine the overheads of maintaining the state of multiple networks at a single hardware device. The experimental setup for comparison is as shown in Figure 2.1(a) and Figure 2.1(b). Figure 2.1(a) shows a setup with one AP and all four clients within the same network. Figure 2.1(b) has the same nodes. However, each of the clients now belongs to a different logical network created by the VAPs. Care was taken to ensure that there is no capture within the network by choosing client nodes such that they had comparable RSSI at the access point. Results are evaluated for both uplink and downlink performance with a saturated channel and equal offered load per client. Other experiment parameters were maintained as shown in Figure 5.1.

Figure 2.3 plots the observed per client throughput ($\frac{Mbits/sec}{client}$) for uplink and downlink traffic. Performance of a single client with a single access point is taken as a reference for comparison. Key observations that can be made from the results are:

- As with any time sharing approach, the entire bandwidth (which is seen in the scenario with 1 client) is now shared across 4 clients.
- Uplink traffic sees a slight deterioration in performance with both the AP and the VAP as compared to the reference flow with 1 client.

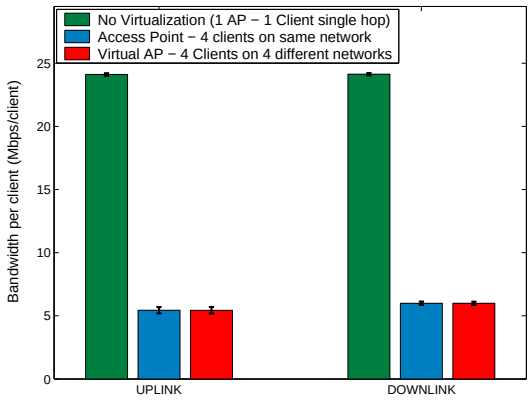
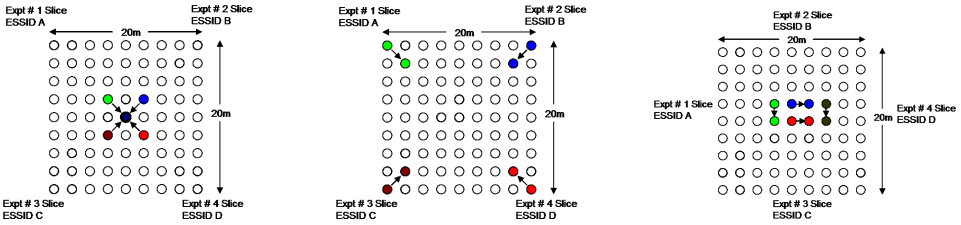


Figure 2.3: Impact of virtualizing using channel multiplexing approaches.



(a) Topology for VAP-based virtualization.

(b) SDMA scenario with maximum spatial separation between experiment nodes

(c) SDMA scenario with experiment nodes placed close together

Figure 2.4: Experimental setup for performance evaluation of VAP and SDMA schemes on ORBIT.

- There is no added deterioration with uplink traffic using VAPs for having clients on multiple networks, as compared to an AP with all clients in one network. Hence, we can conclude that the deterioration seen in both cases which leads to a net channel throughput of $21.76 Mbps$ as compared to $24.11 Mbps$ is due to the increased channel contention overhead.

- Downlink overheads for both AP and VAP with 4 clients are neglibible as compared to that with a single client.

- Error bars for both cases show little variance in throughput.

Hence we can conclude that using a VAP adds no conspicuous overhead to the throughput performance of a convention AP. We confirmed this behavior by investigating the source code for the MADWifi [34] driver where the VAPs are created. The driver does minimal additional

processing to differentiate between the packets received for the different virtual interfaces. The above study suggests that experiments evaluating aggregate throughput with test setups running a single AP or multiple VAP should generate comparable results with the channel utilization being determined by the number of clients. Based on this conclusion, we can now compare the performance of virtualization with VAP and that with space separation based on:

1. Offered load
2. Packet sizes

2.4.2 Variation With Offered Load

Performance comparison of the VAP versus space separation (SDMA) uses the experiment setup as shown in Figure 2.4(a) and Figure 2.4(b). We compare the performance of both virtualization schemes by mapping four co-existing experiments. Each individual experiments consist of an AP-client single hop wireless.

Figure 2.5 shows the results for the aggregate throughput for virtualized experiments with varying offered load. We observe that below saturation both SDMA and VAP have the same performance. However, as the offered load is pushed into the saturation limits of the channel, there is a clear difference in the throughput.

The difference in performance observed in Figure 2.5 is due to the physical layer capture [35]. Capture is the phenomenon by which a receiver correctly decodes one of the many simultaneously colliding packets due to relatively high signal to noise ratio. Physical layer capture was detected either by sniffing packets from the channel with multiple sniffers (since the sniffers themselves are susceptible to capture) or by comparing the number of MAC retries with a case without capture. Figure 2.6 shows the the aggregate number of MAC retries with the VAP and the SDMA case. It is clearly seen that the number of MAC retries with SDMA were significantly lesser than with VAP since the receivers are able to decode colliding packets due to capture.

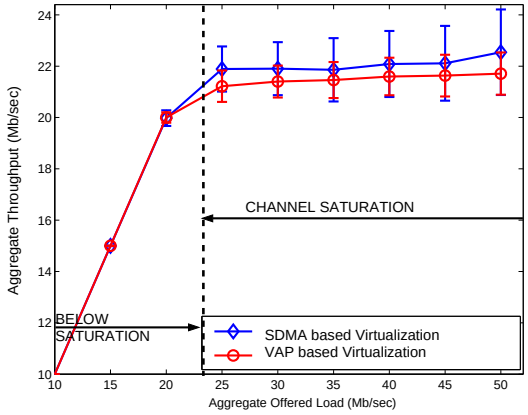


Figure 2.5: A comparison of available bandwidth for SDMA and VAP based virtualization schemes supporting four concurrent experiments.

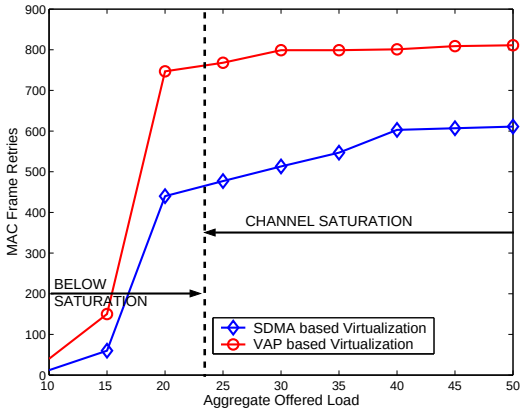


Figure 2.6: A comparison of number of MAC frame retries for SDMA and VAP based virtualization schemes supporting four concurrent experiments.

2.4.3 Variation With Packet Sizes

Packet sizes in a saturated channel impact both the MAC and physical layer overhead, as well as the aggregate channel access time. The goal of varying the packet sizes with experiments is to test if they have similar effect on performance with both the VAP and SDMA approach.

The setup of these experiments is the same as shown in Figures 2.4(a) and 2.4(b). To determine the effect of node positioning on the capture effect with SDMA, we measure SDMA performance with two setups as shown in Figures 2.4(b) and 2.4(c). In Figure 2.4(b) the nodes of the experiments are setup far from one another. In Figure 2.4(c) the experiments are setup next to each other. For each experiment packet sizes were varied and the aggregate throughput

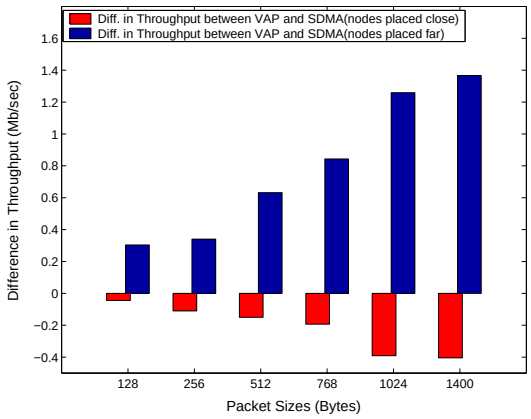


Figure 2.7: A comparison of available bandwidth for SDMA and VAP showing the effect of space and transmission power control.

was measured. In Figure 2.7, we plot the difference in throughput of each of the SDMA setups from the VAP experiment and show the performance gains.

The general trend for both setups follows intuition where performance is poor for small packet sizes and vice versa. However, SDMA setup with nodes placed far away had the advantage of decreased interference and improved performance with higher capture. The positive increase in difference in throughput shows that the benefits of capture increase with packet sizes. The SDMA setting without spatial separation shows a degraded performance as compared to the VAP setting. The MAC-ACKS in the downlink see lesser interference and collisions in the VAP due to time scheduled downlink transmission and hence the setting has a better performance as compared to the SDMA without spatial correlation. As the packet size increases this difference is even more pronounced since the effect of a collision is more pronounced for larger packet sizes.

2.5 Delay-Jitter Comparisons

Experimenters often use delay as a metric measured for performance of an experimental setup. Jitter, defined as the variation of delay is also an important metric in the performance of real time traffic, such as voice or video. We will compare the effect of time and space separation for virtualization on both observed delay and jitter per experiment.

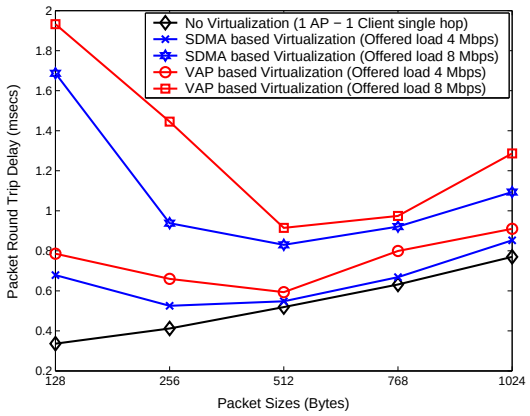


Figure 2.8: Round trip delay variations with packet size for VAP and SDMA based virtualization schemes as compared to the non-virtualized scenario.

Experiment setup for delay and jitter measurements is the same as that shown in Figures 2.4(a) and 2.4(b). Figure 2.8 shows the round trip delay measurements for the following cases: 1)No Virtualization, 2)SDMA and 3)VAP with different offered loads. We use two different offered loads to test the deterioration in delays with varying offered loads. With no virtualization, experiments show a linear increase in delay with packet sizes due to increase in transmission times. This deduction is based on the assumption that the individual experiments have a one hop wireless topology with single flows. Hence there are no CSMA contentions. However, in the case of virtualization, experimenters have a V-shaped curve for delay results. The nodes of every experiment face CSMA contentions with nodes from other experiments. Delay values decrease with packet size for smaller packets as the CSMA contentions decreases with lesser number of packets. However for large packet sizes, the transmission and queueing times are more prominent than CSMA contentions and the delay increases with packet size. The per-packet delays for SDMA experiments are lower as a result of capture effect. Capture ensures that the MAC frames are received despite collision, which lowers the net MAC retries for getting a packet across and consequently the queueing delays.

Figure 2.9 shows the round trip jitter as a function of different packet sizes and offered load. The trend for jitter follows the same pattern as that for delay i.e., high for small packets, decreases for bigger sizes and slightly increases for the biggest packets sizes. However, unlike delay, the jitter decreases with packet size for no virtualization scenario. Since we measure RTT jitter, there is contention even with one hop, single flow topologies. Hence, as the packet

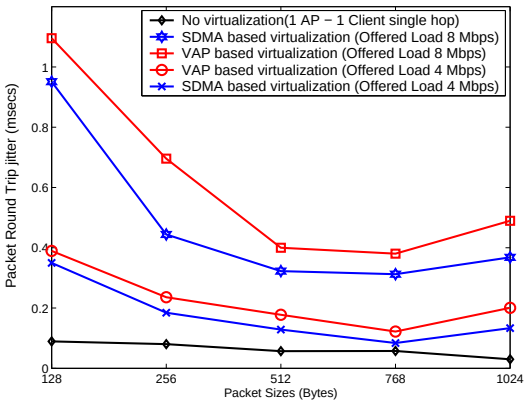


Figure 2.9: Round trip jitter variations with packet size for VAP and SDMA based virtualization schemes as compared to the non-virtualized scenario.

size increases, for a constant offered load the number of contending packets decrease resulting in decreased jitter.

2.6 Inter-experiment interference Illustrations

Repeatability of experiments is strongly related to the isolation in the experimentation environment. Often it is seen that abuse of resource by one device sharing a resource leads to a deterioration in performance for other experiments sharing the same platform. We will elaborate the consequences of these inter-experimental effects with time and space separation for virtualization and suggest approaches to mitigate them. In this section, we use the same experiment set-up as used in the throughput, delay and jitter characterization of VAP and SDMA-based virtualization schemes. The experiment setup is shown in Figures 2.4(a) and 2.4(b).

2.6.1 Metrics

For lack of accurate delay characterization tools, we consider inter-experimental effect primarily in terms of throughput. To quantify the inter-experiment effects we define a coupling factor between virtualized experiments as:

$$\sigma_{(nv\_num,\ v\_num)} = \frac{(T_{non-virtualized} - T_{virtualized})}{T_{non-virtualized}} \tag{2.1}$$

$\sigma_{(nv\_num,v\_num)}$ indicates the coupling between non-virtualized experiment nv_num and

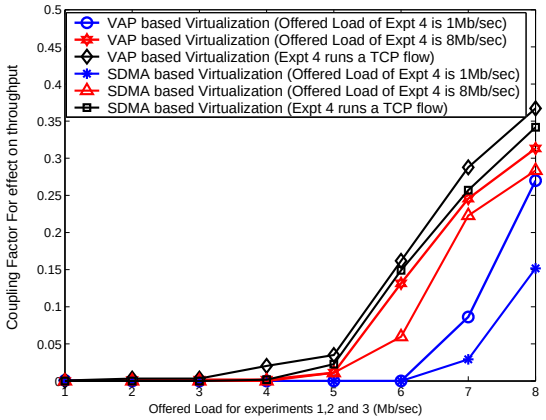


Figure 2.10: Coupling Factor for effect on throughput of experiments due to other experiments.

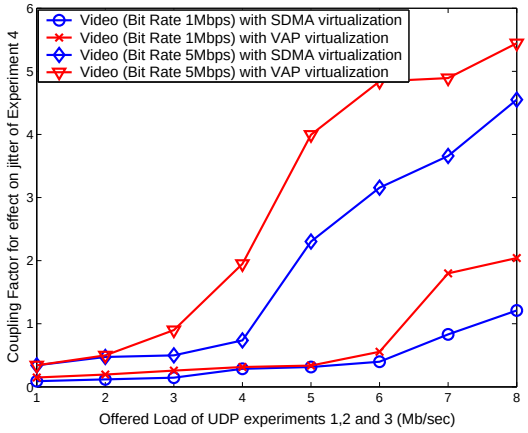


Figure 2.11: Effect on jitter measurements in channel multiplexing virtualization approaches.

virtualized experiment v_num. $T_{non-virtualized}$ and $T_{virtualized}$ represent the throughput of the experiments in the non-virtualized and virtualized cases respectively. A σ of 0 indicates an ideal experiment setup where there is no interference between experiments while a σ of 1 indicates complete interference of one experiment with the others. The Coupling Factor gives a direct indication of the level of interference expected between the virtualized experiments sharing a common wireless medium. Another approach is to use anti-correlation among the throughput of virtualized experiments.

2.6.2 Coupling Factors

Throughput Coupling Factor

In this subsection we study the transient behavior of the experiments using VAP and SDMA-based virtualization schemes. In the scenario with four concurrent experiments for both VAP and SDMA, we observe the impact of the fourth experiment on the first three experiments for different traffic scenarios of the fourth experiment. We plot the coupling factor for the first three experiments with varying offered loads for both VAP and SDMA-based approaches. The packet size used by all four experiments was set to 1024 bytes. The plot of the throughput coupling factor is shown in Figure 2.10:

- In the initial runs we keep the offered load of the fourth experiment at 1 Mbps and find the coupling factors for both virtualization schemes is negligible for low offered loads and start to become prominent after the offered loads for the three experiments crosses 6 Mbps. The channel is driven into saturation and effects the performance of the experiments. The effect is less for SDMA, since the performance of SDMA is superior to the VAP.

- In the case where the offered load of the fourth experiment is about 8 Mbps the channel saturates at lower values of offered loads of the first three experiments and therefore the coupling factor is higher.

- In the case where the fourth experiment uses TCP, the coupling on other experiments observed is relatively higher than that with UDP. TCP flow pumps traffic at the maximum possible rate and its effect is more significant on the other experiments than that observed with a UDP flow. This increase can also be accounted by the overhead of the TCP-ACK traffic that increases the amount of contention among the different experiment flows.

Jitter Coupling

Similar to throughput results, the experimental measurements of packet jitter is affected by traffic from other experiments. We investigate jitter coupling in VAP and SDMA-based virtualization approaches by streaming a video from a client to an AP as a part of one experiment and

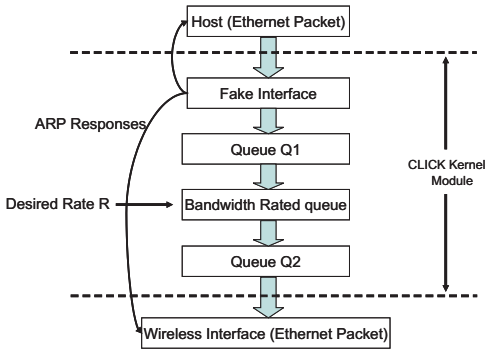


Figure 2.12: Click Modular Router Elements for Bandwidth Shaping.

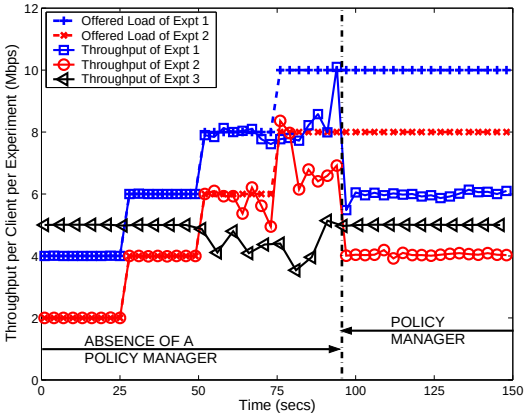


Figure 2.13: Application of a policy manager in enforcing channel throughput.

running UDP flows as part of the other three experiment. Figure 2.11 shows the plot for jitter coupling factor values for videos of different bit-rates for VAP and SDMA-based virtualization scenarios. The jitter values are calculated for a real-time experiment that streams videos of different bit-rates from a client to an AP. With no virtualization, it was observed that the jitter of the video does not depend upon its bit-rate. However, in the virtualized case as the bit-rate increases the jitter value increases. Moreover, the jitter values of the video increase as the channel approaches saturation due to increase in the offered load of the other 3 UDP experiments. Similar to throughput results, the jitter coupling is more for the VAP setting as compared to that with the SDMA virtualization.

2.6.3 Summary

A comparative evaluation of the coupling factors for various offered loads and packet sizes have been shown. The coupling factor could be thoroughly evaluated by calculating it as a matrix. Where each experiment coupling with all other is measured with varying experimentation parameters such as packet sizes, offered loads and channel rates. On an average it is seen that the setup with SDMA behaves better than that with VAP in terms of relative coupling. However, the absolute values of coupling in both the cases are significantly high, thereby making the setup unsuitable for scientific experimentation. Thus to assuage inter-experiment effects we propose and implement a policy manager.

2.7 Traffic Shaping/Policy Management for Virtualization

Results in Section 2.6 show that though the coupling factor is low for SDMA as compared to a VAP based approach, it is a non-zero entity and needs to be limited. Enforcing resource management across multiple experiments requires a systematic control framework for bandwidth assurance across multiple experiments.

2.7.1 Policy Manager

We will describe the implementation and testing of our policy manager with a VAP based setup. However, the same mechanism can be replicated and used without change for SDMA. A policy

manager primarily performs the following functions:

- Admission control: To make a decision to allow or deny an experiment to share a VAP (slice for SDMA) with other experiments depending on its bandwidth requirements.
- Assigning And Enforcing Bandwidth: To allot the different experiments a maximum bandwidth value based on the number of experiments on a single VAP (slice) and their bandwidth requirements.

The Policy manager could be integrated with the experiment scheduling and resource tracking mechanisms to ensure that each of the experiments get a fair share of the resources. The experiments would be rate limited to their maximum assigned bandwidth, even before the experiment execution is started. Our implementation is based on a kernel module created with the CLICK modular router. The configuration setup for the CLICK [36] module is as shown in Figure 2.12.

Figure 2.13 shows the throughput results. For demonstration purposes, we allow each of the experiments to have unbounded bandwidth for the first 75 secs. We see that increased offered loads for experiments 1 and 2 results in performance degradation for experiment 3. Enforcement of the policy manager results in limiting experiment 1 and 2 to $6Mbits/sec$ and $4Mbits/sec$ respectively. Thus by artificially reducing the bandwidth available to each of the experiments we reduce the inter-experiment coupling factors to 0.

2.8 Scheme Selection

Previous sections show the relative efficiency and inter-experiment coupling with the use of Space and time separation. Apart from these quantitative aspects other considerations for selection of a scheme are:

- Topology: VAPs are limited to infrastructure mode setups, while SDMA can work with ad hoc as well.
- Space Separation: Achieving isolation and efficiency with SDMA requires considerable spatial separation between slices (of the order of $10dB$) or artificial stretching [31] of the testbed by use of noise.

- Scalability: Number of experiment slices with SDMA is limited due to the number of nodes, space constraints of the testbed and or the granularity of the noise generation mechanism.

Thus the quantitative approaches along with a qualitative comparison would possibly yield the best virtualization for a given testbed.

2.9 Conclusions And Discussion

Our study shows two approaches to channel conservation for a wireless testbed. Evaluation of the space and time separation scheme reveal benefits and weaknesses for both. Space separation provides relatively higher efficiency, lesser coupling between experiments. We layout selection criterion for each of these schemes based on the requirement of the testbed and finally propose and implement a policy manager for controlling inter-exeperiment interference. Finally, incorporating arbitrary topologies in a slice or across VAPs allocated to the experiment may be challenging or impossible for some experiments.

Chapter 3

SplitAP Architecture For Supporting Virtualized WLANs

3.1 Chapter Summary

Providing air-time guarantees across a group of clients forms a fundamental building block in sharing an access point (AP) across different virtual network providers. Though this problem has a relatively simple solution for downlink group scheduling through traffic engineering at the AP, solving this problem for uplink (UL) traffic presents a challenge for fair sharing of wireless hotspots. Among other issues, the mechanism for uplink traffic control has to scale across a large user base, and provide flexible operation irrespective of the client channel conditions and network loads. In this study, we propose the SplitAP architecture that address the problem of sharing uplink airtime across groups of users by extending the idea of network virtualization. Our architecture allows us to deploy different algorithms for enforcing UL airtime fairness across client groups. In this study, we will highlight the design features of the SplitAP architecture, and present results from evaluation on a prototype deployed with: (1) *LPFC* and (2) *LPFC+*, two algorithms for controlling UL group fairness. Performance comparisons on the ORBIT testbed show that the proposed algorithms are capable of providing group air-time fairness across wireless clients irrespective of the network volume, and traffic type. The algorithms show up to 40% improvement with a modified Jain fairness index.

3.2 Introduction

The advent of pervasive wireless systems in the form of inexpensive handheld devices is expected to lead to an ever increasing deployment of wireless hotspots [37]. With more and more ISPs aiming to provide services at locations such as airports, cafes and common shopping areas, differentiation in the quality of service provided on shared hardware for wireless ISPs provides

a substantial challenge. A mechanism is required to ensure that this access point (AP) sharing will work across a wide range of client hardware, while providing each user group (clients belonging to a single ISP) with aggregate air-time commensurate to the revenue contract of the ISP with the wireless infrastructure provider. Apart from providing baseline fairness in air-time across groups, other requirements for sharing WLAN access point hardware across different ISPs include: (1) Different broadcast domains, (2) Different levels of security, (3) Support different protocols above a basic L2 connection, (4) Ease of deployment, and (5) Minimum bandwidth loss for resource partitioning.

To solve this problem we propose the SplitAP architecture that employs wireless network virtualization. Network virtualization is a concept derived from the server systems area of research which has recently been applied to network sharing. Virtualization is a mechanism that allows for seamless sharing of a particular resource by using three key features: *Abstraction, Programmability and Isolation*. We apply each of these features as shown in the Figure 3.1. *Abstraction* allows the users of the system to use our architecture with minimal changes to the client hardware or software. As shown in the Figure, we use virtual access points [33] supported by most commodity AP hardware to emulate the functionality of two different physical APs (ISP1, ISP2) with a single physical AP, thus allowing us to use the client MAC protocols and hardware unchanged. We provide *programmability* in the setup by allowing the person deploying the hardware to allocate different UL air-time quotas for individual virtual access points. Finally, *isolation* across groups of wireless users is provided through air-time control at the clients based on the information provided by the SplitAP controller running at the AP. Since downlink air-time fairness has been studied previously [27], and a spate of recent applications such as those supported by web 2.0 [38], peer-to-peer file sharing [39], and video conferencing have resulted in significantly increased uplink air-time usage, we specifically address the problem of uplink air-time control across the virtual networks formed by wireless user groups. Through the use of a SplitAP prototype discussed in the paper, we will show the performance of our sample algorithms for providing uplink air-time fairness across user groups, while providing all of the features discussed above.

Specifically the contributions of this study are:

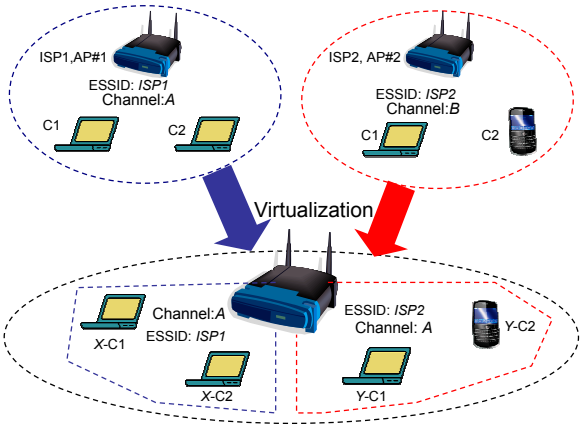


Figure 3.1: A single wireless access point emulating multiple virtual access points. Clients from different networks associate with corresponding VAPs though they use the same underlying hardware.

1. We propose, design and implement the SplitAP software architecture based on the extension of the virtual access point functionality for sharing a single physical AP across groups of users.

2. We design and evaluate the *LPFC* and *LPFC+* algorithms for group UL air-time control using our SplitAP setup on commercial *off-the-shelf* hardware.

3. Extensive evaluation is performed to show that the results obtained on our infrastructure are as per the requirement while achieving the system performance with minimal overhead.

Rest of the chapter is organized as follows. In Section 8.6 we discuss related work on previous approaches to providing uplink air-time fairness in WLANs. Section 3.4 discusses the problem of providing uplink air-time fairness across user groups, and presents the design of our SplitAP architecture. Section 3.5 presents a discussion on the two sample algorithms evaluated with the SplitAP framework. Section 8.5 presents the results from the system, and finally, Section 8.7 discusses the conclusions and future work.

3.3 Related Work

Among AP based infrastructures, the *DenseAP* architecture proposed in [40], describes a mechanism for sharing airtime by managing handoffs across APs. Another setup to share downlink air-time has been discussed for WiMAX radios in [27]. Our *SplitAP* setup specifically deals with the problem of providing an architecture for sharing UL air-time of a single AP across multiple WLAN user groups. In terms of the methodology itself, a comparison of wireless virtualization approaches is presented in [24]. However, it does not address the problem of fair sharing of UL air-time across client groups.

In the domain of air-time fairness, a body of work [41–44] discusses the use of EDCA parameters such as contention windows and transmission opportunities for controlling airtime usage across clients. The study in [41] attempts to ensure fairness across competing uplink stations with TCP traffic using EDCA parameters. Time fair CSMA protocol proposed in [42] controls minimum contention window size to achieve estimated target uplink throughput for each competing station in multirate WLAN. In [44] authors suggest that in a proportional fair allocation based on $802.11e$ EDCA parameters, equal share of channel time is given to high and low bit rate stations and, as a result, high bit rate stations can obtain more throughput. Another study in [43] proposes two control mechanisms for airtime fairness, one using AIFS and the other using contention window size. The studies in [41–44] are based on simulations.

One study in [45] proposes a Time Based Regulator system that achieves uplink air-time fairness by ensuring equal ”long term” channel occupancy time for every node in the WLAN. Though this study presents results based on an implementation, it does not deal with the problems of clients sending traffic with different frame sizes, offered loads, and sharing of airtime across user groups. The TWHTB system discussed in [46] uses information on current channel quality to the respective station associated with AP to schedule downlink transmission to that particular station by limiting frame transmission rate. However, this scheme does not take into account Uplink flows and corresponding traffic variations. Another study discussed in [47] discusses an approach where each station monitors the number of active stations and calculates the target access time based on this information. The study uses sniffing on the client side, while also requiring modification of NAV field in the MAC header, and results are based on

simulations.

In addition none of these studies address the problem of enforcing *client-group* UL airtime fairness which is addressed by algorithms run on our SplitAP setup.

3.4 SplitAP Design Overview

Throughout this study we use the notion of *slices* to refer to the resources allocated to a group of users belonging to a single ISP. The terms *groups* or *slices* will be used interchangeably in further discussion. Our infrastructure will try to enforce fairness in uplink (UL) airtime usage across slices, thus allowing individual ISPs to fairly share the underlying WLAN hardware and the corresponding channel. We begin with a formal definition of the problem of sharing UL airtime across a group of users, followed by a conceptual description of our virtualization based design. Eventually, we will discuss the details of the algorithms used for UL airtime allocation.

3.4.1 Group Uplink Airtime Fairness: Problem Statement

Consider a set of *M* client groups (slices) with each group S_i having N_i clients. Let the fraction of UL air time allocated for every slice $S_i \in M$, be denoted by W_i. W_i for each slice is decided during the time of deployment of the infrastructure and can be dependent on a wide range of criterion like pricing, importance of the group and so on. If ϕ_j^i denotes the measured UL air time consumed by the client $j \in S_i$ slice, the fraction of UL air-time used by every client associated with the access point is calculated as C_j^i:

$$C_j^i = \frac{\phi_j^i}{\sum_{p=1}^{M} \sum_{q=1}^{N_i} \phi_q^p} \tag{3.1}$$

The condition of group fairness requires that, the total measured UL airtime for all clients within a slice S_i is limited to W_i:

$$\forall\{i \in M\}, \quad \sum_{j=1}^{N_i} C_j^i \leq W_i \tag{3.2}$$

The above condition should be fulfilled while placing no limitation on the individual values of C_j^i i.e all nodes within a single slice S_i should be able to share the UL airtime fairly, independent of the usage on other slices. Hence, in the worst case every client should be able to utilize UL airtime $0 \leq C_j^i \leq W_i$ as long as the Equation (3.2) is not violated and all clients within S_i share the available UL airtime fairly.

Qualitatively summarizing the constraints of the slice/group fairness mechanism: (1) Flexiblity: If the channel usage is below saturation, and there are no hard guarantees, each client should be able to access the entire available channel time for the slice, (2) Within a group: Sharing of UL airtime should be fair and equal, (3) Scalable: Should work with a large number of clients without significant control overheads, (4) Adaptable: Should be able to comfortably adapt to changing environment with dynamic addition or removal of wireless clients, the network load, protocol type and the channel conditions for individual clients. Hence, to allow deployment of algorithms that will be able to realize such a group airtime fairness mechanism, our SplitAP infrastructure will need to provide all needed control and measurement features while being transparent to the users of the system.

3.4.2 Virtualization Based Design

We will now discuss how each of the virtualization features are implemented as a part of our SplitAP architecture.

Abstraction: We employ and extend the functionality of virtual access points which are available as a standard feature on commercial access points for emulating multiple virtual access points on a single physical access point while operating on the same wireless channel [33]. Using this feature the physical AP will be able to broadcast beacons for independent virtual networks (ISPs). Hence clients belonging to different ISP slices can see the ESSID of their ISP and associate with it, thereby making client side connectivity transparent and simple.

Programmability: Each of the ISPs should have independent control of settings in their network. Using virtual access points, we can set different features per WLANs such as different security policies, broadcast domains, IP settings, independent control of MAC settings such as aggregation and *802.11e* based WMM parameters.

Isolation: Isolation across virtual networks (client groups) is a fundamental requirement for

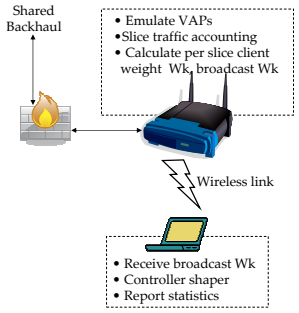


Figure 3.2: A single wireless access point emulating multiple virtual access points. Clients from different networks associate with corresponding VAPs though they use the same underlying hardware.

supporting multiple networks and will be the main topic discussed in this chapter. Ideally, this could be done through a strict TDMA scheduler across the virtual networks. However, such a scheduler would require a large change in the MAC mechanism of the clients, thus making them completely incompatible with other 802.11 based commercial access points. The SplitAP mechanism proposed in this chapter is an incremental design to the existing *802.11* framework and is currently capable of existing as a stand alone entity outside of the driver.

The functionality in our system is split as shown in the Figure 3.2. The SplitAP controller at the AP is responsible for emulating the virtual access points, accounting of traffic by client groups, and determining the weights of UL airtime for each group. The client software is responsible for enforcing the commands broadcasted by the controller and reporting usage statistics like the physical layer rate and the average packet size reported by the client interface. The remaining discussion will focus on the implementation of individual components, followed by a brief overview of our algorithms for providing uplink airtime fairness across the ISP slices.

3.4.3 SplitAP Controller

The access point infrastructure runs a multi-threaded ruby controller that performs the actions described in Algorithm (1). In the controller, $sliceID$ is a unique identifier used for identifying independent slices owned by different ISPs. The algorithm computes slice UL airtime usage $time[sliceID]$ for every $sliceID$, by iterating and determining the UL airtime usage reported by individual clients within every slice $sliceID$. Based on this estimate, it determines the

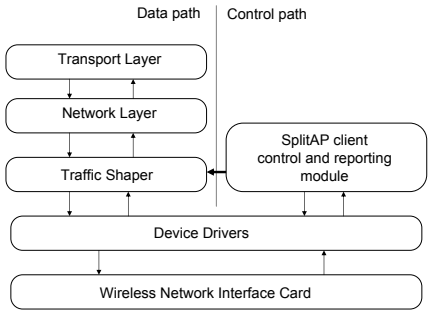


Figure 3.3: Network stack at the wireless client associating with the SplitAP infrastructure.

offset of the actual slice utilization from the allocated UL airtime fraction. If this offset is greater than a threshold (Θ), the AP controller broadcasts[1] $C_{sliceID}$ the maximum UL airtime fraction that can be consumed by any individual client within the slice $sliceID$. The value of $C_{sliceID}$ is always chosen as inversely proportional to the UL airtime utilization for that slice. This fraction of channel time is calculated based on the previously broadcasted value and the corresponding slice utilization. *LPFC* and *LPFC+* algorithms discussed later are two means of calculating $C_{sliceID}$ based on current UL airtime utilization numbers and or the number of associated clients.

3.4.4 Client Plugin Design

In the current design, the client needs to install an application that allows the user to connect to a SplitAP based wireles service provider. Eventually, to make this application platform independent, it could be implemented as a web browser *plugin* that controls client's UL traffic based on commands from the controller. The client software stack in the current SplitAP architecture is as shown in Figure 3.3. The SplitAP client control and reporting module is responsible for two functionalities: (1) Determining and reporting client side parameters such as physical layer rate (through access of the rate table maintained in the driver), and average packet sizes by querying the *proc* filesystem or using the driver statistics. (2) Converting the

[1]UDP broadcast is deliberately used as a means of sending $C_{sliceID}$ to clients to limit control traffic, since the number of control messages are now dependent on the number of slices rather than number of clients. Ideally, these $C_{sliceID}$ will be included in the beacons of individual virtual access points, thereby eliminating the need for a separate signalling mechanism.

Algorithm 1: The SplitAP controller running at the AP that monitors slice usage and correspondingly broadcasts slice weights to clients.

```
W_{0..M} = init_slice_weights()
while True do
    foreach sliceID = getNextSlice() do
        time[sliceID] = 0
        foreach clientID = getNextClient(sliceID) do
            rate = getPhyRate(clientID)
            bytes = getDataVol(clientID)
            cl_time = (bytes×8) / rate
            time[sliceID] + = cl_time
        δ = time[sliceID] - W_{sliceID}
        if (abs(δ) > Θ) then
            C_{sliceID} = getWt(sliceID, slice_wt, δ)
            broadcast(sliceID, C_{sliceID})
```

maximum airtime limit enforced by the SplitAP controller to a rate value, and accordingly controlling the shaping module to rate limit the client. The shaping module is implemented by using the Click [36] modular router that transparently controls outbound traffic from the interface.

3.5 Algorithms For Deployment With SplitAP

Our SplitAP design offers a convenient way to deploy different algorithms on the AP for controlling uplink airtime across slices. Each of the algorithms discussed in this section are ways to implement the *getwt()* function discussed in Algorithm (1) and provide the value $C_{sliceID}$, which is the maximum airtime that can be consumed by any client in Slice $S_{sliceID}$.

3.5.1 Algorithm(1): LPFC

This is a simple linear proportional feedback control (LPFC) based algorithm that uses a dynamic estimate of the number of clients associated with the AP to calculate the $C_{sliceID}$. Information on the number of clients associated with the AP is available in the SplitAP controller through querying of the *proc* interface on the AP. The algorithm calculates $C_{sliceID}$ simply by determining current number of clients in the slice $S_{sliceID}$ and proportionally splitting the

available (quota of) airtime $W_{sliceID}$ among the number of clients $N_{sliceID}$ within the slice. The SplitAP architecture allows this corrected $C_{sliceID}$ to be broadcasted every one second or at another interval desired by the ISP using the slice.

3.5.2 Algorithm(2): LPFC+

Instead of generating and broadcasting the $C_{sliceID}$ purely based on the slice UL airtime quota and number of clients in the slice, the LPFC+ algorithm relies on monitoring the current UL airtime utilization for the slice, which is available through the SplitAP client reports and appropriately controlling $C_{sliceID}$. The algorithm selects $C_{sliceID}$ in such a way that even if the offered load by clients in a slice is not the same, it allows the clients to increase traffic, by increasing $C_{sliceID}$ until the UL airtime quota for the slice is reached. If the quota is exceeded (or under-utilized), the LPFC+ controller proportionally reduces (or increases)$C_{sliceID}$, the maximum airtime that can be used by any client in the Slice $sliceID$. As with the LPFC algorithm, the $C_{sliceID}$ can be broadcasted every one second or at any other value desired by the ISP owning the slice.

3.6 Configuration Through Simulations

In this section, we will discuss the implication of selection of different design parameters on the performance of the algorithms designed in the previous section. Specifically, the idea of these simulations is to evaluate the baseline performance of the system, and gaining a better understanding of the selection of different parameters on the performance of the system.

3.6.1 Simulation Model

Our simulations are setup in MATLAB. The goal of these simulations is to show a broad impact of the control loop established with the *LPFC+* algorithm on the SplitAP framework.

Channel model: As a part of our simulations we do not model the physical layer characteristics like packets lost due to interference, noise or 802.11 MAC failures such as collisions. Rather, this model takes an orthogonal approach, where we study the achievable performance of the system by considering specific failure rates in the signalling mechanism in the SplitAP

Table 3.1: Parameters for the simulation model.

$Parameter$	$Value$
Number of clients (Slice 1,2)	{5, 20}
Desired airtime (Slice 1,2)	{0.5, 0.5}
Number of control loop runs	1000
Hysteresis time	0
Number Of Beacon Repititions	1
Probability Of Beacon Loss	0.4
Control conservativeness	None

framework.

Network and traffic model: In our simulations we consider two virtual access points created on a single physical access points. For the simulations we assume that both of these virtual access points in the SplitAP framework are allocated 50% of the entire radio resources on the access point. Uplink traffic is generated with the assumption that the first slice has 5 clients, while the second slice has 20 clients on an average. The amount of traffic generated by each of the clients within these slices is determined by a scaled poisson random variable (emulating number of flows per client).

In all further evaluations, the parameters used are as shown in the Table 3.1, unless mentioned otherwise. In all of the experiments with this model, we measure the sum of the fraction of airtime used by all clients within the slice. It is important to note that the performance seen with these simulations will be different from those seen with a physical system. In the simulations we allow for the sum of fraction of channel times used by clients across slices to exceed 1. Specifically, in a real system when a slice exceeds its quota it can lead to a reduction in the airtime available to the other slice. However, instead of modeling that effect, we allow the total airtime of the both slices to exceed their quota, which allows us to clearly determine when either a single or both slices are not able to successfully curtail the airtime used by their clients.

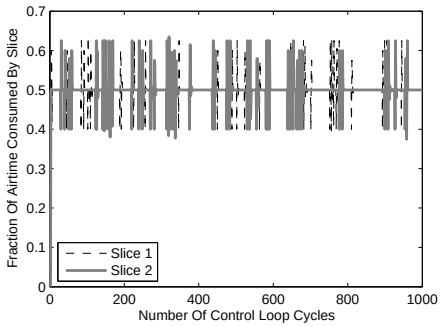


Figure 3.4: Performance of the *LPFC+* algorithm on the simulated SplitAP controller. The graph plots the sum of airtime used by all clients per slice.

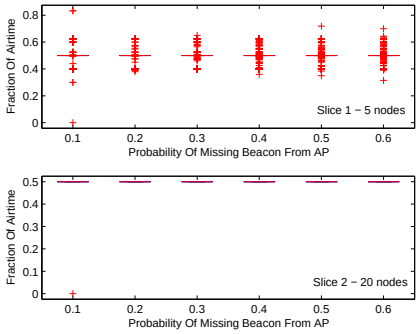


Figure 3.5: Boxplot showing the performance of the airtime usage per slice as a function of varying fraction of control beacon loss probability. We observe that as the probability of loosing beacons increases, more points for the slice usage measurement are obtained as outliers.

3.6.2 Baseline Performance

To study the baseline performance of the system, we present the results of the experiment as shown in the Figure 3.4. The x-axis of the plot is the number of control loop iterations with the SplitAP framework. The y-axis of the plot shows the fraction of airtime used by both the slices. We observe that both the slices are able to achieve 0.5 fraction of airtime on an average, which corresponds to the set value that needs to be achieved by the system. We further observe that there are a considerable number of oscillations in the airtime that is used by each of these slices. We can classify the reasons for these oscillations as:

- Changing offered loads: Due to continuous changes in the offered load of the system, the framework has a certain amount of response time before which corrective action can be taken.

- Hysteresis: The default settings does not have any hysteresis, because of which there could be oscillations in the system.

- Missing control beacons: In our simulations we have also modeled random drop of control beacons from the SplitAP due to noise or interference and broadcast (non-acknowledged) nature of the messages. This will result in clients not being able to update the shaping rate with changing demand and utilization.

We will now investigate the impact of each of these parameters on the performance of the system.

3.6.3 Varying Beacon Loss Probability

In this experiment we will determine the impact of loosing control beacons from the SplitAP framework on each of the wireless clients. This experiment is important from a practical perspective because we use a broadcast UDP client-server as a part of the framework, and hence there are no guarantees of control beacon delivery to clients. To measure performance with this phenomenon, we vary the probability with which a beacon is successfully received at every client.

Results from the experiments are as shown in the Figure 3.5. We will begin by inspecting the first subplot which indicates overall performance of the clients in the first slice. As seen in the boxplot for slice 1, we observe that as the probability of loosing beacons increases, more points for the slice usage measurement are obtained as outliers. Thus we see a direct correlation of the stability of the system with the control beacon loss probability. Though, this result is as per expectations, we observe through the second subplot (for slice 2) that even though the probability of loosing beacons increases, the system still performs very well. This performance is justified because, the traffic generated by the 20 clients in slice 2 is causing the SplitAP controller to allocate very little traffic shares to individual clients. Since each of the clients are generating more traffic (on an average) than that allowed by the controller, we see very small

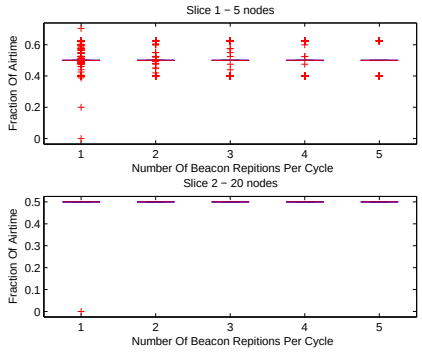


Figure 3.6: Performance with repeating beacons per control loop of the system. Increasing the number of beacon transmissions per control loop increases the probability of delivering the broadcast beacons to the clients.

deviations in the offered load from each client. This in turn leads to very small change in the overall airtime consumed by the slice.

Thus this experiment allows us to draw two inferences. (1) When the number of clients in a slice are significantly high (of the order of 10s of nodes), the impact of the beacon loss probability does not affect the overall performance of the system, and (2) When the number of clients in the slice are less, it is advisable to have some mechanism to counter the loss of control beacons. We will test two mechanisms to determine how performance can be improved in such a case using:

- Replication of beacon transmissions
- Conservativeness of the shaping

Each of these will be discussed in further experiments.

3.6.4 Replicate Control Beacons

In order to improve control beacon delivery on lossy links, we did not actually make the delivery of the broadcast message reliable, rather, we consider multiple duplicate broadcasts of

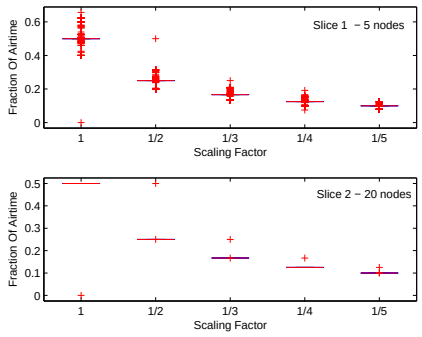


Figure 3.7: Performance with varying conservativeness of the traffic shaping scheme of the clients. As seen in the results, increasing the conservativeness reduces the chances of the slice overshooting its quota, but can also result in decreasing the overall performance of the system.

control beacons. Replicating broadcast messages instead of making unicast delivery guaranteed takes advantage of the inherent broadcast nature of the wireless medium, thus providing better spectrum efficiency. Results from this experiment are as shown in the Figure 3.6. The boxplot shows that increasing the number of beacon transmissions per control loop increases the probability of delivering the control beacons to the clients. We observe that for slice 1, as the number of beacons per control loop are increased, we have fewer outliers in terms of the number of times the slice airtime overshoots the allocated quota. Another interesting tradeoff in this decision to replicate broadcast packets is that the number of times a control broadcast is replicated should be inversely proportional to the rate at which the control updates in $C_{sliceID}$ are done at the SplitAP controller.

We also observe that for the lower subplot depicting slice 2, there is no significant difference in performance. This reasoning is the same as that described in the earlier experiment, and is basically due to very small variations in the broadcasted control weight ($C_{sliceID}$) when the slice is operating at saturation with a large number of clients. Thus the rate of duplication of the control beacons should also be chosen in inverse relation to the number of clients in the virtual network.

[3]If we define the beacon packet loss probability (discussed in the previous section), at every client as $P_b \,|P_b \leq 1$ by definition, and the number of duplications as m, then the probability of loosing beacon control packets at any client drops to P_b^m

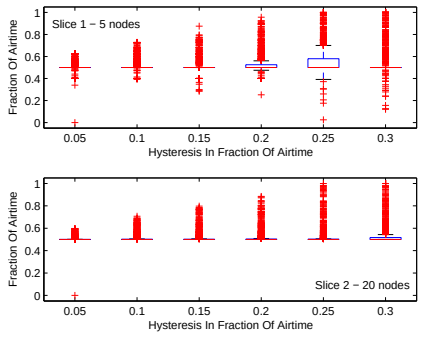


Figure 3.8: Performance impact of selecting different hysteresis parameters with the system. Results show that increasing the hysteresis may result in possibly less oscillations with the control loop, but it also causes a more sluggish reaction with changing load, resulting in worse performance.

3.6.5 Vary Shaping Conservativeness

Varying the conservativeness with which we shape the offered load per client can also provide a mechanism to counter against lost control beacons. To determine the impact we scale the rate at which we shape the traffic using the weights: $\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}\}$. Results from the experiments are as shown in the boxplot in Figure 3.7. As observed for slice 1, scaling with smaller multiplying factors results in comparatively lesser deviations from the median. We also observe that the overall utilization in terms of fraction of airtime consumed by the slice also deceases as we increase the conservativeness of the shaping scheme. However, it is important to note that increasing the conservativeness of the shaping algorithm beyond a certain point does not significantly improve performance because, at that point performance is limited by the offered load on the slice rather than shaping rate. As expected, in the second subplot, with higher number of clients in slice 2, the system works well even without increasing the conservativeness of the shaping algorithm. Thus, we can conclude from these experiments that it would make sense to increase the conservativeness of the shaping algorithm only when we have a small number of clients.

3.6.6 Impact Of Hysteresis δ

Finally, we will try to assess the impact of selecting a hysteresis parameter on the stability of the system. Hysteresis is implemented at the SplitAP controller by updating the $C_{sliceID}$ parameter only if the difference in the current airtime used by the slice exceeds the allocated quota by a certain margin (δ). Results from this experiment are as shown in the boxplot in Figure 3.8. The x-axis shows the amount of hysteresis used by the controller in terms of fraction of airtime. As seen in the results for slice 1, we observe that increasing the hysteresis may result in possibly less oscillations with the control loop, but it also causes a more sluggish reaction with changing load, resulting in worse performance. We observe a similar performance for slice 2. However, we observe that there are no outliers on the lower side of the slice quota of 0.5. The reason we see this is because even though the airtime consumed by individual slices can go above the quota due to the hysteresis. However, the load on the slice with large number of clients does not decreases significantly below the median resulting in the performance trend shown in the plot.

3.6.7 Summary Of System Parameters Selection

Experiments discussed above allow us to have a better insight into the working of the system, and help select better suited control parameters. Based on the discussion above we recommend following guidelines for selecting various system parameters with different deployments of the system:

1. When the number of clients in the slice are large, and the airtime quota is fairly used, we observe that the system will stay stable irrespective of the loss of control beacons, and as long as there are no large fluctuations in the number of active clients in the slice.

2. When the number of clients in the slice are less or vary a lot, we can use mechanisms like multiple control beacons per control loop, or change the conservativeness of the shaping algorithm to improve performance.

3. Number of replications of broadcast beacons should be inversely proportional to the rate at which the control updates in $C_{sliceID}$ are done at the SplitAP controller, and directly

proportional to the beacon loss probability.

4. Conservativeness of the shaping algorithm can improve performance at the cost of efficiency of the system and should not be used when airtime utilization is a concern.

5. Finally, adding significant hysteresis in the system will only add to the sluggishness of the controller and is not very useful when the offered traffic keeps changing rapidly. However, a small amount of hysteresis may be useful for preventing the system from oscillations.

Remainder of the paper will show the performance of the SplitAP prototype on the ORBIT testbed.

3.7 Experimental Evaluation

All experimental results presented in this evaluation are based on the clients with Atheros 5212 chipsets, and using Madwifi 0.9.4 [34] drivers. The clients are all operating in the 802.11a mode with a frame size of 1024bytes, and 54Mbps physical layer rate unless mentioned otherwise. Traffic is generated with the Iperf tool [48]. We begin with a brief definition of the metrics used, followed by baseline performance of the *LPFC* algorithm and a comparison with *LPFC+*.

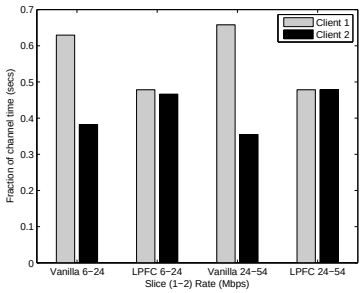
3.7.1 Metrics

Preliminary evaluations with a small number of clients will be based purely on comparison of UL airtime allocated to individual slice. Further, in our evaluations, we modify and use the Jain fairness index [49] for determining weighted UL airtime fairness across flows and flow groups.

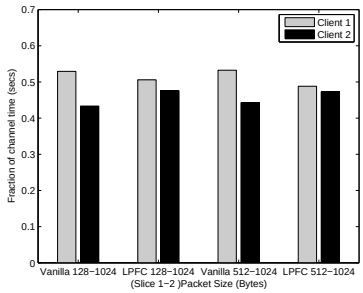
Modified Jain Index: Let the sum of fraction of channel time used by all clients in slice k be denoted as C_k.Then,

$$I = \frac{\left(\sum_{k=1}^{N} C_k\right)^2}{N \times \sum_{k=1}^{N} C_k^2} \tag{3.3}$$

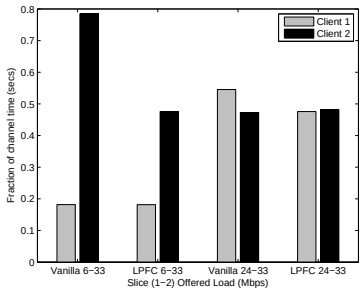
The fairness index (I) determines the global variation in channel utilization across slices. We further scale the airtime by slice quotas to evaluate fairness under saturation with different slice weights, while also accounting for performance deterioration due to bad channel quality.



(a) Varying rate



(b) Varying packet size



(c) Varying load

Figure 3.9: Baseline results for comparison of performance with and without the the SplitAP setup with the *LPFC* algorithm. Results indicate performance with two clients on different virtual networks with varying physical layer transmission rates used with a UDP saturated offered load.

3.7.2 Baseline Performance With LPFC

To measure the baseline performance with the *LPFC* algorithm, we consider a setup with two clients on different slices sending UDP UL traffic.

Varying Transmission Rates In the first experiment, we vary transmission rates of the two clients on Slice 1 and 2 as shown on the x-axis in Figure 3.9(a). We observe that, in the vanilla case (without the SplitAP mechanism running the *LPFC* algorithm), the air-time used by the two clients are inversely dependent on the transmission rates. This is a result of statistical multiplexing of packets by the CSMA MAC operating as a part of the 802.11 DCF mechanism. To alleviate this problem, the SplitAP framework controls the shaping of traffic at the source to consequently provide proportional fair channel usage across clients. As shown in the results in Figure 3.9(a), we are able to control air-time provided to each client in desired proportion. Sample results are provided for a $50 - 50$ percent air-time sharing across the two clients.

Varying Packet Sizes Now we will vary the packet size of the uplink traffic from each client

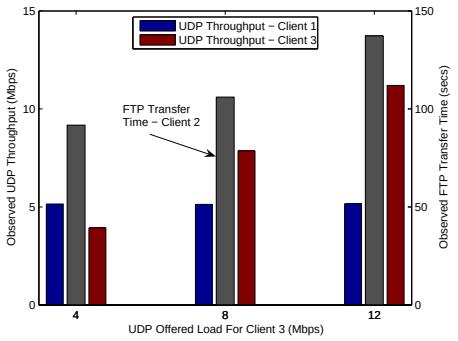


Figure 3.10: TCP and UDP co-existence in a single slice with *LPFC+*. Constant UDP traffic of 5Mbps is supported by slice 1, while the Client 2 with FTP transfer and the client 3 with varying UDP loads share the slice 2.

to check its impact on the overall sharing of air time at the access point for the two clients. As seen in the results in Figure 3.9(b), the air-time consumed at the access point without the use of our scheme (vanilla) is directly dependent on the size of the packets used by the uplink traffic. Typically, this results from a statistical multiplexing of packets over the air. However, using our SplitAP infrastructure with the *LPFC* algorithm we are able to control uplink traffic in direct proportion to the air time usage by each client. Our scheme accounts for the extra air-time spent in channel accesses and PHY/MAC overheads with smaller packet sizes resulting in fair sharing across the clients and thus virtual networks. As before we observe, that our infrastructure allows control of air-time across the clients in a preset $50 - 50$ percentage. In later experiments this percentage will be changed.

Varying Offered Loads In this experiment, we vary the offered loads across the two clients. Combinations of offered loads used across the clients are as shown on the x-axis in the results in Figure 3.9(c). The maximum offered load is limited to 33Mbps because the channel saturates at that value[2] of the offered load when the physical layer rate is 54Mbps. We observe that the *LPFC* algorithm limits airtime of slice 1 (with 33Mbps physical rate) even though the other client is not using its share. Even though the *LPFC* scheme is conservative, it limits airtime of Slice 2, to ensure better fairness as compared to the vanilla case with no control.

[2]The channel saturates at a slightly higher value than normal since the Madwifi drivers use fast framing optimizations to improve performance within allocated txops. However, this does not affect our evaluation since it is enabled in all measurement cases.

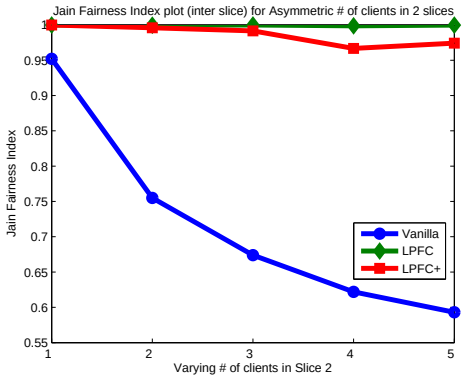


Figure 3.11: Comparison of UL airtime group fairness for: LPFC, LPFC+, and a vanilla system without our SplitAP framework.

3.7.3 Improvement With LPFC+

Since the LPFC+ algorithm allows the allocation of slice weights such that within a slice we may have varying utilization by independent clients, such a mechanism allows for fair co-existence of transport protocols with different requirements. In this experiment we have two slices: Slice 1 has a client sending constant UDP uplink traffic, while the Slice 2 has two clients. The first client in Slice 2 is sending varying amount of UDP uplink traffic, while the other client in Slice 2 is transfering a 200MB file with a FTP file transfer. Results from this experiment are as shown in Figure 3.10. We observe that the client on slice 1 is not affected despite one of the clients on Slice 2 using UDP traffic. We also observe, that the clients on Slice 2 share the UL airtime. When the UDP offered load is less at 4Mbps, the FTP transfer is faster and happens at an aggregate rate of 18.3Mbps. When the UDP offered load on the client increases, the FTP client reduces its rate, thereby requiring longer time for the FTP transfer completion. It is important to note that a similar performance could be achieved even by using *LPFC* instead of *LPFC+*. However, in that case the FTP client on slice 2 would always be limited to a fixed uplink rate thereby resulting in a wastage of free bandwidth.

3.7.4 Comparison: LPFC Vs LPFC+

In a final experiment we consider a setup with two slices: Slice 1 has a single client pumping UDP UL traffic at saturation, while Slice 2 has 5 clients associated with it. For different

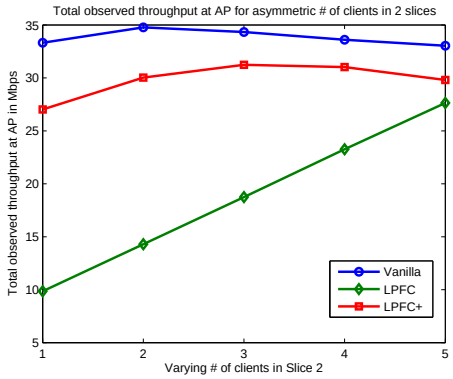


Figure 3.12: Comparison of UL throughput for: LPFC, LPFC+, and a vanilla system without our SplitAP framework.

experiments, varying number of clients $1-5$ on Slice 2 will send saturation UL traffic along with the client on Slice 1. In this case we consider the performance of both *LPFC* and *LPFC+* algorithms, as compared to that without our SplitAP setup (Vanilla). A comparison of the measured modified fairness index is as shown in Figure 3.11. We observe that the group fairness index I is always greater than 0.97 with the use of our infrastructure, while it falls down up to 0.6 in a vanilla system without our setup. The throughput measurements in Figure 3.12 show that the improvements in fairness are at the cost of a small decrease in net throughput with *LPFC+*, thus justifying the use of our scheme. The throughput performance with our *LPFC* scheme is less when lesser number of clients on Slice 2 pump traffic. This is because it sees 5 clients associated with the slice from the beginning, and presents a conservative estimate of $C_{sliceID}$ which results in lower throughput. The *LPFC+* scheme on the other hand dynamically measures airtime for every slice and adapts its $C_{sliceID}$ resulting in better performance. It cannot reach channel capacity since it keeps a 15% tolerance, but is able to divide the remaining airtime fairly.

3.8 Conclusions And Future Directions

This study discusses the design of the SplitAP architecture that allows the operator to deploy a shared physical access point, which is capable of running algorithms that control UL airtime

across user groups. We demonstrate the feasibility of the proposed architecture by implementing the *LPFC* and *LPFC+* algorithms on a prototype. Results obtained from the measurements on the ORBIT testbed show a significant improvement in the group airtime fairness, while resulting in marginal degradation of overall system throughput. Future directions include search for more efficient algorithms that can be deployed on the SplitAP framework.

Chapter 4

Virtual Basestation Design

4.1 Chapter Summary

This chapter 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.

4.2 Introduction

Capital and operating expenditures (CapEx and OpEx) are the dominant costs in operating a cellular basestation. These costs typically scale with the number of independent basestations deployed rather than the capacity allocated at each cell site deployment. If the CapEx and OpEx are normalized with the capacity allocated and used across all deployments, we see that the aggregate expenditures will be lower if the network of basestation deployments is well provisioned. Approaches at different layers of the protocol stack have been proposed earlier for maximizing capacity allocation and utilization of basestation deployments. However, all of

these approaches address the problem within a single administrative domain, i.e. basestations belonging to a single entity like an internet service provider (ISP), or a cellular service provider. The reason for doing this was partly because, previously there was no clean approach to share the underlying basestations across multiple independent entities. In this study we present the virtual basestation design that allows the network provider to share a single basestation across multiple entities. Before we delve into details of the design itself, we motivate our work with two concrete examples where sharing basestations across entities would prove beneficial.

4.2.1 Motivational Examples

(1) Mobile Virtual Network Operators (MVNOs): With the advent of a new generation of wireless technologies like $4G$, the number of subscribers and the corresponding set of customers are on the rise in developing countries [13–15]. MVNOs are wireless network providers that lease the physical network from the mobile network operators (MNO) to provide wireless services catering to niche needs among customers. The current approach to supporting MVNOs requires the mobile network operator to lease basestations, spectrum and other related infrastructure to the MVNOs. The proposed virtual basestation design will allow the network provider to house multiple MVNOs on a single basestation, and share its spectrum resources as opposed to provisioning independent basestations.

(2) Experimental Testbeds: Sharing a basestation will also prove useful on large scale heterogeneous testbeds such as GENI [16]. The GENI testbed envisions a nationwide shared infrastrucuture with mobile wireless edges for supporting simultaneous experimentation. To support shared end to end experiments on such a framework, the wireless edge will need to be shared among multiple experimental slices. Using our proposed virtual basestation design, experimenters will be able to control independent virtual basestations and perform their experiments in a repeatable fashion.

4.2.2 Contribution

The applications described above clearly show that a mechanism is needed by which cellular basestations can be shared among multiple entities (MVNOs or experimenters). Our proposed virtual basestation design, which addresses this concern is based on the systems concepts of

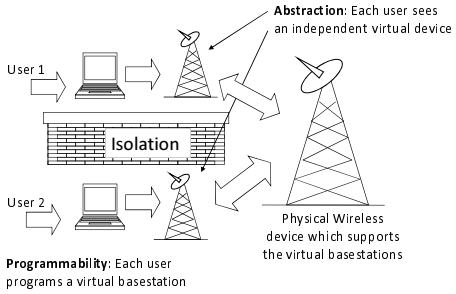


Figure 4.1: Application of the abstraction, programmability and isolation principles within the virtual basestation design.

virtualization. Virtualization is achieved by applying its three fundamental principles: (1) *abstraction* - for emulating multiple basestations on a single physical basestaion, (2) *programmability* - for control, and (3) *isolation* - for ensuring performance guarantees. As seen in Figure 4.1, each of the system users will now be able to use an independent virtual basestation for running MVNOs (or testbed slices depending on the applications), while behaving as if operating on a physical basestation albeit with a less powerful radio. For example, if the airtime on the physical basestation is shared equally by two virtual basestations, then both the slices will see a radio which is half as powerful as the physical radio. However, the advantage of such an approach is that in the MVNO application context, these deployments on the virtual basestations can be easily ported to a physical basestation, and that each slice will always receive its share of radio resources. In the testbed application context, experimenters will be able to program virtual basestations as if they were programming real physical basestations while designing repeatable experiments, made possible because of the slice isolation. Thus even though the results obtained in the experiments might not be the same as that would be obtained on a physical basestation, they can still be easily used for performance evaluation by appropriately scaling them with the allocated quota of the physical radio.

4.3 Related Work

Network Virtualization: A previous study on virtualizing commodity wireless devices [11] proposes enhancement to the ORBIT radio grid, through the use of a time division multiplexing scheme for scheduling wireless networks. This study provides a motivation for challenges

in scheduling and context swapping virtual networks. Our approach is orthogonal, in that we share the inherent time sharing nature of the MAC, and build mechanisms on top to ensure isolation across slices. The Multinet [12] project discusses an architecture for supporting a virtualized client connection to multiple networks from a single wireless client. Our virtual basestation setup discusses a solution for virtualizing a $802.16e$ basestation. A survey of general virtualization techniques is discussed here [50]. However, it does not address the application of virtualization to wireless networks. Apart from these, the GENI [16] working group has also proposed some approaches for virtualization of the wireless medium which rely on MAC multiplexing, but they do not discuss specifics.

Wireless 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) [33]. VAPs were proposed as abstractions on a physical access points (APs), such that the functionality provided by the VAP is similar to that of an AP. VANU supports a MultiRAN virtual basestation design [51]. This design aims at running the entire virtualized BTS radio in software while our 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 [52] which relies on implementing a 2G BTS in software, though it does not support virtualization. A study on isolating groups of flows as a means to virtualization is discussed in [53]. This study relies on solving a utility maximization problem for allocating resources within the MAC scheduler. This approach requires us to have access to the MAC frame scheduler for resource allocation, which is not available for most carrier grade basestations. Another study [54] discusses approaches in which the network architecture may be emulated using virtual machines, but does not deal with the emulation of the radio itself. One more feature which differentiates our work from previously described work such as [51], [52], and [53] is that they work only with a specific hardware product. The approach we propose is backward compatible and can be evaluated with existing basestations (which do not have native support for virtualization), that support a similar network architecture.

4.4 Design Methodology

We begin this discussion with a brief description of a conventional WiMAX network. This is followed by a discusion on how we apply the concepts of virtualization to the WiMAX framework in our virtual basestation design.

4.4.1 Conventional WiMAX Network

A standard Profile-A/C WiMAX system typically consists of two important components: (1) Basestation transceiver system (BTS)[1], (2) Application service network gateway (ASN-GW or ASN)[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 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.

4.4.2 Virtualized WiMAX Network

Virtualization can be defined as the methodology by which an underlying *resource* is shared across multiple consumers, while providing each of the consumers with the illusion of owning the entire resource independent of the other consumers. Though this concept originated as a part of conventional server systems and operating systems virtualization, it has been extended to networks under various usage cases [7, 8, 18, 55, 56]. In case of network virtualization, the *resource* under consideration could be the network interface, bandwidth, airtime, device drivers, and logical devices among others. In this paper, using our virtual basestation design we discuss how virtualization can be extended to wireless networks, specifically, wireless networks

[1] From here on, the terms BTS and basestation will be used interchangeably.

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

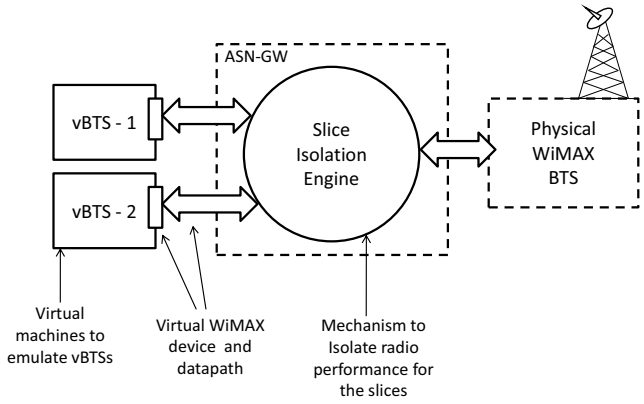


Figure 4.2: Basic building blocks of the virtual basestation design.

with a star topology.

Our virtual basetation relies on multiple components that are used to emulate basetations within individual slices. A brief overview of the virtual basestation design is as shown in the Figure 4.2, and the components are explained as below:

- vBTS Environment: We will need to select a virtual machine technology for housing the virtual basestations (vBTSs) or slices. The virtual basestation framework also supports a set of services for providing arbitrated access to the underlying physical basestation from within every slice. The environment and services are discussed in our previous work [57, 58] and will not be discussed here.

- Virtual WiMAX Network Device: As shown in Figure 4.2, within every slice our virtual basestation framework, provides a network device which acts as an interface on an actual basestation. Hence every frame directed to this interface transparently reaches the WiMAX clients through the physical basestation. In further sections we will discuss different aspects of this virtual interface such as the datapath connecting the virtual basetations to the physical basestation, and the delay performance with such a virtualized device.

- Resource Mapping and Isolation: We need to virtualize the WiMAX radio itself including important sub-components such as the WiMAX MAC scheduler, service flows, and

the time-frequency slots of the spectrum used by that scheduler for sharing across slices. Other important aspects for achieving true virtualization of the radio interface include emulation of MAC specific features such as multiple SSIDs to represent virtual BTSs for different slices.

Details on each of these design aspects will be discussed through the rest of the paper.

4.5 Virtual Basestation Environment

Our virtual basestation design emulates an independent physical basestation to each slice by using virtual machines. The first part of this discussion will focus on selection of the virtual machine (VM) technology for emulating the vBTSs, while the second part will provide a discussion on the grid services required for providing an actual BTS like interface within the virtual machines.

4.5.1 VM Technology Selection

Choosing an appropriate virtualization technology plays a strong role in the overall system performance. In our virtual basestation framework, the features and the set of experiments that can be supported within the slices are largely dependent on the type of virtualization technology used for running the virtual machines (VMs). We begin with a discussion on the choices available for the virtualization platform, followed by selection criterion for usage in the virtual basestation framework, and finally, our choice for the prototype.

Production scale virtualization systems can be broadly classified into full, para and OS virtualization. Full virtualization [59,60](e.g.,VMWare, KVM) refers to a technique that emulates the underlying hardware and uses a software layer called hypervisor that runs directly on top of the host hardware to trap and execute privileged instructions on the fly[4]. Full virtualization is the least intrusive[5] form of system virtualization. In para virtualization [61, 62](e.g.,

[3]Throughout the rest of the paper we will use the terms *slice*, *user*, and *vBTS* interchangeably.

[4]Native virtualization is a virtualization approach where the processor has support for virtualization e.g.,IBM System/370 and allows multiple unmodified operating systems to run together. Full virtualization does not include these systems.

[5]Intrusiveness refers to the degree of changes that need to be made to the guest OS to get it working with virtualization.

Xen, UML) the hypervisor layer exists within the host operating system to intercept and execute privileged instructions. Unlike full virtualization, para virtualization requires changes to the guest operating system. The most intrusive form of virtualization is operating system based [63](e.g.,OpenVZ) where the virtualized systems run as isolated processes in the host operating system. The host OS is modified to provide secure isolation of the guest OS. In case of operating system level virtualization, since the guest OS is not able to run a separate kernel, we prefer to chose from the previous two mechanisms for virtualization. A more comprehensive comparison on the different types of virtualization schemes and their suitability to the wireless testbed are discussed in previous studies [57, 58].

For the purpose of a virtualized WiMAX basestation, we consider the following qualitative criteria as the main selection parameters among full virtualization and para virtualization mechanisms:

1. Ease of administration: Clean API to schedule node resources such as CPU, disk and memory on a per slice basis should be possible.

2. Shared or exclusive interface mapping: The setup should allow flexible mapping of virtual interfaces within the slice to physical interfaces or one or more virtual interfaces (on the hardware like virtual access points).

3. Control over network connectivity: Mechanisms should be available to bandwidth limit slices and control interaction between slices.

4. Support with mainstream kernels: As long as the virtualization technology does not require huge changes to the kernel, there are high chances of it supporting the latest kernels, and their upgrades thus allowing the software to stay current.

All types of virtualization schemes discussed above support most of these functions. However, in our experience, the most flexible and easy approach for controlling the VMs is through using KVM [60], a full virtualization mechanism that works directly with debian based systems which is widely used in the rest of the ORBIT testbed. Such a setup also allows for the reuse and extension of regular system administration tools (such as IPTABLES, DHCP, SSH, LDAP) for controlling VMs.

4.5.2 OMF Based Grid Services

To emulate a physical basestation accurately to a user, we need to make the user interface of the virtual basestations as similar to the physical BTS as possible. To achieve this, we propose using OMF [64] based services for arbitrated control of the underlying basestation from within the slice. Software services are needed which will allow the user of the virtual basestations to control the physical basestation while being oblivious to other users of the systems. These software services, referred to as "grid services" in the ORBIT testbed [28] context are used to provide both, control of the virtual machine (slice) instances as well as control of the RF related features.

In our virtual basestation framework, we highlight four fundamental service functionalities that are essential for the working of the virtualized framework: 1) environment control - for providing API to the experimenters to initiate, start and stop their slices, (2) virtual radio control - for providing API to the experimenters for controlling the radio device through change of parameters. These parameters will be a subset of all the parameters that can be changed on the physical basestation (3) slice feedback - this service will provide feedback about performance of the wireless clients belonging to the slice, and finally, (4) a virtual radio isolation service - will be responsible for enforcing radio isolation across multiple slices.

To encompass all of the functionalities described above, we build three services for our virtual basestation framework. The WiMAX-VM service is responsible for implementing the slice control, and also automatically controlling the datapath connecting the vBTSs to the physical basestation. We built the WiMAX-RF service which provide access to different aspects of the WiMAX radio. Finally, the slice isolation engine is implemented as a separate module and discussed in more detail in Section 4.7B.

4.6 L2 Device Datapath

Since the impact of virtualizing the physical BTS by running virtual machines has been discussed previously [57, 58], we will focus on the design choices and the possible performance implications of the datapath mechanism required for forwarding frames from the vBTSs to the physical basestation.

4.6.1 L2 Datapath To BTS

The WiMAX network interface for every vBTS is emulated by virtual ethernet interfaces within the virtual machines acting as the vBTSs [26]. However, as seen in Figure 4.2, we need some mechanism by which these frames can be forwarded from and to the physical BTS. In order to ensure that the datapath is able to work in a wide range of application environments, we will require our frame transport mechanism to satisfy three core requirements:

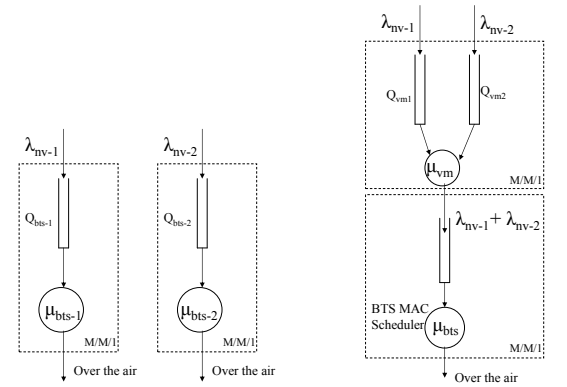
- Protocol independence: We want our datapath to work independent of any layer-3 mechanism, such as IP, IGMP, IPsec, ATM, or any other new protocol. Also, it should be able to work with existing and new protocol mechanisms for mapping network layer addresses to link layer addresses, such as the address resolution protocol (ARP). Independence from protocol requirements allows the vBTSs to customize their own protocol stack and possibly use the same for client interaction, thus providing another dimension for service customization in the hosted MVNOs or the operation of experimental protocol stacks in the testbed context.

- Slice traffic separation: We need some mechanism by which we can separate traffic from and two vBTSs.

- Transparent: This datapath mechanism should be transparent to the slices.

We will discuss two different design alternatives and determine how they may be used to satisfy the requirements discussed above.

Case 1: An L2 Lookup-based Frame Forwarding: One approach to the creation of the datapath from the vBTS to the physical basestation is using pure layer-2 frame information for grouping clients from each slice together, and forwarding them to the BTS queues accordingly. In this case, we use the source MAC address as a classifier on the host of the vBTS substrate to classify traffic and forward it to a pre-determined VLAN interface. Virtual local area networks (VLANs) provide a convenient mechanism for separating and grouping traffic through tagging of MAC frames before and after they arrive at a VLAN device. The VLAN interface identifier is the slice identifier allocated by the the framework. This design satisfies both the protocol independence and the grouping requirements listed above.



(a) Non-virtualized BTS Model (b) Virtualized BTS Model

Figure 4.3: Model of service classes with and without virtualization on the WiMAX BTS.

This design has a fundamental issue which makes it infeasible. In this case, it was assumed that the packets arriving from the virtual interface within the vBTSs will be forwarded irrespective of the updation of their IP and TCP headers. However, since this is not a pure L2 tunnel and frames are being forwarded, the IP and transport checksums need to be re-calculated at intermediate points in the datapath. Frames are forwarded purely on MAC addresses, the packets are not propagated in the kernel stack, the forwarding framework has to implement the checksum re-calculation mechanism. However, such a design places a limitation on the number and type of network and transport protocols that may be supported, since all their checksum re-calculation mechanisms need to be pre-provisioned in the datapath, which is infeasible.

Case 2: An L2 over L3 Tunnel: This approach is an improvement over our previous design. Instead of using layer-2 forwarding, we tunnel all layer-2 frames over a layer-3 based IP network. In this case, protocol independence is achieved because of tunneling, and the grouping of slice traffic is achieved by separating packets within different IP tunnels. This mechanism also allow us to geographically de-couple the vBTS substrate and the ASN-GW i.e. as initially mentioned, we do not need them to be present on the same local area network, and can be housed on different networks as long as the required tunnels are established.

4.6.2 Delay Performance

We will now discuss how the virtualization framework discussed before will impact the overall delay performance of the system. A general approach for achieving bounded queueing delay in multiplexed flow systems is discussed in the GPS study [65]. It is shown that bounded delays for different sessions can be achieved by limiting the queuing delays of admitted flows. This study can be extended for understanding flow admission control on the virtual basestation substrate. However, the goal of the analysis in this section is rather to compare the achievable system delays with the virtual basestation design as compared to the non-virtualized case.

The comparison of the queueing system in the virtualized and non-virtualized system are as shown in the models described in Figure 4.3. We first consider the overall delay for downlink frames in a non-virtualized basestation. Consider the two independent BTSs operating with poisson downlink frame arrival rates as λ_{nv-1} and λ_{nv-2}, and respective service rates of the basestation schedulers to be exponential as denoted by μ_{nv-1} and μ_{nv-2}. If these independent BTSs are modeled as $M/M/1$ queueing systems, then the average delays at each of these BTSs are calculated as $T_{nv-1} = \frac{1}{\mu_{nv-1}-\lambda_{nv-1}}$ and correspondingly $T_{nv-2} = \frac{1}{\mu_{nv-2}-\lambda_{nv-2}}$. Generalizing, for k independent basestations, their individual delays in delivering downlink frames would be:

$$T_{nv-k} = \frac{1}{\mu_{nv-k} - \lambda_{nv-k}} \tag{4.1}$$

To virtualize these independent BTSs, we host them as virtual basestations (vBTSs) within virtual machines which eventually share the same underlying BTS radio as shown in Figure 4.3. In this case we continue with our earlier assumptions that the packets arriving from the virtual BTSs (vBTSs) are poisson distributed with the rates λ_{nv-1} and λ_{nv-2}. Let the service rate with which the operating system redirects packets from each of these virtual devices to the actual WiMAX interface be denoted by μ_{vm-wm}[6]. In this setup the total delay seen by the slices will be the addition of the forwarding delay (T_F) and the queueing delay seen at the BTS (T_{bts}). The delay for forwarding frames from the virtual basestations to the physical basestation is given

[6]Note that assuming two separate servers with equal capacities instead of the one for traffic from two VMs will not change out analysis.

by:

$$T_F = \frac{1}{\mu_{vm-wm} - \sum_{i=1}^{k} \lambda_{nv-k}} \tag{4.2}$$

Since, we are generating traffic for a WiMAX BTS which can typically handle less than 25Mbps of peak traffic, and the packet forwarding from the VMs to the physical basestaion is done in software by the operating system $\mu_{vm-wm} >> \lambda_{nv-1} + \lambda_{nv-2}$. Hence, the forwarding delay from the vBTSs (T_F) is typically very small. Thus the total delay seen by the slices: $T_F + T_{bts}$ can be approximated by T_{bts}. Also, since $\mu_{vm-wm} >> \sum_k \ \lambda_{nv-k}$, it is also clear that the packets arriving at the BTS MAC scheduler are arriving at the rate of $min(\mu_{vm-wm}, (\lambda_{nv-1} + \lambda_{nv-2}))$, i.e. $(\lambda_{nv-1} + \lambda_{nv-2})$.

Now, since we assume that the work of both BTSs in the non-virtualized case is handled by the virtualized radio, we can design the system such that $\mu_{bts} = \mu_{nv-1} + \mu_{nv-2}$. Also, using Burke's theorem [66] we know that the two queuing systems (for the vBTSs and that of the actual BTS) are independent. Hence, we can consider the physical BTS as an $M/M/1$ system with arrival rate of $\lambda_{bts} = \lambda_{nv-1} + \lambda_{nv-2}$ and service rate of $\mu_{bts} = \mu_{nv-1} + \mu_{nv-2}$. Using this information we can determine the steady state delay in the system as:

$$T_{bts} = \frac{1}{\mu_{bts} - \lambda_{bts}} \tag{4.3}$$

$$T_{bts} = \frac{1}{(\mu_{nv-1} - \lambda_{nv-1}) + (\mu_{nv-2} - \lambda_{nv-2})} \tag{4.4}$$

Substituting from equation 4.1, we get:

$$\frac{1}{T_{bts}} = \frac{1}{T_{nv-1}} + \frac{1}{T_{nv-2}} \tag{4.5}$$

We can further generalize the above equation for k virtualized slices to:

$$\frac{1}{T_{bts}} = \sum_{i=1}^{k} \frac{1}{T_{nv-i}} \tag{4.6}$$

Since we know that $\forall i, T_{nv-i} \geq 0$, we can conclude that the delay in the virtualized system will be lesser than the delay observed at any individual non virtualized BTS (T_{nv-i}). The actual overall delay in the system is the sum of T_{bts} and the forwarding delay T_F from the VMs to the physical basestation. However as seen earlier in equation 4.2, the forwarding delay from the vBTSs is very small, $T_{bts} >> T_F$, and hence the overall delay is dominated by T_{bts} which is always lower than that can be achieved in any individual non-virtualized system. It is important to note that this inference will hold true only when the following conditions always hold true: (1) $T_{bts} >> T_F$, and (2) the radio capacity of the virtualized system is equal to or greater than the sum of radio capacities of the individual non-virtualized systems. For example, in terms of rate, if we had two radios which could do *X* and *Y*Mbps each, for the delay to be better in a virtualized case, we would need to have a virtualized radio which could at least support $X + Y$Mbps. It is important to note that in some practical cases, even if the delay performance of the virtualized system is slightly worse than the original non-virtualized systems (by design), we still gain significantly in terms of spectrum reuse in the virtualized case. A similar analysis can be done for uplink delay performance, where we will observe that the overall delay in the system will improve in a virtualized case as compared to *equivalent independent* non-virtualized setups.

This reasoning can be used to further calculate the minimum service rate required on the virtualized setup to achieve the delay which at least matches the best delay performance seen in the non-virtualized case. Let us assume that the slice k has the minimum average delay in the non-virtualized case, given by T_{nv-k}. Hence it is clear that for the virtualized setup to do better than or equal to this delay,

$$T_{bts} \leq T_{nv-k} \tag{4.7}$$

Also, we know that the delay in the virtualized case is:

$$T_{bts} = \frac{1}{\mu_{bts} - \sum_k \lambda_{nv-k}} \tag{4.8}$$

Hence, we can calculate the minimum μ_v to match the minimum delay of any non-virtualized

slice as:

$$\mu_{bts} \geq \frac{1}{min_{\forall k}(T_{nv-k})} + \sum_{k} \lambda_{nv-k} \tag{4.9}$$

As long as the above mentioned lower bound condition on μ_v is always maintained by the system designer, the average delay seen in the virtualized system will be less than or equal to the delay seen in any of the non-virtualized cases.

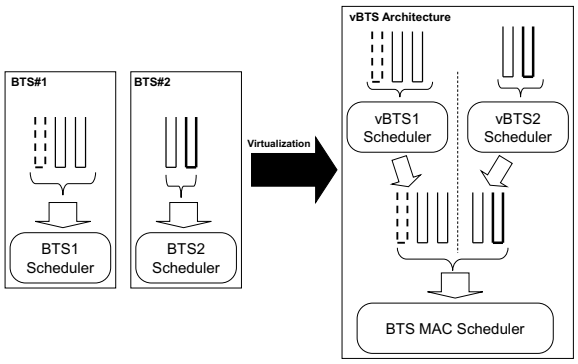
4.7 BTS Radio Virtualization

In this section we will discuss the design issues encountered in the provisioning of radio resource for users across multiple slices. This problem differs significantly from that of radio resource provisioning for the users of a single BTS since the allocation needs to be done both at the slice level (virtual basestation), and at the BTS scheduler. We will specifically deal with two aspects of radio virtualization:

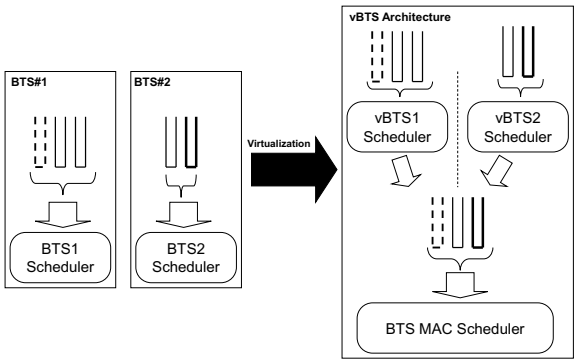
- Service flow abstraction: This part of the study will focus on extending the idea of service flows in the physical basestation to the virtual basestations.
- Virtual radio isolation: This part of the study will focus on isolating the performance of groups of service flows belonging to one virtual radio from others.

4.7.1 Service Flow Abstraction

The conventional WiMAX medium access control architecture relies on a connection oriented MAC for providing quality of service differentiation across wireless clients. It does this by ensuring that both the uplink and downlink traffic differentiation is carried out at the BTS MAC scheduler. The scheduler works with logical entities known as service flows which define a unidirectional flow of packets either from the BTS to the clients, or the other way around, and is identified by a service flow identifier (SFID). These service flows allow the definition of certain QoS features such as maximum throughput, minimum throughput, maximum delay, minimum delay, jitter tolerated, and other features such as the ARQ mechanism used used for the packets controlled by that service flow. The five standard service flow types supported by



(a) Model 1: One to one mapping.



(b) Model 2: Re-using service classes

Figure 4.4: Virtual service flow mapping approaches for the virtual basestation framework. Both the models discussed here can be used without significant changes to the framework itself. The line types in the figure are used to show service flows of different types.

WiMAX are: (1) Unsolicited grant service (UGS) - typically used for voice services without silence suppression, (2) Real Time Polling Service (rtPS) - supports real time traffic such as video, (3) Non-Real Time Polling Service (nrTPS) - used for services such as FTP, (4) Best Effort (BE) - used for most generic traffic, and (5) Extended real time variable rate (ertVR) - for supporting applications such as VOIP with silence suppression. More details of these service flow types and their applications can be found here [67]. In this section we will focus on how different service flow types which are defined in the physical WiMAX BTS are exposed to each of the vBTSs. Specifically, we discuss the issues with provisioning of service classes on the physical BTS and the mechanisms for mapping service classes within the vBTS to the service classes in the physical BTS.

Virtual service flows: Since the idea of service flows is originally maintained only in the physical basestation, we extend this abstraction to the virtual machines (vBTSs) with virtual service

flows. These virtual service flows will be eventually mapped to underlying service flows in the basestation and will be identified using virtual service flow identifiers (vSFIDs). The identifier for these virtual service flows consists of the *slice-id* and the *flow-number*. Appropriate data structures are maintained in virtual basestation framework for identifying flows across slices. We will now discuss how packets will be classified into these individual flows.

Identifying service flows: Service flows are created in a WiMAX BTS depending on a pre-defined set of rules when the client associates with the BTS. A general approach to definition of service flows on the BTS is through the specification of five-tuples: source IP address, destination IP address, source port, destination port and the IP TOS. This information is used by the BTS to classify and identify different service flows, and appropriately schedule their packets. However, this approach has the limitation of requiring it to work with the internet protocol (IP), which may or may not be used in all our vBTSs. Another approach is by using a modified version of our BTS R6 controller [67], which allows the user to create service flows by defining a flow of one flow type such as UGS, BE, ertPS, nrtPS, or ertVR per MAC address associated with the BTS. However, this approach is not the best since it limits us to using one service flow type with every client. Hence, as a compromise in the design, we plan to use simple port address translation (PAT) to carry out the service flow provisioning. This eliminates protocol dependence, and allows the definition of more than one service flow per client per slice. The entire port address space is used at the physical BTS to identify unique service flows. An identical port address space is used by the virtual service flows for classification of their flows. Hence, a virtual service flow being classified on port X may eventually be mapped to a physical service flow in the BTS at either the same port X or any other port Y, as decided by the mapping mechanism discussed below. Note that we can afford replication of port address space for classification since the classification happens at two different levels. For virtual service flows, classification is at the vBTS substrate, and for the physical service flows, classification is in the BTS's MAC scheduler.

Mapping Of Service Flows: The options available for this mapping virtual service flows to physical service flows are as depicted in the Figure 4.4. The first approach shown in the Figure 4.4(a) relies on a simple *one-one* mapping of each of the vBTS's virtual service flows to corresponding service flows in the BTS. The figure depicts different service flow types with

different line types like dotted, plain and bold, both for virtual and physical service flows. Such an approach allows for the clear definition of flow metrics and an easy application of these settings to the BTS MAC scheduler. However, this approach also limits the number of service flows that may be defined to the number of available flow classification ports. An alternate design is shown in Figure 4.4(b), where we define a broad set of service flows in the BTS and reuse them by mapping multiple service flows from the vBTSs to the same flow in the BTS. For example, in this case, all virtual service flows of type UGS will be mapped to a single underlying UGS physical service flow and so on. Hence the physical basestation will only require creation of the 5 fundamental service flow types. This would normally be infeasible with a conventional service flow definition. However, only by using a port address classifier for service flow determination, we can multiplex multiple virtual service flows from different vBTSs on the same service flow in the BTS. This provides an opportunity for conservation of physical service flows at the BTS. A disadvantage with this approach is that some flow specific parameters defined in the BTS such as maximum or minimum throughput, may apply to the entire pipe of virtual service flows encompassed in that single service flow at the BTS, and a lot would depend on the statistical multiplexing of frames from the virtual service flows and channel conditions to the clients. This condition can be partly alleviated by using the radio isolation mechanisms discussed in the following sections, but for now we use our previous approach of mapping one client to a single service flow type.

4.7.2 Radio Isolation

Once the virtual service flows from the vBTSs have been mapped to the physical radio, we need a mechanism by which we can limit the amount of total radio resources consumed by the virtual service flows within a slice. The problem can be formulated as follows. In all further discussion we will use the term airtime as a measure of fairness. Though airtime literally means the amount of time allocated to a single flow or wireless client, in the WiMAX context, this *airtime* actually refers to a set of $time \times frequency$ blocks in the OFDMA WiMAX radio, which we will also refer to as a slice of the radio. Hence, in this context whenever we mention the term airtime fraction, we are referring to the fraction of the WiMAX scheduler resources allocated to that client or flow.

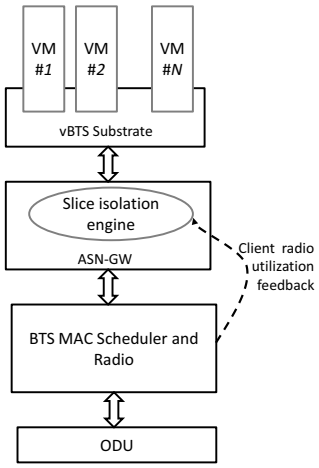


Figure 4.5: The virtual basestation architecture with the slice isolation engine. The VMs are used to indicate virtual machines for different slices. Traffic shaping happens in the slice isolation engine on the ASN-GW.

Problem Statement: If the airtime used by every flow $f_{i,j}$ from a client C_i belonging to slice S_j is denoted by $\phi_{i,j}$, then we need to have a mechanism by which we are able to limit $\sum_{\forall f_{i,j} \in S_j} \phi_{i,j}$, the total airtime consumed by all flows belonging to all clients within the slice. It is important to note that this limit should be enforced while placing no other constraints on the flows within the slice i.e we should not have a requirement that splits slice airtime equally within all flows. This discretion should be left up to the slice (vBTS frame schedulers).

Software Architecture For VNTS: To enforce this radio airtime fairness across slices, we propose and use the virtual network traffic shaping mechanism (VNTS) [27]. This mechanism is responsible for adaptively determining usage of resources by every slice slice and limiting the traffic flowing into the BTS scheduler from every slice. The software architecture for achieving the same is as shown in the Figure 4.5. Traffic from each of the virtual machines (slices) passes through the ASN-GW before being fed to the BTS MAC scheduler. In our virtual basestation architecture all of these frames must pass through a slice isolation engine (SIE) on the ASN-GW. The slice isolation engine is responsible for limiting the aggregate flow (across all virtual service flows belonging to a slice) of traffic to the BTS radio in such a way that the fraction of radio resources used by the slice are conformant with that allocated in the quotas. The SIE is implemented by a click router [36] module based datapath that is responsible for

handling the frames from and to the vBTS substrate. The number of frames per slice that are dropped by the click datapath are controlled by the SIE datapath. The SIE control algorithm is implemented in Ruby, and it communicates with the SIE datapath module through a TCP socket. The control algorithm computes appropriate rates for shaping aggregate traffic from slices, and writes the control information to the datapath module. The rate at which aggregate traffic from the slices needs to be traffic limited is computed by comparing overall airtime usage by all clients belonging to the slice with the allocated quota. If this quota is exceeded either due to a high offered load or poor channel conditions of certain clients within the slice, the shaping rate for that slice is appropriately throttled. Feedback from the BTS radio such as the modulation and coding scheme used to reach the client, throughput, and MAC retries is obtained through SNMP [68] calls to the physical BTS. More details on the exact shaping algorithms are discussed here [27].

4.8 Performance Evaluation

The goal of experiments presented in this section will be two fold:

(1) Experiments on abstraction and programmability will be used to show that we are able to use our virtual basestation framework in experimental scenarios where a physical basestation would be conventionally used. We explain applications in the testbed context by showing how a handoff may be emulated. We also show an application of the programmability of the framework in an MVNO type scenario where a user may chose video service from different MVNOs depending on wireless channel conditions.

(2) Experiments on isolation will be used to show that the isolation engine in our virtual basestation framework is capable of providing performance isolation across slices irrespective of service classes, and the layer at which traffic is scheduled. Isolating slice traffic is essential in both the testbed context (for repeatability), and in the MVNO context (for performance guarantees).

We begin with a description of our prototype, followed by experiments to show programmability, abstraction, and isolation built in the virtual basestation framework.

4.8.1 Virtual Basetation Prototype

All experiments are performed on a Profile-A WiMAX BS deployed at the Rutgers University, WINLAB campus. The BS is capable of up to 8 modulation and coding schemes (MCS) and implements two mechanisms for downlink (DL) rate adaptation. Default downlink rate adaptation is active and changes rate every 200frames. Downlink-Uplink symbol ratio is variable between $(35 : 12) - (26 : 21)$, and is currently set to $35 : 12$ to allocate a higher ratio of symbols for downlink traffic. A Profile-A BS allows the creation of pre-provisioned *best effort* service flows which are used in all experiments. The BS is operating at 2.59Ghz, with a 10MHz bandwidth, and the clients are using a Beceeem chipset.

Client connectivity is provided by PCMCIA cards. Experimenters can include clients as a part of their slices through virtual machine instances [60] running on the vBTS substrate machine. The layer-2 datapath from the vBTSs to the physical basestations is built using layer-2 tunnels over a layer-3 network. The prototype uses a one - one flow mapping i.e.every virtual service flow corresponds to a pre-provisioned physical service flow in the basestation. Finally, the slice isolation engine (SIE) is running on the ASN-GW as a service. The quota allocations to this service are controlled by a slice control grid service. It is important to note that QoS only provides traffic prioritization, while our virtual basestation framework supports weighted fairness provisioning. Rest of the evaluation section will describe different use cases of the virtual basestation framework for the purpose of supporting multiple simultaneous experimenters / MVNOs on the same physical basestation.

4.8.2 Programmability

As discussed before, abstracting the underlying hardware allows for transparent sharing of the underlying resources. We will show how our virtual basestation abstraction along with its support for programmability helps us provide support for some conventional usage cases of a wireless basestation. Specifically, we will discuss how we can support: (1) handoff strategy performance evaluation across different VBTSs, and (2) a sample video rate matching scheme on the edge that does not involve video transcoding.

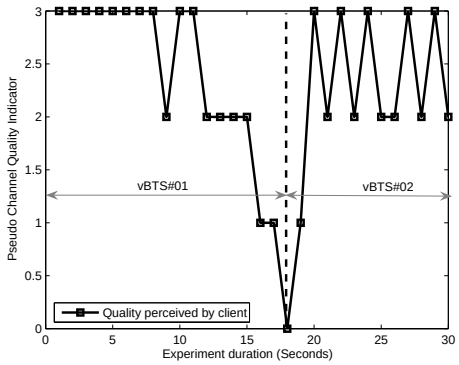


Figure 4.6: Perceived channel quality by the mobile client in terms of the pseudo channel quality indicator. We notice that when the perceived channel quality from $vBTS\#1$ drops significantly the handoff is initiated. We observe that the pseudo channel quality indicator improves dramatically after handoff.

Case 1: Handoff Emulation Across VBTSs. An experimental prototype to evaluate the performance of a handoff mechanism will typically involve the following apparatus: (1) multiple wireless basestations across which the handoff is to be implemented, and (2) some back end gateway from which flow switching will be implemented to a different basestation based on the handoff strategy. We will now discuss how our virtual basestation framework can be used to emulate a generic handoff strategy supported on physical basestations. The goal of this experiment is not to demonstrate a new approach for implementing handoff itself, but rather to show that we can support evaluation of handoff strategies.

To provide a proof of concept demonstration, we consider a simple handoff strategy where the handoff is infrastructure initiated, and is decided directly depending on the channel characteristics seen by the wireless client. We will consider handoff between two physical basestations and both the basestations will execute the following common logic: If any one of the two basestations find that the downlink modulation and coding scheme (MCS) used by their client(s) is below a particular threshold value, it requests the other basestation to accept the client.

We will now explain how the above handoff strategy can be evaluated on our virtual basestation framework. The two physical basestations are now emulated by two virtual basestation slices running on a single physical basestation. Since we are using a single physical BTS, the perceived channel condition of the client to both the vBTSs does not change. Hence to emulate

channel condition change, we divide the modulation and coding scheme (MCS) index range experienced by the basestation for any client into half using a pseudo channel quality indicator (θ). The value of θ is calculated from the actual MCS measurement of the client as follows:

$$\theta_{vBTS1} = max(0, MCS - 18) \tag{4.10}$$

$$\theta_{vBTS2} = max(0, 18 - MCS), \tag{4.11}$$

where MCS is between 16 and 21[7]. Thus if the MCS of the client is greater than 18, the virtual basestation 1 ($vBTS01$) sees good channel quality, and for MCS less than 18, the $vBTS02$ sees good channel quality. Hence we observe that when a client moves away from (or towards) the physical basestation, though the actual MCS used by the basestation monotonically decreases (or increases), the two vBTSs perceive the pseudo channel quality indicator as mapped by the individual functions described above. Though we have chosen simple linear functions for implementing the pseudo channel quality indicators, more realistic conditions could be emulated by using exponentially decaying θ_{vBTS1}, and a complimentary function for θ_{vBTS2}. The programs for detecting client MCS, and deciding if a handoff is necessary are implemented as identical ruby scripts running in two vBTSs. For providing a more realistic emulation, we can also ask our framework to de-register the client, and register again to account for association delays.

A preliminary evaluation of the system is as shown in the Figure 4.6. The experiment consists of a client moving away from coverage of a single physical basestation. The y-axis of the results shows the θ at the wireless client. We notice that as the client moves away, at one point, the θ at the vBTS02 is much better because of which the handoff happens. Using this information, other metrics can be used for measurement such as handoff delays with different mobility conditions. The goal of this experiment was not to show the performance of the handoff mechanism itself, but rather to show the ease with which experiments for understanding performance with handoffs can be emulated.

[7] 16 indicates the slowest supported rate with the most robust modulation and coding (MCS) scheme and 21 indicates the fastest possible rate.

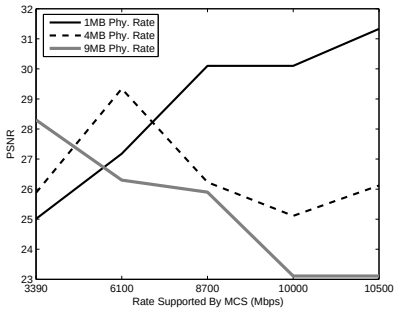


Figure 4.7: Measured value of PSNR for the reception of the same video encoded at different bitrates at different clients connected with different physical layer rates. We see that at low physical layer rates the video encoded at a lesser bit rate does better and vice-versa.

Case 2: Video Service On The Edge. Transcoding independent videos for rate matching has been an active experimental research area [69, 70]. In this example instead of actually doing transcoding, we show how our virtual basestation design facilitates specialized MVNOs. We consider a hypothetical situation where we have three MVNOs, all housed on the same physical basestation. All of these MVNOs specialize in video delivery. However, one of them provides a cheap low resolution video coverage which is typically encoded at 1Mbps, while the other MVNO is expensive, and encodes the same video at 9Mbps. The third MVNO covers the same popular video event at 4Mbps. We envision that when all of these MVNOs are covering popular live events, wireless clients might want to have the flexibility to chose service from either of the three MVNOs.

In our example, we consider that the cost is not a deciding criterion for the subscriber, but the perceived video quality is. The motivation for choosing a specific MVNO for receiving video service is as shown in the Figure 4.7. This plot shows the perceived PSNR [71] at WiMAX receivers for the reception of the same video, encoded at different bitrates for clients connected with different downlink physical layer rates. We see that at lower physical layer rates the video encoded at a lesser bit rate does better and vice-versa. Hence, in our setup described above, it would be ideal if a wireless client is able to select video coverage based

[8]Even though we consider this situation with one shared physical basestation, in a practical situation MVNOs will possibly share more than one physical basestation in the network that support the virtual basestation framework. The results for this use case will remain unchanged though.

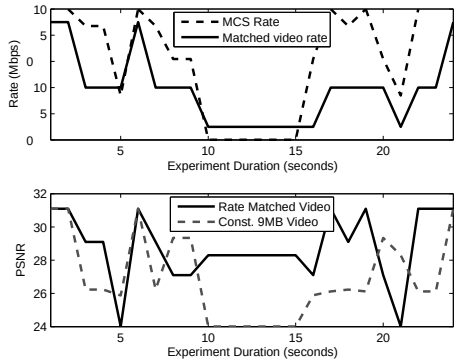


Figure 4.8: An experiment showing video rate matching done by the client. In this case the upper graph shows the matched video rate in comparison with the physical rate. The lower graph shows the PSNR achieved with the adapted video rate. We observe that the PSNR with the adapted video rate is better than that with the best rate.

on its connection with the physical basestation[9]. Under the assumption that the MVNOs agree among themselves and allow their clients to switch among any of their services (as done in providing roaming services), the wireless clients can significantly improve the performance of the video delivered to them by making this decision adaptively. It is important to note that the goal of this experiment is not to show the best way of implementing the wireless video delivery solution, but rather to show that deployment of the virtual basestation framework fosters innovation in the network by housing multiple custom MVNOs on the same BTS.

To show how a user may switch among services provided by the three MVNOs, we implement a simple algorithm on the client that allows it to switch to the MVNO which provides video rate that is closest to its physical rate. Movement across MVNOs is achieved by a mechanism similar to the handoff described in the previous experiment. Results from an experiment with such an adaptive video rate matching done by the client is as shown in the Figure 4.8. In this case we consider a mobile client moving within the coverage area of the BTS over a period of time. The physical layer rate used by the MAC scheduler in the BTS to reach the client is as shown in the upper subgraph in the figure. The matched video bit rate achieved by switching between the three MVNOs is as indicated. Finally, the lower subgraph within the

[9]Conventional dynamic transcoding is highly compute intensive if done by a single BTS since it would require every individual BTS to rate match video flows for every client.

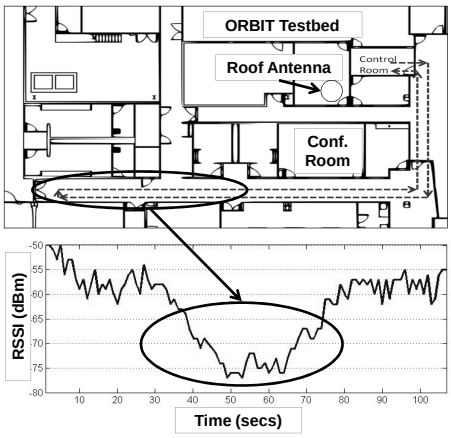


Figure 4.9: Experiment topology for indoor Femtocell emulation experiment. Position of stationary client is fixed in the control room, and the mobile client moves along the marked trajectory.

same Figure 4.8 shows the PSNR as perceived by the wireless client receiving the video. We see that the PSNR with adaptive switching is always better than or equal to that achieved with a high bit rate video. Specifically, the average improvement in PSNR across all locations is at least $1.7dB$. In some cases, the improvement in PSNR is up to $5dB$ over the high bit rate video. It is important to note that these performance improvements in PSNR are independent of the cost savings that are achieved in the spectrum (by adaptively switching to lower bitrate videos). Thus, through this experiment we have shown that our virtual basestation framework is capable of supporting MVNOs for providing specialized services, which can be dynamically selected by the users of the system.

4.8.3 Isolation

The goal of these set of experiments is to evaluate the performance of the slice isolation mechanism. These experiments are performed in two parts, the first set of experiments are just testing isolation across slices which support service classes of different priorities. Later we will show how our isolation mechanism facilitates custom scheduling within the slices.

Case 4: Across Service Classes. 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 *vBTS*1 and *vBTS*2 to their corresponding clients. Client for the flow from

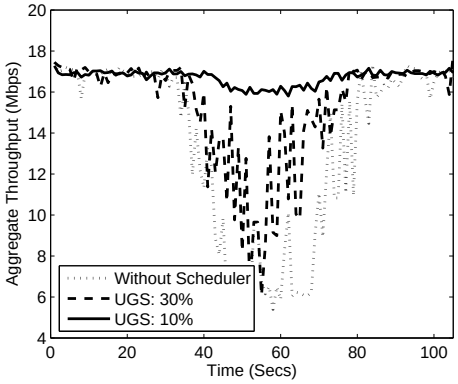


Figure 4.10: Improvement in the aggregate throughput to the wireless clients due to improvement in isolation across slices.

*vBTS*1 (slice1) is stationary, and the client for the flow from *vBTS*2 (slice2) is mobile. The link from each vBTS to the corresponding client constitutes of a slice. The stationary client is located such that it has a channel to interference and noise ratio (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 5.3. 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.

To justify the performance across service classes, 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. The total throughput for both clients is as shown in Figure 4.10. 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 UGS 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

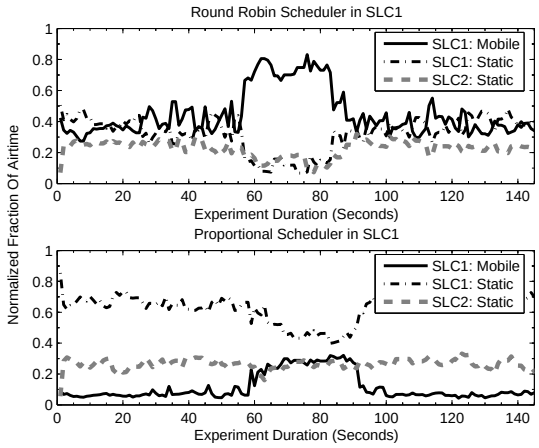


Figure 4.11: Airtime utilization with two custom scheduling schemes for the scheduler running within the first slice.

region with poor coverage. The minimum value of the aggregate throughput improves by up to 193%.

Case 5: Hierarchical Scheduling 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 for 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.

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 5.3. 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 4.11. 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 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.

4.8.4 Discussion

Client Side Interface: We will now discuss the implication of our virtualization mechanism on the interface exposed to the clients from different slices. Since the virtualization is done outside of the physical BTS radio and the MAC layer, the client will still see standard WiMAX beacons as it did from a single physical BTS. However, in this case, after associating with the physical BTS, based on the clients credentials and MAC address, it can be claimed by any operating slice on the BTS. The BSID contained in the beacons from the physical BTS can be generated as a hash of the slice identifiers belonging to the slices hosted on that BTS. In this way, the wireless clients scanning for the physical BTS will be able to determine if the WiMAX BTS in range supports their carrier or not.

Uplink (UL) Slice Fairness: In the case of WiMAX, even though the UL is scheduled, there is no way to directly limit the amount of UL traffic, thus making the problem of enforcing UL fairness complicated. A possibly tractable solution to this problem would require instituting some mechanism that controls or limits the number and type of UL flows defined by each slice thereby limiting the UL usage per slice. It is to be noted that any UL unfairness that could possibly result from a lack of an enforcement mechanism will affect the UL traffic only. This is because the uplink downlink symbol ratio used by the MAC is fixed and the MAC uses time

division duplexing.

4.9 Conclusions And Future Work

This study describes the design of the infrastructure for supporting a virtualized WiMAX framework. Specifically, through an elaborate design discussion this study highlights different options available for building such a virtualized substrate, and the design tradeoffs of choosing one approach over the other. Further, using proof of concept evaluations, it shows how different services that are usually supported on a conventional non-virtualized basestation can be supported on the proposed virtual basestation framework.

Chapter 5

Virtual Network Traffic Shapers

5.1 Chapter Summary

The $802.16e$ standard for broadband wireless access mandates the presence of QoS classes, but does not specify guidelines for the scheduler implementation or mechanisms to ensure air time fairness. Our study demonstrates the feasibility of controlling downlink airtime fairness for slices while running above a proprietary WiMAX basestation (BS) scheduler. We design and implement a virtualized infrastructure that allows users to obtain at least an allocated percentage of BS resources in the presence of saturation and link degradation. Using Kernel virtual machines for creating slices and *Click* modular router for implementing the virtual network traffic shaping engine we show that it is possible to adaptively control slice usage for downlink traffic on a Profile *A* WiMAX Basestation. The fairness index and coupling coefficient show an improvement of up to 42%, and 73% with preliminary indoor walking mobility experiments. Outdoor vehicular measurements show an improvement of up to 27%, and 70% with the fairness index and coupling coefficient respectively.

5.2 Introduction

One of the general trends in wireless networking research is the growing use of experimental testbeds for realistic protocol evaluation. Open networking testbeds such as ORBIT [28], Dieselnet [72] and Kansei [73] have been widely used in the past 5 years for evaluation of new wireless/mobile architectures and protocols based on available radio technologies. With the emergence of so-called "4G" networks, there is a need to support open experimentation with wide-area cellular radios such as mobile WiMAX or LTE. A recent initiative by GENI [16] is aimed at making an open WiMAX base station available to GENI and ORBIT outdoor testbed

users. As a part of this initiative, we address the design challenges of integrating a WiMAX basestation as a part of a virtualized wireless testbed.

Virtualization of the WiMAX Basestation (BS) provides a convenient approach to provide separate environments for different slices. A slice refers to the subset of BS resources and WiMAX clients allocated to a user/experimenter. The problem of ensuring radio fairness and policies across slices is specially hard since the channel for mobile wireless devices changes continuously, thereby consuming varying amount of resources at the BS transmitter.

This study describes challenges in ensuring air time fairness when access to the 802.16e scheduler is unavailable. Evaluation is provided for a carrier grade BS deployed at the Rutgers university. The WiMAX clients are accessible to experimenters through virtual machines (VMs) running on the ORBIT network [28]. The entire experimenter space from the VMs to its corresponding clients refers to a single slice. Implementation and evaluation of our proposed $VNTS$ architecture is provided under both walking and vehicular mobility.

Rest of the chapter is organized as follows. Section 8.6 provides a brief survey of related work. Section 5.4 provides pilot measurements that help justify the use of our $VNTS$ mechanism for administering fairness policies across slices. Section 5.5 discusses details and design considerations for implementation of the VNTS architecture. Section 5.6 provides a thorough evaluation of the system under varying network conditions. Finally, Section 5.7 gives concluding remarks.

5.3 Related Work

WiMAX being a relatively new technology, most of the recent work in this area has been focussed on theoretical modeling. There have been significant efforts with the development of models to modify the scheduler for quality of service [74–77] guarantees and or for ensuring fairness [78, 79]. However, in our problem we take a practical approach that QoS is provided as per pre-set classes by the built in proprietary scheduler. By treating the scheduler as a black-box device, we provide an architecture that provides air time fairness across slices.

A study in slice control [24] has addressed issues of space versus time multiplexing of 802.11 links for interference minimization. However, since we operate in $802.16e$, links with

$Parameter$	$Value$
$Channel\ \ Rate$	Adaptive
Frequency	2.59GHz
DL/UL Ratio	$35:12$
Bandwidth	10Mhz
Client Chipset	Beceem

Figure 5.1: Basestation (BS) settings for all experiments. Explicit change in parameters are as mentioned in the experiments.

a single BS are interference free. A token-passing based air time fairness mechanism was implemented in [80]. Though this approach is suitable for implementation on 802.11 devices, it does not demonstrate performance with varying traffic loads, and frame sizes. To the best of our knowledge we are not aware of any study which implements an air time fairness mechanism for user groups in $802.16e$ devices at this time.

5.4 Motivation

We begin with a brief description of the experiment apparatus followed by a pilot experiment to motivate the study.

5.4.1 Hardware Setup

The deployed BS is capable of up to 8 modulation and coding schemes (MCS) and implements two mechanisms for downlink (DL) rate adaptation. DL-link adaptation is active and changes rate every 200frames. Downlink-Uplink symbol ratio is variable between $(35:12)-(26:21)$. A Profile-A BS allows the creation of pre-provisioned Best effort service flows which are used in all experiments. Other BS settings unless mentioned otherwise are as shown in Figure 5.1.

The system architecture integrated in the ORBIT framework is as shown in the Figure 5.2. The Application Service Network gateway $(ASN-GW)$ is used for connecting the basestation to the outside world. The internal control interfaces of the basestation (network and RF) as well as the ASN-GW are networked as a part of the *instrument* network. The external interface of

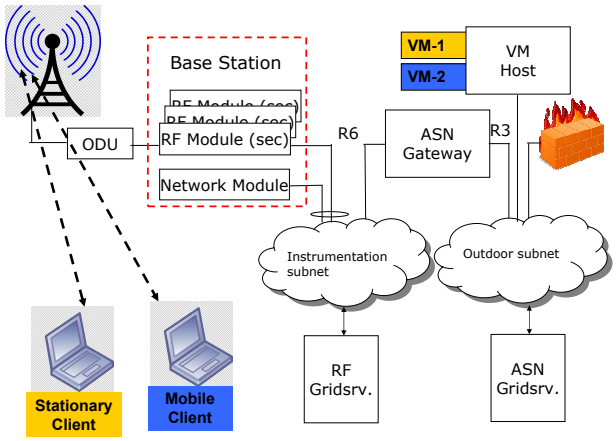


Figure 5.2: Integration of the WiMAX setup into the ORBIT testbed.

the ASN-GW is a part of the *outdoor* network. Client connectivity is provided by PCMCIA cards. Due to limited hardware we have to limit one client per slice. However, this can be easily extended to multiple clients per slice using our setup. Experimenters can include clients as a part of their slices through virtual machine instances [60] running on the *VM Host* machine. Access through the VM host machine also allows administrators to control slice access to pre-provisioned service flows, thereby determining slice QoS. It is important to note that QoS only provides traffic prioritization, while our mechanism aims at providing fairness.

5.4.2 Baseline Experiment - Femtocell Mobility

Our baseline experiment emulates mobility in a Femtocell deployment, and is repeated in all indoor measurements. As shown in Figure 5.2, the virtual machines VM1 and VM2 are sending traffic to a stationary and and mobile client respectively. 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 5.3. 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 VM is configured to saturate the link to its client with UDP traffic. Other parameters as shown in Figure 5.1. The observed downlink throughput for both the clients is as shown in Figure 5.4(a). We observe that as the mobile

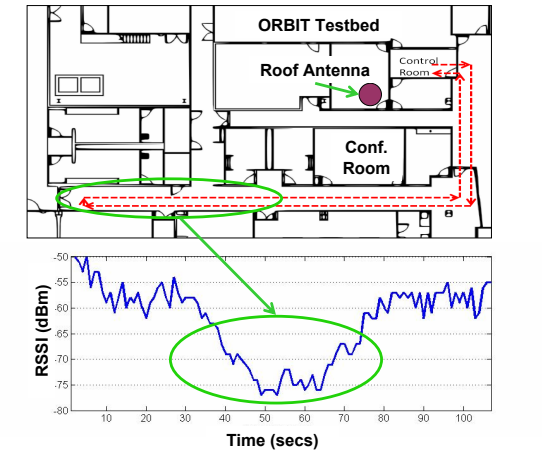
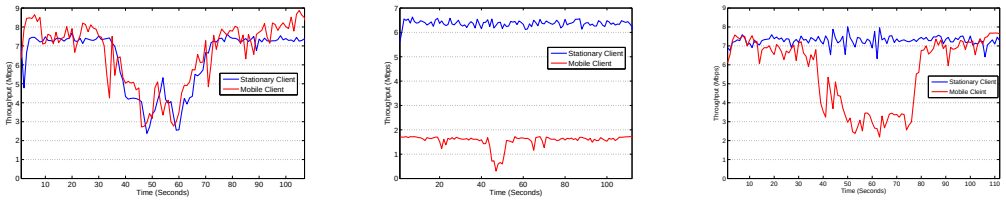


Figure 5.3: Walking path used by the mobile client for indoor experiments *(Topo-1)*. Typical, variations observed in RSSI are as shown.



(a) No Shaping (b) Static Shaping (c) Adaptive Shaping (VNTS)

Figure 5.4: Performance improvement using VNTS for walking mobility shown in Figure 5.3 (*Topo-1*).

client reaches areas where the RSSI drops below 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.

For simplicity, we will consider the initial problem of assigning 50% of BS resources to each of the two clients. A conservative solution to this problem is through static shaping. Assuming that we have some way of knowing about the movement of the mobile client, we could calculate throughput based on 50% channel time at the slowest MCS that would be used by the BS to reach the mobile client. In this case, the basestation uses $\frac{1}{2}QPSK$ when link quality

deteriorates the most. Hence we use this information to statically shape the throughput of the slice with the mobile client. The goal of this shaping mechanism is to limit the offered load for the slice in such a way, that the slice can use only the allocated share of basestation resources. The results of an experiment that demonstrates performance with such a static shaping policy is as shown in Figure 5.4(b). The measurements show that this time the throughput of the slice with the stationary node becomes independent as a virtue of the shaping for the mobile client. Pro-active traffic shaping results in lower throughput for the mobile client, which helps prevent the saturation of the basestation and eliminates coupling between slice performances. However, we also observe that while fairness increases, aggregate downlink throughput decreases significantly even when channel to the mobile client is good. Static shaping will also require the knowledge of mobility and link saturation in advance.

Hence, no shaping gives better overall channel utilization with no slice fairness, and static shaping gives us better fairness with significantly lesser utilization. To alleviate this problem we propose and implement the virtual network traffic shaping (VNTS) technique, which adaptively controls slice throughput. Results from the experiment repeated with *VNTS* are as shown in the Figure 5.4(c). Even as the channel for the mobile client deteriorates, the VNTS mechanism is able to appropriately limit the basestation utilization for the mobile client (slice) thereby providing fairness to the stationary client. The next section will provide details on the design and working of the VNTS architecture.

5.5 VNTS Architecture

The VNTS architecture is as shown in the Figure 5.5 and consist primarily of the VNTS engine and the VNTS controller. The VNTS engine is responsible for performing the traffic shaping from the virtual machines, and the VNTS controller component is responsible for controlling the VNTS engine by determining slice throughput and current operating conditions.

5.5.1 VNTS Engine

The virtual network shaping engine is implemented in the Click Modular router [81] run in user mode as shown in Figure 5.5. This mechanism is responsible for-

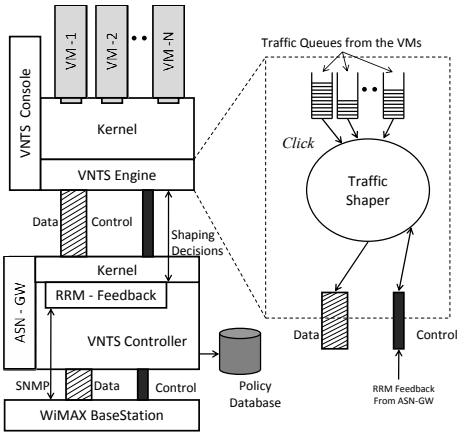


Figure 5.5: Overview of version 1 architecture for virtualization with the Wimax Basestation

Algorithm 2: Algorithm for adaptive virtual network traffic shaping (VNTS) controller.

Input: MCS Rate A_i, Slice Weight W_k,
Number of clients per slice N_k
Output: Shaping rate per client γ_i
while $True$ **do**
  foreach *Slice* k **do**
    $N_k = getNumClientsForSlice(k)$ **foreach** *Client* $i \in$ *Slice* k **do**
      $SNMPGet(A_i)$
      $\gamma_i = A_i \times \frac{W_i}{N_k}$
      Set(γ_i)
  $sleep(UpdateInterval)$

1. Separating VM traffic based on a slice identifiers. We are currently using MAC identifiers of virtual machine interfaces as the slice identifiers.

2. Arping, routing and shaping as per policies to and from the virtual machines.

3. Providing element handlers that allow dynamic control of virtual machine traffic by using the VNTS controller.

5.5.2 VNTS Controller

Resource blocks at the BS are defined as a set of time - frequency tiles in the OFDMA radio that are allocated to individual links by the BS scheduler. Isolation between slices is compromised when the WiMAX basestation runs out of resource blocks and the slice with a poor link, eats

into the resources of the slice with a better link. Since allocation of these resource blocks is only accessible to the BS scheduler, we aim to alleviate this problem by input to the scheduler.

The controller is based on our observation that the number of resource blocks used per slice at the WiMAX BS are directly proportional to the offered load per slice in terms of channel time required per second. Hence the end goal of our VNTS controller is to detect saturation, and limit the offered load (in Mbps) per slice in such a way that channel time required per second for every slice scales in proportion to the weight assigned to that slice.

The complete algorithm is as shown in Algorithm 2. The rate at which a client is receiving downlink data is determined by sending $SNMP\text{-}GET$ queries to the basestation. The current rate adaptation algorithms on the WiMAX BS change MCS every 200 frames. Hence, there is a good chance that the MCS received from the SNMP query varies from the MCS used for the batch of frames for which shaping is done. Averaging the MCS bit rate obtained for every 200 frames would provide a solution to this problem, but this would demand a large amount of control overhead and make it infeasible with current latencies for SNMP call completion. Instead, we shape the traffic every 1sec with a conservative estimate of the MCS based on the BS feedback. The control algorithm scales the saturation throughput for slices in proportion to their weights. The $Set()$ function is used to remotely set throughput limit for each virtual machine. $UpdateInterval$ is a parameter that determines how often the control loop is run. Both of these parameters are controllable. By default, this loop is executed every second, to achieve repeated control. The sleep duration for the loop can eventually be made adaptive based on observed link conditions.

5.6 Evaluation

In this section we present results from experimental evaluation of the VNTS architecture. We begin with a brief description of the metrics used for slice fairness followed by results from indoor and outdoor experiments.

5.6.1 Metrics

In our evaluations, we modify and use the Jain fairness index [49] for determining weighted fairness across flows for varying levels of offered loads. Let the throughput observed at a client i be given by T_i, bit rate achieved with the current modulation and coding scheme for the slice i be given by A_i. Number of clients (flows) belonging to a slice k be given by n_k. Total number of such concurrent slices is given by N. Fraction of channel time used by all links in slice k can be calculated as ϕ_k-

$$\forall \ \ Clients_i \in Slice_k, \ \ \phi_k = \sum_{i=1}^{n_k} \frac{T_i}{A_i} \tag{5.1}$$

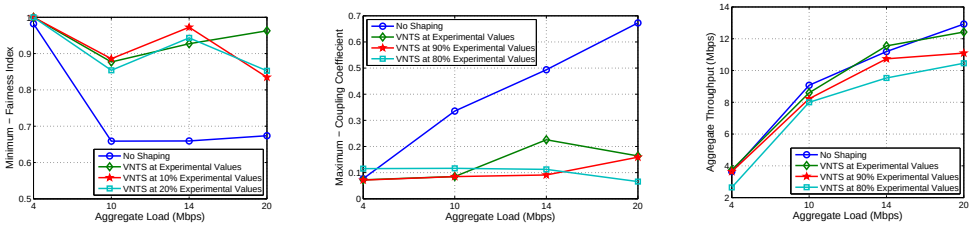
$$I = \frac{\left(\sum_{k=1}^{N} \phi_k\right)^2}{N \times \sum_{k=1}^{N} \phi_k^2} \tag{5.2}$$

The fairness index (I) determines the global variation in channel utilization across slices. We further modify the index to evaluate fairness under saturation with different slice weights while also accounting for performance deterioration due to bad channel quality. To measure worst - case performance, our experiments will determine minimum value of this index for different scenarios. Due to the availability of only two clients at this time, each slice constitutes of a single client.

Another metric used in our measurement is defined as the coupling coefficient. The coupling coefficient is used to measure the performance impact of the mobile slice on throughput obtained by a stationary slice. The coefficient C_k for every client k is measured as-

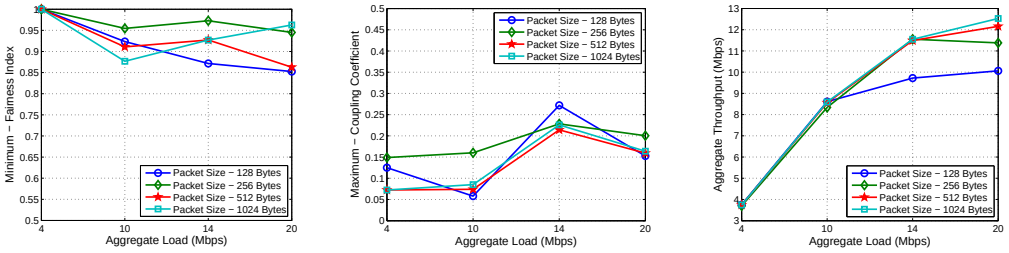
$$C_k = \frac{(T_k - T_k^{fix})}{T_k} \tag{5.3}$$

T_k^{fix} is the average throughput measured at client k, when all clients are stationary and in similar channel conditions. T_k denotes the average throughput of the stationary client k over one second, with the other client being mobile. The smaller the coupling coefficient, lesser the coupling between client k and other clients. In our measurements we focus on improving the worst case performance. Hence we will plot the maximum value of the coupling index seen



(a) Fairness Index (max) (b) Coupling Coefficient (min) (c) Channel utilization

Figure 5.6: Performance for indoor walking experiments *(Topo-1)* with various shaping policies.



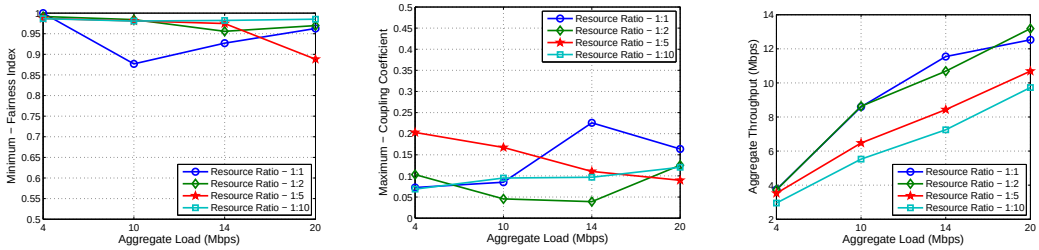
(a) Fairness Index (I) (b) Coupling Coefficient (C_k) (c) Channel utilization

Figure 5.7: Performance with varying frame size. Frame sizes are varied for both flows.

over the duration of the experiments in all measurements.

5.6.2 Policy Conservativeness

Limiting slice throughput to more conservative values will enable better performance isolation between slices. However, this will also lead to a lesser net utilization of the available resources at the BS. To determine optimum rates for shaping, we perform more experiments with walking mobility using the same layout from Figure 5.3.



(a) Fairness Index (I) (b) Coupling Coefficient (C_k) (c) Channel utilization

Figure 5.8: Performance with ratio of flow weights across slices. Flow weight for the mobile client is fixed while that for the stationary client is increased.

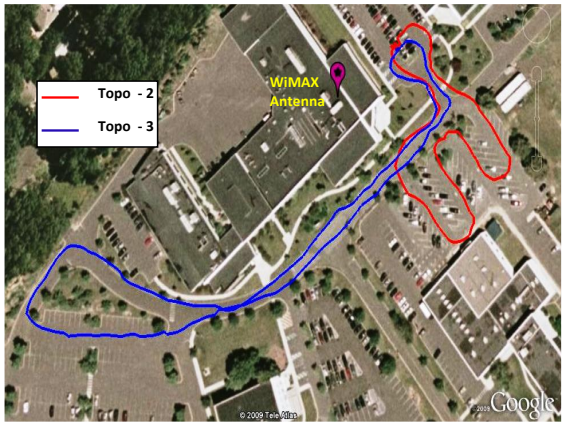
We measure performances by shaping at 80% (most conservative), 90% and 100% of the prescribed channel rates for each of the modulation and coding schemes. We repeat the same baseline experiment as in the previous section, by having a stationary client and a mobile client moving as shown in Figure 5.3. Figure 5.6(a) plots the minimum value of the fairness index from equation (5.2) for each slice based on the radio resources promised to that slice as a function of aggregate load on the system. Without shaping, the fairness index hits lows of up to 0.67. However, with adaptive shaping at 80%, 90% and 100% of saturation values, we see the index ranging from $0.82 - 1$ while being independent of the system load. We also see an improvement of up to 42% in certain cases. Performance of the coupling coefficient for stationary client is as shown in Figure 5.6(b). We observe that without the VNTS mechanism coupling between slices reaches up to 0.66 under saturation, which can be limited to around 0.22 by the use of the VNTS mechanism. It is also seen that irrespective of the conservativeness of the shaping scheme, similar reduction in coupling is observed. Since we do not see a significant difference in fairness or coupling by varying conservativeness, we use channel utilization as a metric for deciding the best shaping rates.

Figure 5.6(c) shows the total channel utilization of the slices as a function of aggregate offered load. These results are as per intuition and show that the mean channel utilization across all system loads is the best when we use the prescribed channel rates without making the policy conservative. Since the throughput drop due to the use of a conservative policy is much more than the benefits in fairness, we chose prescribed channel rates as the default shaping values.

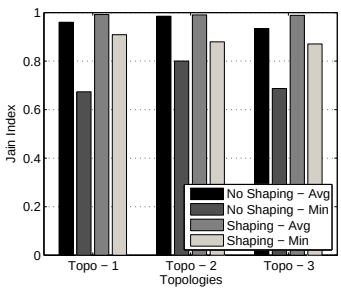
5.6.3 Varying Frame Sizes

As seen in previous experiments, slice fairness improves significantly after using our VNTS mechanism. We will now evaluate the performance of our system in the presence of varying frame sizes for slices. Experiment setup is the same as described earlier. Frame sizes used for the UDP traffic are varied as $128, 256, 512$ and 1024Bytes.

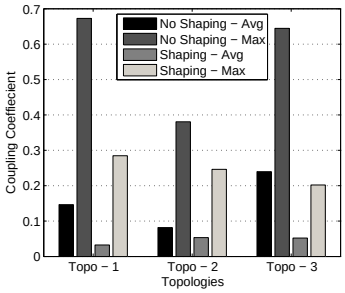
Figure 5.7(a) shows the minimum value of the fairness index with the use of different frame sizes as a function of aggregate offered load across clients. We observe that the fairness index varies between $0.85 - 1$. This shows that the fairness obtained across slices is independent



(a) Experimental topology



(b) Fairness Index (I)



(c) Coupling Coefficient (C_k)

Figure 5.9: Relative performance of indoor and Vehicular experiments.

of the frame sizes used by the slice. In all experiments, worst case fairness index or coupling performances are similar for loads $> 10Mbps$ because, as seen in Figure 5.4(a), the aggregate link capacity with mobility drops below $10M$ causing the BS to run out of resources despite low offered loads.

Results in Figure 5.7(b) show that despite varying the frame size for slices, we see that the coupling coefficient for the stationary client reaches a maximum value of 0.27 which is significantly lesser than that achieved without the VNTS mechanism. Finally, results in Figure 5.7(c) show that the VNTS mechanism works fairly across the clients and the aggregate throughput is similar to that achieved without shaping.

5.6.4 Varying Flow Weights

To study weighted fairness performance, we vary the weights of downlink resources given to the mobile and stationary slice in the following proportion- $1:1$, $1:2$ $1:5$, and $1:10$. The

results are plotted for different values of aggregate load on the system and are as shown in the Figure 5.8(a), 5.8(b) and 5.8(c). Each client loads the system equally during these tests. Across all loads we observe that the minimum value of fairness index in Figure 5.8(a) is maintained under acceptable limits. Typically as the weight for the stationary client is increased, fairness improves since we expect lesser impact of unfairness from the mobile slice. The Figure 5.8(b) shows that maximum coupling for the stationary client is also kept within limits (< 0.25) under all experiment conditions. Channel utilization is only limited by the offered load, and weights assigned to each slice.

5.6.5 Vehicular Measurements

To further validate the working of our VNTS system, we perform experiments with the mobile client placed in a vehicle. Experiment setup is same as that in the previous sections, with the exception that performance is determined only under saturation conditions, where each slice has a downlink offered load of 10Mbps. Outdoor results are measured for two topologies (*Topo-2,3*) as shown in Figure 5.9(a). Average velocity of the vehicle is maintained greater than $10Mph$ for both *Topo-2,3*. These results are compared with those obtained from the indoor experiment *Topo-1* as shown in Figure 5.3.

Figure 5.9(b) shows the modified fairness index for the three experiment setups. We observe that in all the cases, we obtain better fairness using our VNTS mechanism. Average values of the fairness index show improvements across all the topologies, and the worst case performance (minimum fairness points) is significantly improved with the use of VNTS. Figure 5.9(c) plots the performance of the coupling factor for the stationary client across all topologies. In the average case, we observe that the coupling reduces for all topologies with the use of our VNTS mechanism. The VNTS mechanism also improves the worst case performance for all topologies with reduction in coupling of up to 70% in some cases. Thus, using our VNTS mechanism, we are able to significantly improve fairness across slices even under vehicular mobility.

5.7 Conclusions And Future Work

This study motivates the need for an airtime fairness mechanism to be built with a virtualized WiMAX basestation. Design and prototyping results are presented to demonstrate the feasibility of such an approach outside the $802.16e$ scheduler thereby making the implementation platform independent. Preliminary evaluation of the VNTS architecture shows that it is possible to assign a certain percentage of the BS resources to experimenters or slice users while assuring significant isolation even under bad channel conditions. Future work involves more extensive evaluation, and using more client feedback to improve performance.

Chapter 6

Virtual Wireless Network Mapping

6.1 Chapter Summary

Virtual network mapping is a useful tool for mapping virtual networks to physical mesh networks. This chapter extends this idea from the wired world by showing potential applications of virtual *wireless* network mapping, on wireless mesh infrastructure. Specifically, using examples, this study motivates the need for such a mapping service and discusses how virtual networks could be used over wireless mesh networks for providing wireless points of presences (POPs) as additions to cellular voice and data services. Specifically, this chapter will show that the deployment of virtual networks over wireless networks varies significantly from that of deployments over wired networks, which has been studied earlier. Using a focus on provisioning capacity to virtual networks from different points on the mesh to the sink, we show how such requirements could be met. We propose two heuristics to solve the mapping problem, one based on a greedy static allocation (GSA) approach, and the other based on a greedy dynamic re-allocation (GDR) strategy. Evaluations based on determining the performance for different mesh topologies, comparison with a similar performance on a wired mesh show that the mobile network operator can easily use this mechanism for maximizing its profits.

6.2 Introduction

A virtual network topology can be defined as a network topology description that when realized on a physical network of nodes, results in the virtual network behaving exactly like a physical network with the same specification. The virtual network topology description is usually dictated by application or the service providers requirement, while the physical network is designed to support a diverse set of virtual network topologies. This fundamental design

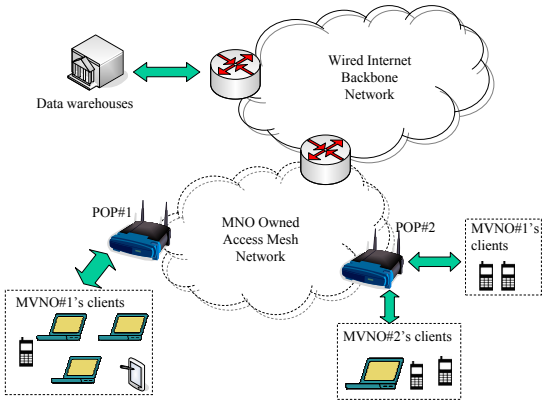


Figure 6.1: Mobile virtual network operators (MVNOs)- This broad architecture diagram depicts how the backhaul, mesh network operator, and the MVNOs operate together. A mechanism for mapping would be needed to support MVNOs on the mesh operators infrastructure.

paradigm of decoupling the physical network design from the virtual network design allows the mobile network operators (MNOs) to provide a more generalized infrastructure, which finds wide application resulting in better utilization of hardware and spectrum. Other benefits of such a network virtualization approach include fault-tolerant network deployment, and load-based leasing [6] among other applications.

6.2.1 Motivation

Though virtual network mapping has been studied extensively in previous literature, the problem has remained relatively untouched in the wireless domain. We argue that the reason for the wired virtual network mapping problem not extending to the wireless domain is essentially due to the nature of application requirements for the wireless domain. To elucidate our point, we consider the following two (possible) applications for virtual wireless networks to be used:

Case (1) Mobile virtual network operators (MVNOs): Consider the example shown in Figure 6.1. Mobile virtual network operators (MVNOs) are entities which rely on providing enhanced network coverage to their clients while not owning any network hardware themselves. MVNOs are typically enabled by Mobile network operators (MNOs) who own the network hardware. In this case we consider an application where the MVNO intends to provide supplementary connectivity to its core coverage through the addition of WiFi mesh hotspots to allow

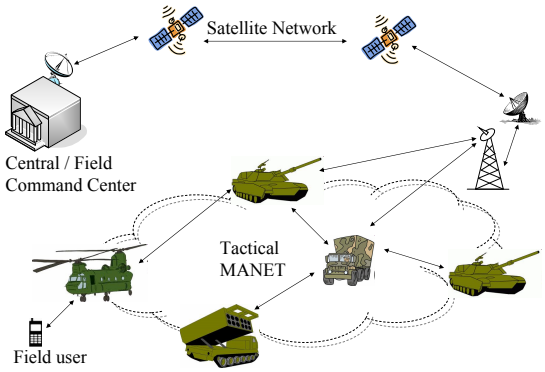


Figure 6.2: Tactical Networks- This network architecture depicts how a typical mobile ad hoc network might be setup with multiple points of usage for reaching some backbone or command center. A wireless topology mapping mechanism could be used to map and provision different services across the network.

their clients to connect to the backend. This is particularly useful in areas such as airport arrival terminals, or movie halls, where flash crowds can cause congestion on the wireless edge. In such cases, the MVNO could leverage the Mesh framework provided by the MNO to provide additional WiFi based coverage to its clients. This coverage requirement could vary across locations with time of the day and would depend on the service level agreement reached between the MNO and MVNO. In this study we will discuss how our virtual wireless network mapping approach could be used by MNOs to maximize their profits in deploying multiple MVNOs on their wireless Mesh infrastructure.

Case (2) Tactical Networks: Another application for our virtual network mapping algorithm could be for tactical networks as shown in Figure 6.2. Typically, a quasi-stationary tactical network which relies on deploying communication nodes at certain points will be setup with the end goal of allowing access to some back office or command center from those nodes. In such a case, the command center may decide to provide specialized ”virtual network mappings” across the mesh so that certain nodes are able to securely sustain a voice connection to the command center. Another requirement could be a data-intensive but a not so secure transfer of huge map files detailing access routes, and so on.

From the applications discussed above, we can deduce the following. The reason why the virtual network mapping problem from the wired world does not translate directly or find applications as is in the wireless world is because of a few fundamental differences in requirements.

(1) Both applications (MVNOs and MANETs) of virtual wireless network mapping are concerned about last-mile connectivity, rather than emphasis on obtaining a topology itself on the physical network, which is typical in wired virtual network mapping. (2) Since the physical network is a wireless mesh, instead of a wired network, we also observe that the accounting of resources at every node for the purposes of mapping are very different from that in the wired domain. Taking these requirements into consideration, we propose our virtual wireless network mapping (VWNM) algorithm.

Specifically, the contributions of this study will be:

1. We discuss and motivate how a virtual wireless network mapping algorithm would prove useful in the context of wireless mesh infrastructure.

2. We discuss how a wireless network mapping problem may be simplified from an arbitrary sub-graph isomorphism problem to a simplified resource allocation problem.

3. Our study takes into account a cost function based on the working of CSMA radios, a popularity metric based on demand at particular points of presence and presents an approach for performing this mapping.

4. Through comparison of performance on three standard wireless topologies, we quantify the results that can be obtained from the mapping algorithm.

Rest of the chapter is organized as follows. Section 8.6 presents a brief discussion on the related work in the area of virtual network mapping. Following this, Section 6.4 formally defines the problem and explains our approach to virtual wireless network mapping. Section 6.5 presents results from the simulations under different scenarios. Finally, Section 8.7 discusses the conclusions and future directions of the study.

6.3 Related Work

We discuss related work for this study in two parts. We will begin by discussing work done in understanding the virtual network mapping problem itself. Following this, we will discuss the progress in a related work of network component virtualization, which allows us to extend the idea of virtual network mapping to wireless devices.

Wired Virtual Network Mapping: Several efficient VNMP heuristics solving different variants of the VNMP have been proposed in the past years [9, 82–85]. Some of these studies deal with data rate constraints for wired links [85], while some studies assume that the link mapping is known before hand [84]. In [82] the mapping is done by simulated annealing, but the problem is limited to topology constraints. Ref. [9] presents a two stage mapping algorithm, handling the node mapping in a first stage and doing the link mapping in a second stage, based on shortest path and multi commodity flow detection.

Network Component Virtualization: Several approaches have been independently proposed for virtualizing individual network components. The PlanetLab testbed [7] relies on time sharing an inter-network of nodes that are virtualized. One of the first approaches for virtualizing core network components has been proposed in the VINI [8] study that attempts to run multiple router instances on the same physical machine. In terms of wireless networks, the virtual access point [86] framework allows a single physical WiFi access point to emulate multiple logical entities. This idea is extended for providing isolation across virtual access points in [25]. A similar attempt to provide multiple virtual basestation transceivers has been discussed in [26, 27].

Thus we have seen that our approach addresses aspects of mapping virtual networks on wireless meshes which differs from previous mapping studies. Also, we see that the progress in the network component virtualization domain has begun an advance into wireless devices thus providing a framework for housing multiple virtual networks on the same physical wireless network.

6.4 Wireless Mapping Methodology

6.4.1 Mapping Approach Overview

To tackle the PoP mapping problem defined above we follow a two-step approach.

- Step 1: The first step of the problem involves analyzing the capacity of the physical network described by G_p by determining the airtime *cost* metric at each node.

- Step 2: And the second step would be to use the set of incoming requests for mapping radio resources at appropriate PoPs.

For performing the first step the mesh network operator can leverage from a comprehensive body of literature that deals with mesh planning and resource allocation and management [87]. Though we will propose an approach for performing this allocation, we will not focus much on this part of the PoP mapping process. For solving the second step of the problem, we propose and evaluate two mapping algorithms: (1) GSA and (2) GDR, which will be discussed in detail in the following sections. We will begin with a description of the approach taken for pre-processing and resource allocation on the physical substrate.

6.4.2 Physical Substrate Pre-Processing

For the purpose of physical substrate processing, we use a simple algorithm for resource provisioning on the substrate. There are other more comprehensive approaches for resource allocation [87], which we do not discuss here. Our mapping algorithms discussed here will work independently of the substrate resource allocation algorithm.

Capacity allocated at every PoP as a part of this substrate pre-processing phase is defined in terms of bandwidth. This bandwidth at the PoP will be used to provide wireless connectivity to clients at the PoP. This resource allocation strategy would require accounting of appropriate capacity allocation on the nodes on-path to the network sink. This resource accounting is significantly more complex in wireless networks due to the inherent nature of the wireless medium. Before we begin a description of our resource allocation approach, we present some assumptions about the physical substrate itself-

- Every transmitter in the physical substrate is able to send frames at different physical rates to different destinations depending on link conditions to the receiver.

- Each node is running a version of the CSMA-CA [88] protocol.

Our substrate pre-processing phase aims to achieve equal resource allocation at all physical nodes in the network. In order to achieve this, we need information on routing path of packets from every node to the network sink. This allows us to calculate $\sum_{i=i}^{N} F_i^k$ which is the sum

Algorithm 3: The greedy static allocation (GSA) strategy for resource mapping at wireless PoPs on the mesh.

Input: $\{V_p, L_v\}$
Output: $\{M, Rev, Cap\}$
$Cap = Rev = 0$;
$Sort\ \ phyiscal\ \ nodes$
$V_p = sortPhyNodes(V_p, cost, popularity)$;
for i = $1 : num\_p\_nodes$ **do**
    $L_v = SelectUnMappedVnodes(L_v, V_p(i))$;
    $N = size(L_v)$;
    # $Generate\ \ knapsack\ \ parameters$
    $Values = revenueAchievable(L_v)$;
    $Weights = capacityRequiredAtPoP(L_v)$;
    $Capacity = phyCapacity(V_p[i])$;
    # $Invoke\ \ knapsack.$
    $amount = sack(weights, values, capacity)$;
    $items = find(amount)$;
    $mapped\_nodes = L_v(items)$;
    # $Calculate\ \ allocations.$
    if $items > 0$ **then**
        $V_p = UpdatePhy(mapped\_nodes)$;
        $Cap = Cap + CapAlloc(mapped\_nodes)$
        $Rev = Rev + Value(mapped\_nodes)$;
        $PopulateMappings(M, items)$

of fraction of air-times (F^k) used at every node i on the path to the sink from the node k. This value is further compensated by the air-time loss at neighbors of all intermediate nodes because of falling into the carrier sense region. Hence, we are able to determine the cost of reaching the network sink from every node in the network, and we can use this information to calculate the maximum possible transmission rate from every node. We can easily incorporate multiple sinks in the mesh network as long as the routing strategy for every node is known. This routing strategy could rely on using either a single sink or multiple sinks to reach the backhaul. However, this extension is not discussed further in the study. Also, note that we do not take into account interfering links in the substrate pre-processing phase. Rather, this is taken into account while allocating resources at individual nodes as discussed in the GSA and GDR algorithms.

6.4.3 Greedy Static Allocation (GSA)

To begin a discussion of the mapping algorithms we will describe how the PoP mapping requests are made by MVNOs. The PoP mapping request contain the following information: (1) *Node characteristic descriptor* of the PoP, (2) Capacity desired at PoP, and (3) Bid for that corresponding capacity for that characteristic of the PoP. We define the characteristic descriptor of a PoP as any metric that could be used to describe the PoP. Examples of characteristic descriptors are: location type (movie halls, coffee shops, schools), or degree of density metric (densely populated, medium density, sparse). In this case, we propose using a popularity metric as a characteristic descriptor[10]. The capacity desired at every PoP is defined in terms of the aggregate bit rate desired at that PoP. Finally, we define the bid as the aggregate amount in any units that an MVNO is willing to pay for that capacity at that PoP. The bidding amount can vary from using simple proportional pricing approaches to the use of Nash games [89] based pricing strategies, depending on individual bidding strategies by MVNOs. This approach allows complete de-coupling of the pricing model from the mapping problem, and allows the mapping algorithms to focus purely on the network operators profit maximization. This mapping phase also uses interference graphs to determine which node's transmissions will result in interference. Pre-constructed interference graphs based on a two hop protocol interference model are used for determining the appropriate subset of nodes that can be scheduled on the path to the sink. For self interference of the flows, we calculate quotas in the phase 1 with the assumption that the nodes are within carrier sense region of each other, thereby allowing us to make the assumption that a higher layer MAC scheduling mechanism can be used to prevent this self interference.

The basic idea of the GSA algorithm is as described in algortihm 3. Using the pre-processed and pre-provisioned physical substrate, the nodes are sorted in a descending sequence of their $\frac{popularity}{cost-per-bit}$. Now, the algorithm selects each of these nodes and determines their characteristic descriptor. Using this descriptor as the selection criterion, we select all PoP requests (from possibly different MVNOs). Using these requests, we populate the standard weights, capacity and value parameters for initializing a knapsack. By solving a $0-1$ knapsack using dynamic

[10]It has to be noted that though we use popularity as a metric in our study, our mapping algorithms can work with any other characteristic descriptors.

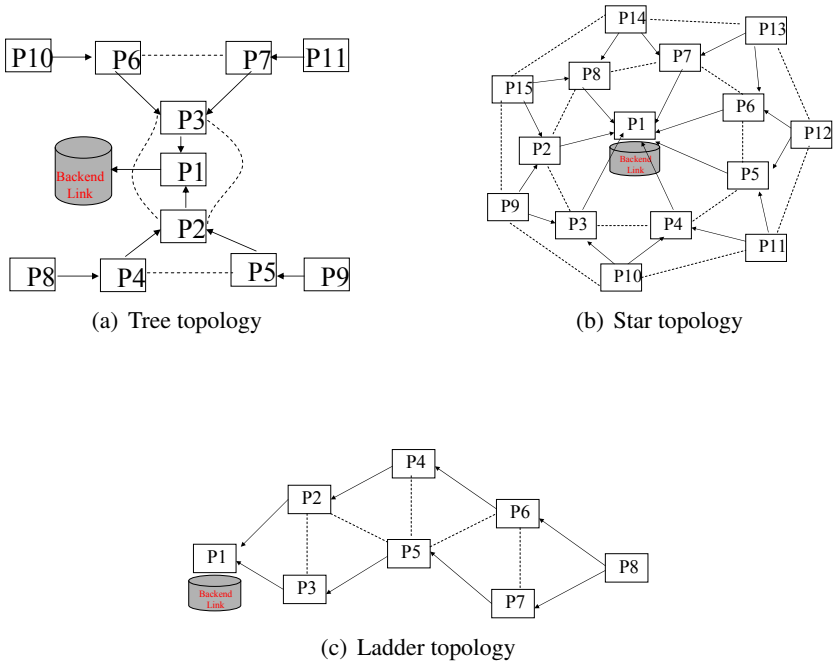
Algorithm 4: Algorithm for greedy dynamic re-allocation of the physical substrate's resources and mapping of the PoP requests from the MVNOs.

Input: $\{node, sink, req\_cap\}$
Output: $\{R_p\}$
$Det.\ \ sink\ \ path$
$P_p = getSinkPathNodes(node, sink)$;
$R_p = getRate(node, P_p, req\_cap)$; # $CS\ \ losses.$
$R_p = accountCS(node, P_p, R_p)$;
$Prev.\ \ allocations.$
$R_p = cmpAlloc(node, P_p, alloc, R_p)$;
if $req\_cap < max\_cap(node)$ **then**
  $R_p = setRate(node, P_p, req\_cap)$;
else
  $R_p = setRate(node, P_p, max\_cap)$;
$Return\ \ allocation.$
return R_p;

programming for that physical substrate node, we are able to fit the best possible combination of incoming requests, that will yield maximum profit for the network operator. Once mapping of the pre-provisioned capacity at the current physical node is completed, the algorithm moves to other nodes in the list.

6.4.4 Greedy Dynamic Re-Allocation (GDR)

The generic structure of the GDR algorithm is similar to that of the GSA algorithm. However, this approach goes one level deeper in the mapping process by dynamic re-provisioning of resources on the underlying physical nodes for revenue maximization. The pseudo-code for the GDR approach is the same as that for the GSA approach, the only difference being a condition that checks if the aggregate required capacity at the PoP is greater than that is currently provisioned. If this is the case, the GDR algorithm requests a re-provisioning request to the re-allocation module described in Algorithm 4. As described, the re-allocation module computes the maximum possible rate that can be achieved at that physical node to the sink. This value is then decremented based on the carrier sense losses and previous allocations made on that path to the sink in the previous mappings. Using this information, it allocates the lesser value of the maximum capacity at that node, and the aggregate requested capacity. This re-provisioning of resources on the physical substrate is done by setting up appropriate routing, and radio resource



(a) Tree topology

(b) Star topology

(c) Ladder topology

Figure 6.3: Topologies used in the simulation for evaluation of the mapping approach.

provisioning on all nodes in the path to the sink.

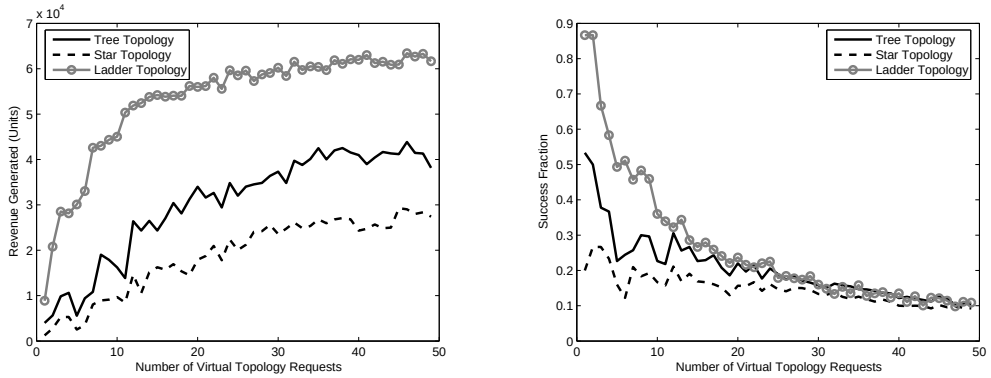
Max. Throughput Calculation: In order to calculate the maximum capacity available from any node to the sink, we calculate the effective rate from that node to the sink and scale it by the airtime required at other nodes along the path. The intuition behind this allocation is that the airtime allocated at different nodes along the path to the sink is inversely related with the physical layer rates of each node. These allocations are further scaled and reduced based on previous airtime allocations along the path to the sink.

6.5 Evaluation And Analysis

Based on the approach proposed in the previous sections, we will now perform evaluation of the algorithms by generating varying mapping requests for the framework.

6.5.1 GSA Baseline Performance

For evaluating the baseline performance of the system we will begin by comparison of the GSA algorithm on the three topologies discussed in Figures 6.3(a), 6.3(b), 6.3(c). The number of

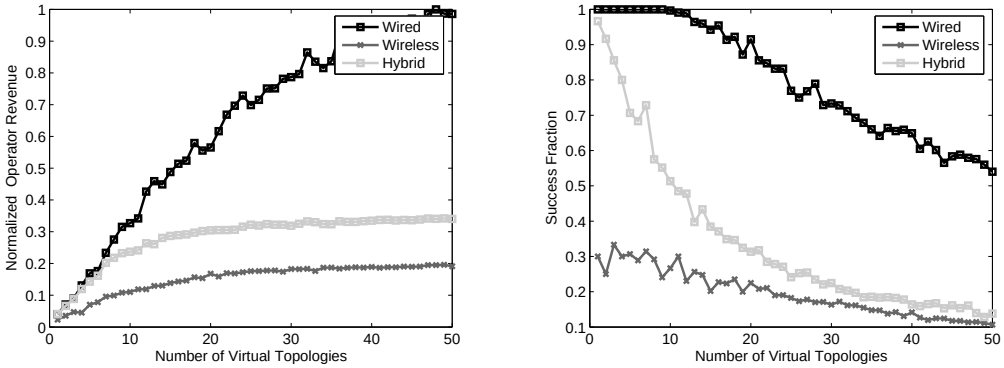


(a) Revenue Generated (b) Success Fraction

Figure 6.4: Comparison of performance for different physical substrates with the GSA algorithm.

simultaneously generated mapping requests vary from 1 to 50. Each of these mapping request will require mapping of 3 PoPs with randomly generated requirements. Specifically, each of these 3 PoP requirements will specify a randomly generated capacity, and a randomly generated popularity value (the node's characteristic descriptor), and a bid value. In our evaluation, we have chosen the bid value for every PoP as: $capacity \times popularity(1 + rand)$, where $rand$ represents a random real number between 0 and 1. The $rand$ value represents a randomly offered bonus on top of a baseline bid which is proportional to the product of the desired physical node popularity and capacity at that PoP. As discussed earlier, we have chosen a simple pricing model, since the model itself is not a critical part of the study, and can be changed to suite every requestors need. Each of the mapping runs are executed 5 times to average the mapping performance.

Figure 6.4(a) shows the amount of normalized revenue generated by the mapping algorithm for the MNO. We observe that as the number of virtual topology requests, and hence the effective number of PoPs requested increase, the amount of revenue generated for all topologies is increasing. Beyond, a certain threshold we observe that an increase in the number of virtual topologies requested does not result in an increased revenue, since the capacity of the physical network is reached. We observe that this threshold is different for different physical network topologies. Typically, the Star network topology has the least capacity due to a high CSMA cost in terms of fraction of airtime ($cost/bit$) needed to send a single bit to the network sink.



(a) Revenue Generated (b) Success Fraction

Figure 6.5: Comparison of mapping on wired, wireless, and hybrid networks for the same physical layer rates on the star topology.

Using this information as a guideline, physical network providers can appropriately limit the number of wireless PoP requests, and hence the number of virtual topologies.

Figure 6.4(b) describes the fraction of successes in mapping of the virtual topology nodes as compared with the total number of PoPs requested in the virtual topology. We observe that initially, for small topologies we have almost 100% success in the mapping. However, as the number of nodes being requested increase, we see that the mapping function declines exponentially. Typically, we observe that the fraction is least for the star topology, which has the highest average mapping cost per node in terms of the fraction of airtime required per bit sent to the sink.

6.5.2 Comparison: Wired and Wireless Meshes

In this experiment, the goal is to measure the performance of the mapping algorithm and the underlying physical network when the wireless links are changed by wired i.e. we consider the mapping problem on a wired mesh. The motivation for performing this experiment is to determine the impact of having wireless links on the physical network as opposed to wired links. Such an analysis allows the network provider to perform a cost-benefit study before making a decision to deploy the mesh as a wired/wireless network. Specifically, if the benefits are significantly higher, the network operator may decide to deploy the substrate as a wired network and vice versa.

To keep the comparison fare, we consider the same topologies for both the wired, and the wireless cases. Specifically, to highlight the impact of a large number of carrier sense regions on the virtual network mapping, we consider the Star topology discussed previously. All other physical layer characteristics such as the physical layer rate and timing constraints of the network are also kept the same in both cases. The only aspects not accounted for wired networks, which are considered by wireless networks are: (1) Interference and (2) Carrier sense regions. The goal is to see the impact of these two wireless aspects, in the presence of varying amount of PoP mapping requirements. All other experiment parameters are the same as that used for the previous experiment.

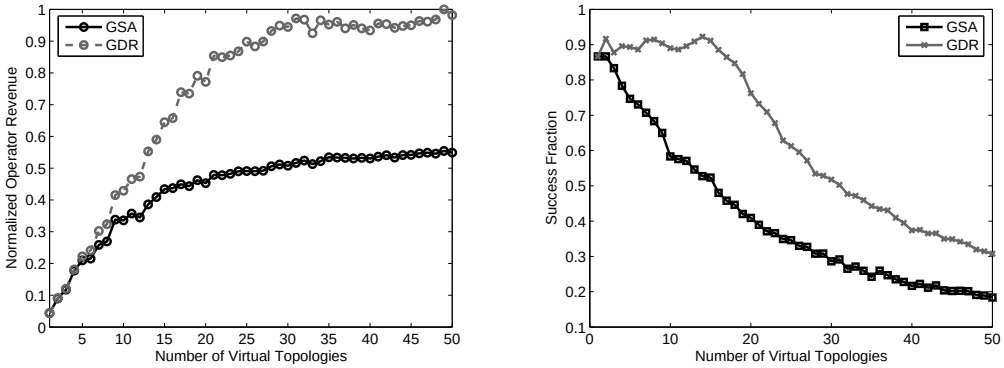
The amount of normalized revenue generated for varying number of virtual topology requests are as described in the results shown in the Figure 6.5(a). We observe that there is a large amount of difference in the revenues generated by the wired and the wireless networks. This difference in revenue is mainly due to the difference in capacities of the networks, caused due to high carrier sense cost in the wireless network, which is absent in the wired network.

The fraction of the virtual topology mapping request which are successfully mapped to the physical substrate are as shown in the Figure 6.5(b). As discussed previously, the results show a non-increasing trend. We also observe that though the fraction of request being mapped fall significantly for the wireless network, the mapping fraction remains almost constant for the wired network, indicating that the capacity is not reached yet for the same. This result corroborates with the findings from Figure 6.5(a), where the revenue from the wired network is always increasing, and does not reach a plateau indicating that the capacity of the underlying network is not reached.

In both the results earlier we note that replacing even two links which have a high CSMA cost with wired links in the hybrid network leads to a dramatic performance improvement over the wireless network performance.

6.5.3 GSA versus GDR performance

We will now compare the performances of the GSA and the GDR algorithms. Comparison is done by evaluating mapping performance on the same physical substrate and the same set of virtual topology requests. As before, we use a wireless mesh in a star topology as the substrate



(a) Revenue Generated (b) Success Fraction

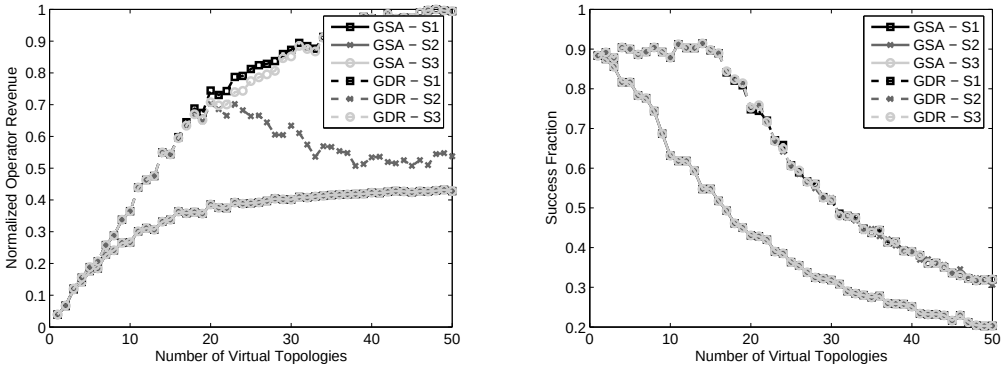
Figure 6.6: Performance comparison of the GSA and the GDR algorithms with the Star physical topology.

and the number of requested topologies are varied from 1 to 50. Results are averaged over 5 runs.

Figure 6.6(a) shows the amount of normalized revenue generated with both the algorithms as a function of the number of requested virtual topologies. We observe that both algorithms are able to generate increasing revenue with increasing number of virtual topology requests. Results show that the GDR algorithm is able to generate more revenue as compared to the GSA algorithm. We notice that up to a few number of requests, the revenue generated by both algorithms is similar. This is because, when the number of requests are less, the GDR algorithm does not have enough individual PoP requests to achieve performance better than the GSA algorithm. As the number of requests increase, the GDR algorithm is able to request re-allocation of more resources at the most profitable physical nodes.

Figure 6.6(b) shows the fraction of mapping requests that were successful for increasing number of virtual topology requests. We observe that the success fraction measurements are similar for both the GSA and the GDR algorithms. This is because the capacity of the physical network is same due to the same network setups for both algorithms. However, the GDR algorithm does slightly better because it is able to better allocate resources at the most profitable nodes with the least cost.

It is to be noted that these results are obtained with a uniform distribution for generating requests. If these requests are skewed towards a particular probability value, the benefits achieved



(a) Revenue Generated (b) Success Fraction

Figure 6.7: .

from the GDR algorithm will be much more significant.

6.5.4 PHY Node Selection Strategy

In this experiment, we wish to test the impact of the PHY node selection strategy on the overall performance achieved in the system. The results from the experiments are as shown in Figures 6.7(a), 6.7(b). As seen in the Figure 6.7(b), the success fraction is directly related to the algorithm used for the mapping process since the overall capacity of the underlying substrate is dependent upon the allocation strategy used for phase 1. The normalized revenue generated is as shown in the Figure 6.7(a). We observe that the normalized revenue with the GDR algorithm is always better than that for the GSA algorithm. However, we note that even though the average capacity using the three strategies S1 (Popularity), S2 (Cost), and S3 (Popularity/Cost) are the same, allocating quotas purely based on cost results in selection of the least profitable nodes for mapping which results in an overall inferior revenue performance. On the other hand either selection on popularity or the popularity value scaled by cost yields reasonable revenue performance as expected.

6.6 Conclusions

This part of the dissertation proposes approaches to map virtual wireless networks on to physical wireless meshes. This study shows that the motivation for virtual wireless network mapping

is different from that typically considered in the mapping of virtual wired topologies. Further, we show how these problem features can be used to simplify the mapping process itself. Results from three different physical mesh networks show that: (1) There is a significant difference in mapping between wired and wireless networks, (2) The performance in the wireless case is largely topology dependent, and (3) our heuristic using greedy re-allocation of physical resources on the mesh is able to produce the best results.

Chapter 7

Conclusions

7.1 Summary

This dissertation discusses the issue of extending virtulization to wireless networks. The following points summarize the findings of this dissertation:

1. Medium Sharing With Virtual Networks: shows two approaches to channel conservation for a wireless testbed. Evaluation of the space and time separation scheme reveal benefits and weaknesses for both. Space separation provides relatively higher efficiency, lesser coupling between experiments. We layout selection criterion for each of these schemes based on the requirement of the testbed and finally propose and implement a policy manager for controlling inter-exeperiment interference. Finally, incorporating arbitrary topologies in a slice or across VAPs allocated to the experiment may be challenging or impossible for some experiments.

2. SplitAP architecture: discusses the design of the SplitAP architecture that allows the operator to deploy a shared physical access point, which is capable of running algorithms that control UL airtime across user groups. We demonstrate the feasibility of the proposed architecture by implementing the *LPFC* and *LPFC+* algorithms on a prototype. Results obtained from the measurements on the ORBIT testbed show a significant improvement in the group airtime fairness, while resulting in marginal degradation of overall system throughput. Future directions include search for more efficient algorithms that can be deployed on the SplitAP framework.

3. Virtual Basestation Architecture: provides 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. And finally,

4. Virtual network mapping: proposes approaches to map virtual wireless networks on to physical wireless meshes. This study shows that the motivation for virtual wireless network mapping is different from that typically considered in the mapping of virtual wired topologies. Further, we show how these problem features can be used to simplify the mapping process itself. Results from three different physical mesh networks show that: (1) There is a significant difference in mapping between wired and wireless networks, (2) The performance in the wireless case is largely topology dependent, and (3) our heuristic using greedy re-allocation of physical resources on the mesh is able to produce the best results.

Chapter 8

Appendix: Platform Selection For Virtualization

8.1 Chapter Summary

A scalable approach to building large scale experimentation testbeds involves multiplexing the system resources for better utilization. Virtualization provides a convenient means of sharing testbed resources among experimenters. The degree of programmability and isolation achieved with such a setup is largely dependent on the type of technology used for virtualization. We consider OpenVZ and User Mode Linux (UML) for virtualization of the ORBIT wireless testbed and evaluate their relative merit. Our results show that OpenVZ, an operating system level virtualization mechanism significantly outperforms UML in terms of system overheads and performance isolation. We discuss both qualitative and quantitative performance features which could serve as guidelines for selection of a virtualization scheme for similar testbeds.

8.2 Introduction

Experimental validation of research ideas in a realistic environment forms an important step in identifying many practical problems. This is specially true for wireless networks since wireless communication environment is hard to accurately model through simulations. Public access testbeds like ORBIT [7, 90, 91], provide the research community with platforms to conduct experiments. ORBIT [90], typically uses a time shared experimentation model where each experimenter can reserve the grid nodes for a fixed duration (slot - approximately two hours) and has complete control of these nodes during the reservation period. An ever increasing demand for grid slots can only be met through sharing of the testbed whenever possible. Since spatial expansion is not an economically viable solution due to the limited space, prohibitive cost of setup and maintenance, we propose virtualization of ORBIT to support simultaneous

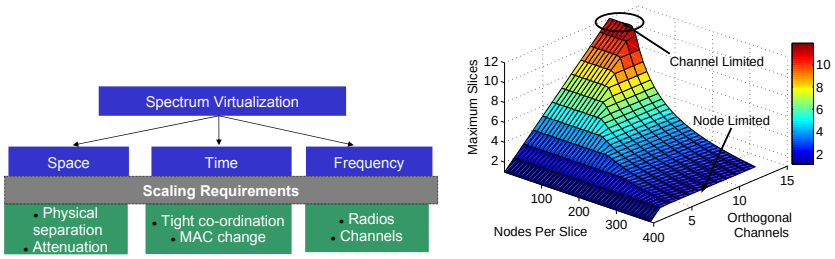
experiments. Wired testbeds like VINI [8] and Planet lab already use node and network virtualization for the same reason. In our study, we will cater specifically to requirements for sharing a *wireless* testbed through virtualization.

Another important motivation for ORBIT testbed virtualization is the integration with the GENI [16] framework. This requires ORBIT to be virtualized for allowing integration with other shared testbeds such as PlanetLab and VINI [30]. GENI also requires combining of control and management across wired and wireless networks, providing researchers with a single programming interface and experimental methodology. Since the ORBIT testbed currently supports only a single experimenter mode of operation, virtualization is essential for integration.

While wired network virtualization may be achieved by pre-allocating memory, CPU cycles and network bandwidth, to achieve perfect virtualization of the wireless network, we need to perfectly isolate both the physical devices and the wireless spectrum while providing flexibility for experimentation. This additional requirement makes the problem of wireless virtualization much harder compared to the wired counterpart [11]. Figure 8.1(a) shows different options for sharing the radio spectrum. The authors in [11] attempt to solve the spectrum sharing problem by separating experiments in time. As observed, time sharing of a single channel can result in a less repeatable performance due to context switching overheads even though it could possibly reduce the wait time for experiments. Due to the availability of a large number of radio interfaces (800 - 2/node), we share the spectrum by allocating orthogonal channels to slices. ORBIT nodes are equipped with two wireless interfaces each and therefore two virtual machines may be run on each node thereby doubling the capacity of the grid. Figure 8.1(b) shows the potential capacity of the ORBIT grid with such a frequency division (FDM) based virtualization. We observe that the number of simultaneous experiments supported on the grid are limited either by the number of orthogonal channels [11] or the number of nodes allocated per experiment.

In order to provide meaningful experimentation in the virtualized wireless testbed, the choice of the vitualization platform is critical. In this work, we start by identifying the requirements and qualitative issues to consider when selecting a virtualization platform in Section 8.3.

[11] Experiments that use other wireless technologies like zigbee and GNU radio may be run simultaneously provided they use non-interfering frequencies



(a) Spectrum sharing options. (b) FDM virtualization scaling.

Figure 8.1: Options for sharing radio resources on ORBIT and potential capacity of the ORBIT grid with 800 interfaces(2/node), 12 channels(802.11a), 2VMs/Node

After discussing the relative merits of OpenVZ for our application, we present a comparative experimental evaluation of UML and OpenVZ in Section 8.5. Related work is discussed in Section 8.6. Finally, conclusions and future directions are presented in Section 8.7.

8.3 Background and Platform Selection

Production scale virtualization systems can be broadly classified as full, para and OS virtualization. Full virtualization [59, 60](e.g.,VMWare, KVM) refers to a technique that emulates the underlying hardware and uses a software layer called hypervisor that runs directly on top of the host hardware to trap and execute privileged instructions on the fly[12]. Full virtualization is the least intrusive[13] form of system virtualization. In para virtualization [61, 62](e.g., Xen, UML) the hypervisor layer exists within the host operating system to intercept and execute privileged instructions. Unlike full virtualization, para virtualization requires changes to the guest operating system. The most intrusive form of virtualization is operating system based [63](e.g.,OpenVZ) where the virtualized systems run as isolated processes in the host operating system. The host OS is modified to provide secure isolation of the guest OS. For the purpose of this study we lay out the main qualitative criteria and select candidates for performance evaluation based on their suitability for the ORBIT testbed.

Qualitative features of a virtualization scheme which are important from a wireless testbed

[12]Native virtualization is a virtualization approach where the processor has support for virtualization e.g.,IBM System/370 and allows multiple unmodified operating systems to run together. Full virtualization does not include these systems.

[13]Intrusiveness refers to the degree of changes that need to be made to the guest OS to get it working with virtualization.

Table 8.1: Comparison of schemes from an ORBIT user perspective

Feature/Experiments	Full - Virtualization	Para - Virtualization	OS - Virtualization
Security Experiments	Yes	Yes	Yes
Network Coding	In Kernel	Overlay*	Overlay*
Mobility and Routing	Yes	Yes	Yes
Rate And Power Control	In Driver	Radiotap**	Radiotap**
Wireless Applications	Yes	Yes	Yes
Phy Measurements	Yes	Yes	Yes
MAC Parameter Control	Yes	Yes	Yes***
Transport layer Modification	In Kernel	$Emulation^{\nabla}$	$Emulation^{\nabla}$

* Transport layer experiments can be implemented as a part of overlays.

*** Radiotap headers allow for per frame rate and power control.

**** For Atheros devices MAC parameters (txop, CW, AIFS)are supported per interface.

∇ Use a click like mechanism on top of IP for custom flow or error control

administrator's perspective are as follows:

1. Ease of administration: Clean API to schedule node resources such as CPU, disk and memory on a per slice basis should be possible.

2. Shared or exclusive interface mapping: The setup should allow flexible mapping of virtual interfaces within the slice to physical interfaces or one or more virtual interfaces (on the hardware like virtual access points).

3. Control over network connectivity: Mechanisms should be available to bandwidth limit slices and control interaction between slices.

All types of virtualization schemes allow for such functions. However, in our experience the most flexible and easy approach for controlling the VMs is through operating system level virtualization such as OpenVZ. Such a setup also allows for the reuse and extension of regular system administration tools (such as IPTABLES, DHCP, SSH, LDAP) for controlling VMs.

From the perspective of an ORBIT experimenter we consider the following:

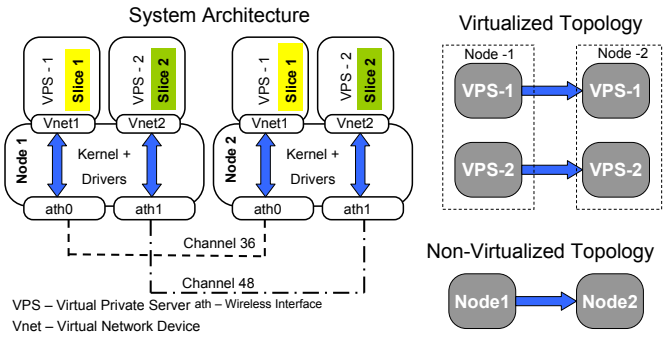


Figure 8.2: Experiment setup for OpenVZ evaluation

1. Support for standard and custom Linux distributions: Orbit nodes supports a wide variety of Linux distributions and users are free to use their own customized version. The virtualization platform running on ORBIT must support similar flexibility for the experimenter.

2. Root access within container: This feature is useful for an experimenter for providing complete freedom within the container.

Multiple Linux distributions with root access in VMs are inherently supported in all three forms of virtualization. A more detailed comparison is shown in the Table 8.2. It is observed that all wireless experiments scenarios can be either directly supported or emulated (using open source radiotap libraries and overlays) with all the virtualization setups. Traffic control elements such as Click [36] can also be run on hosts to allow for bandwidth shaping and interface mapping. Appropriate API can also be exposed from the driver to allow experimenters to have a controlled interaction with the driver. The only experiments not supported in operating-system level virtualization is the option of customizing the host kernel itself to cater to individual VMs. Despite needing emulation to support experiments that would conventionally be done by direct changes in the host kernel, the possibility of obtaining very tight slice isolation [92] make OS-level virtualization a strong candidate for evaluation.

Based on these inferences, the choice of a virtualization mechanism for ORBIT is not limited to any one type. However, full virtualization such as KVM requires specific CPU virtualization extensions (E.g. Intel VT or AMD-V) which are currently not available with our ORBIT boxes, and hence is not considered for evaluation. We consider OpenVZ (OS level) and User Mode Linux (Para - level) virtualization for quantitative comparison with testbed deployment.

Since UML based virtualization has been performed in a previous study [93], this study focusses on the performance analysis of OpenVZ. We ruled out Xen in this performance study due to incompatibility with the Via C3 processors used in the ORBIT testbed.

8.4 Experiment Setup

The Orbit testbed is a two-dimensional grid of 400 small form factor PCs with 1GHz Via C3 CPU, 512 MB RAM, 20 GB hard disk, three ethernet ports (control, data and chasis management) and two WiFi interfaces[14]. We used Atheros 5212 chipset cards with the MadWiFi(0.9.4) [34] drivers for our experiments.

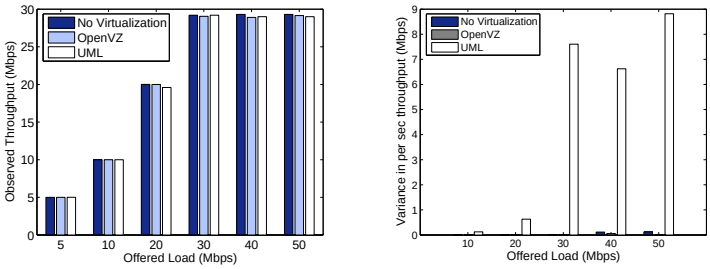
Figure 8.2 shows our experiment setup. OpenVZ uses the concept of a container also called virtual private server (VPS), an entity that performs like a stand alone server. It also provides a virtual network device named as *venetX* per VPS that acts as a point to point link between the container and the host system. We configure the *venet* devices from each of the two VPSs (on every node) to map to a corresponding WiFi card on the host. Effective virtualized and non-virtualized links are as shown in the figure. The UML virtualization setup is described in detail in [93] and is quite identical to the OpenVZ setup.

We run each experiment for 3 minutes using UML and OpenVZ setups as well as with no virtualization. The operating mode of the WiFi cards was 802.11a with bit rate of 36Mbps set at channel 36. The debian linux distribution (*Woody*) was used for both guest and host operating systems.

8.5 Performance Evaluation

We measure overheads in throughput and delay, followed by measurement of *slice isolation* achievable between slices. Quantitative evaluation presented in this section takes into account the importance of different measurement criterion. For instance, the isolation achieved between slices is far more important than sustainable peak throughput as it directly determines experiment repeatability.

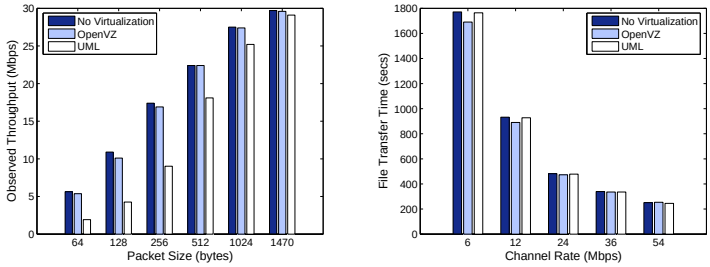
[14]It should be noted that though the results presented in the following section are hardware specific, performance trends will hold and scale with hardware capacity and load on the system.



(a) Average UDP throughput comparison.

(b) Variance in UDP bandwidth.

Figure 8.3: UDP throughput and variance in throughput as measured with different schemes. Performance is measured as a function of offered load per flow with a fixed packet size of 1024bytes. Variance in UDP bandwidth is measured over per second observed throughput at the receiver.



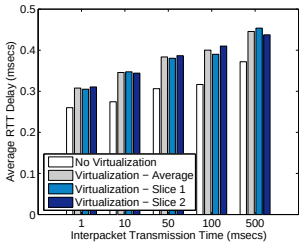
(a) Difference in UDP throughput with varying packet sizes.

(b) FTP performance with 1GB file transfer.

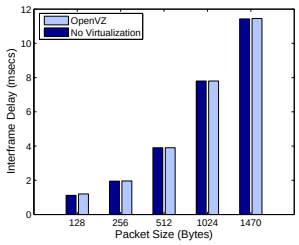
Figure 8.4: Measurement of UDP throughput with varying packet sizes and file transfer time with FTP. For the UDP throughput measurement, channel rate is constant at 36Mbps and packet size is varied. For the FTP experiment, packet size is constant at 1024 and channel rate is varied.

8.5.1 Throughput Measurements

We use the *iperf* [94] tool to generate saturation UDP traffic and average the througphut over 3 min intervals. We plot the observed UDP throughput with varying offered loads and fixed frame size of 1024bytes in Figure 8.3(a) and its variance in Figure 8.3(b). Throughput obtained in the virtualized case are averaged over the two links. We observed that both below and above channel saturation there is no distinct difference in throughput with or without virtualization. This trend indicates that both virtualization platforms perform efficiently under saturation conditions. However, the variance in throughput with UML increases with offered load specially near and above saturation. Typically, this suggests that the OpenVZ platform benefits from tighter scheduling and lower overheads compared to UML.



(a) Average RTT delay.



(b) Interframe Space Vs Packet Sizes.

Figure 8.5: Delay in different experiment scenarios. Minimum and average round trip time measurements are based on ping while interframe space measurements are based on difference in arrival times of packets at the receiver.

To determine the effect of varying packet sizes, we fix the offered load to 40Mbps and transmission rate to 36Mbps, and vary packet sizes from 128bytes - 1470bytes. Figure 8.4(a) shows that for packet sizes less than and equal to 1024 bytes, UML has a significantly higher packet processing overhead which leads to a degraded performance. We attribute this degradation in performance with UML to the lack of support for virtualization in the host kernel.

Finally, we measure throughput performance of TCP by setting up a FTP transfer of a 1GB file with varying channel rates. Resulting file transfer times are as shown in Figure 8.4(b). For all channel rates, performance of UML is on par with OpenVZ and no virtualization due to the use of larger IP frames resulting in less performance overheads.

Thus for all three cases, we observe that OpenVZ has satisfactory performance, while UML's throughput performance suffers for small frame sizes.

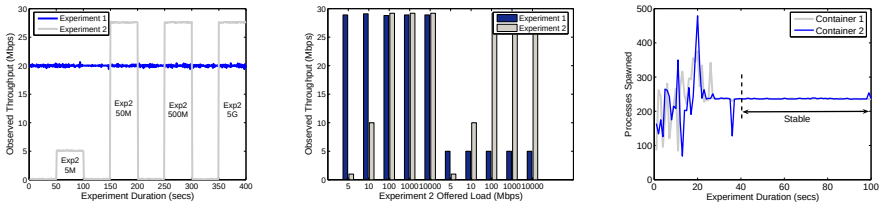
8.5.2 Transmission Delay

Delay and jitter are typically important for experiments that measure performance of real time systems or data. We measure delay and jitter performance in terms of distribution of delay across slices and distribution of delay overhead with varying packet sizes.

To measure the distribution of delay across slices we generate ICMP traffic (ping) across both slices and measure the round trip times (RTT). In Figure 8.5(a) we present the average RTT over an interval of 300 secs for varying packet arrival rates using OpenVZ. We plot delay measurements without virtualization, average delay across both slices, and delay across individual slices. The results show that in all cases, OpenVZ adds a very small average overhead (of

the order of $0.05msec$) in terms of absolute delay. The RTT delays for slices increase slightly with smaller sending rates due to slight decrease in CPU time spent on network tasks. Despite the overhead being negligible, we notice that the performance across both slices is always comparable. Efficient buffer copying mechanisms enable OpenVZ to operate with little or no delay overheads, and it is safe for making temporal measurements across slices. A separate study [93] has shown performance degradation in UML under similar experiment settings.

In order to evaluate the processing delay using OpenVZ, we measure the arrival time differences consecutive packets at the receiver with a constant sending rate. This difference in arrival times is also directly proportional to the delay [95]. We present this result as an average over $10,000$ consecutive frames of UDP traffic at 36Mbps in Figure 8.5(b). We repeat these experiments with various packet sizes. We observe that the delay increases with packet sizes due to increasing transmission times but there is little or no difference between the measurements with and without virtualization. Therefore we conclude that OpenVZ adds little overhead in packet processing and the overhead does not vary with packet size.



(a) UDP transient performance. (b) Cross coupling between experiments. (c) Process isolation test with *Fork bombs*.

Figure 8.6: Experiments for measuring the cross coupling and interference between experiments. First plot shows a performance with time, while the second plot displays results averaged over 180secs.

8.5.3 Slice Isolation

Isolation is an important requirement for a virtualized testbed since it directly determines the degree of repeatability achievable in a virtualized setting. Since OpenVZ has clearly outperformed UML in the previous experiments we will rule out UML for further experiments. To measure isolation we coin two performance measurement metrics: transient response and cross coupling between experiment.

We define transient response as the instantaneous change in throughput of an experiment

running on one slice caused due to time varying change in offered load on another slice. To measure the transient response, we maintain the offered load for the experiment running on slice 1 at a constant value of 20Mbps and vary the offered load on slice 2 from 5 Mbps to 5 Gbps in steps. Results are presented in Figure 8.6(a). We see that there is little or no correlation in the throughput of the experiment running on slice 1 (over time) in response to the change in offered load in slice 2. Therefore we may conclude that OpenVZ provides reasonable isolation between slices.

We define cross coupling as the difference in throughput with virtualization as a percentage of the throughput without virtualization. To measure cross coupling we maintain the offered load of the experiment in slice 1 at constant values of 30Mbps and vary the offered load of the experiment on slice 2 from 5 Mbps to 10Gbps in steps. This experiment is then repeated with slice 1 fixed at 5Mbps. The throughput of each experiment averaged over 180seconds are as shown in Figure 8.6(b). We see that the results of the experiments in slice 1 are never affected by the change in offered load on slice 2 and therefore we concur that there is negligible cross coupling of experiments. It is important to note that these results are achieved without tweaking features of OpenVZ that allow the user to set custom cpu usage per slice.

Finally we present test results that measure process space isolation between the VPSs. As a part of these tests, each of the containers are triggered with fork bombs, and the number of processes spawned in each of the VPSs are as shown in Figure 8.6(c). We observe that the system quickly settles to an equilibrium where each of the containers share equal number of processes. Thus we observe that OpenVZ allows for successful containment of processes within each VM.

8.6 Related Work

There are several prior works that provide comparative analysis of virtualization platforms [96–98]. However, most of this work is in the context of server/machine virtualization. Authors in [96] study the scalability of four virtual platforms: Vserver [92], UML [62], Xen [61] and VMWare [59]. They perform a quantitative evaluation by measuring virtualization overhead, and isolation between VMs. They also measure startup time and memory occupancy

of each virtualization platform. A similar study [98] has presented a comparative analysis of Xen, OpenVZ and VMWare Server using industry standard benchmarks for evaluating filesystem, network, multiprocessing and parallel processing performances. While these performance measures are important in our context as well, we concentrate more on the networking aspect of virtualization and platform suitability from a wireless testbed perspective.

The study in [93] discusses virtualization performance using UML by running two instances on a single Orbit node and isolating slices based on orthogonal channels. In our work, we extend this study by comparing the performance of OpenVZ based virtualization with the UML based scheme. Other previous wireless testbed [99] studies have more focus on the system architecture rather than features exported by the technology itself.

8.7 Conclusion and Future work

This study presents a comparison of qualitative features and performance which are useful from the perspective of a virtualized wireless testbed deployment. Our qualitative comparison shows that all forms of system virtualization could be used for virtualization of a wireless testbed. Measurements presented in the chapter show that OpenVZ consistently outperforms UML in terms of system overheads, slice isolation and its performance is closest to that of the native non-virtualized system. This performance can be attributed to a tight virtualization mechanism and efficient approach to packet handling. Having selected Open VZ as the platform for Orbit virtualization, integration with the orbit framework and measurement library are the most important next steps. From a measurement standpoint comparison with Xen and Vservers on newer Intel chipset based machines are important future research items.

References

[1] 802.1v - vlan classification by protocol and port. *http://www.ieee802.org/1/pages/802.1v.html*.

[2] Paul Ferguson, Geoff Huston, and Vpn Motivations. What is a vpn?, 1998.

[3] Joe Touch and Steve Hotz. The x-bone. In *Proc. Global Internet Mini-Conference / Globecom*, 1998.

[4] Joseph D. Touch, Yu shun Wang, Lars Eggert, and Gregory G. Finn. A virtual internet architecture 1, 2003.

[5] J.E. van der Merwe, S. Rooney, L. Leslie, and S. Crosby. The tempest-a practical framework for network programmability. *Network, IEEE*, 12(3):20 –28, 1998.

[6] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to lease the internet in your spare time. *ACM SIGCOMM Computer Communication Review*, 37:61–64, 2007.

[7] Larry Peterson, Steve Muir, Timothy Roscoe, and Aaron Klingaman. PlanetLab Architecture: An Overview. Technical Report PDN–06–031, PlanetLab Consortium, May 2006.

[8] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. In vini veritas: realistic and controlled network experimentation. In *Proceedings of SIGCOMM*, pages 3–14, New York, NY, USA, 2006. ACM.

[9] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, 2008.

[10] Yi Wang, Eric Keller, Brian Biskeborn, Jacobus van der Merwe, and Jennifer Rexford. Virtual routers on the move: live router migration as a network-management primitive. *SIGCOMM Comput. Commun. Rev.*, 38:231–242, August 2008.

[11] Gregory Smith, Anmol Chaturvedi, Arunesh Mishra, and Suman Banerjee. Wireless virtualization on commodity 802.11 hardware. In *Proceedings of Wintech*, pages 75–82, New York, NY, USA, 2007. ACM.

[12] Ranveer Chandra. Multinet: Connecting to multiple ieee 802.11 networks using a single wireless card. In *in IEEE INFOCOM, Hong Kong*, 2004.

[13] World population using mobile phones, news report. *http://www.websiteoptimization.com/bw/1002/*.

[14] Projected number of mobile users, news report. *http://news.softpedia.com/news/5-9-Billion-Mobile-Subscribers-by-2013-126085.shtml*.

[15] Wifi usage and market penetration on wikipedia. *http://en.wikipedia.org/wiki/Wi-Fi*.

[16] GENI. $http://www.geni.net/$.

[17] Dipankar Raychaudhuri et. al. NetSE: Large:Collaborative Research: COMPONENT - Converged Multi-Protocol Network Architecture for Future Mobile Internet Services. In *NSF Proposal*, Dec 2008.

[18] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to lease the internet in your spare time. *SIGCOMM Comput. Commun. Rev.*, 37(1):61–64, 2007.

[19] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4):445 – 487, 2005.

[20] Geoffrey Fox. Peer-to-peer networks. *Computing in Science and Engg.*, 3:75–77, May 2001.

[21] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 27–34, New York, NY, USA, 2003. ACM.

[22] Paolo Costa, Davide Frey, Matteo Migliavacca, and Luca Mottola. Towards lightweight information dissemination in inter-vehicular networks. In *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, VANET '06, pages 20–29, New York, NY, USA, 2006. ACM.

[23] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, Aug 2002.

[24] R. Mahindra, G.D. Bhanage, G. Hadjichristofi, I. Seskar, D. Raychaudhuri, and Y.Y. Zhang. Space versus time separation for wireless virtualization on an indoor grid. In *proceedings of NGI*, pages 215–222, April 2008.

[25] Gautam Bhanage, Dipti Vete, Ivan Seskar, and Dipankar Raychaudhuri. SplitAP: leveraging wireless network virtualization for flexible sharing of WLANs. In *IEEE Globecom 2010 - Next Generation Networking Symposium (GC10 - NGN)*, Miami, Florida, USA, 12 2010.

[26] Gautam Bhanage, Ivan Seskar, Rajesh Mahindra, and Dipankar Raychaudhuri. Virtual Basestation: architecture for an open shared wimax framework. In *ACM Sigcomm conference, VISA workshop*, New Delhi, India, 09 2010.

[27] Gautam Bhanage, Ronak Daya, Ivan Seskar, and Dipankar 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.

[28] 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.

[29] Jeff Dike. A user-mode port of the kernel. In *5th Annual Linux Showcase and Conference, Oakland, California*, Nov 2001.

[30] S. Paul and S. Seshan. Virtualization and Slicing of Wireless Networks. In *Technical Report GENI Design Document 06-17, GENI Wireless Working Group*, sep 2006.

[31] S. Kaul, M. Gruteser, and I. Seskar. Creating Wireless Multi-hop Topologies on Space-Constrained Indoor Testbeds Through Noise Injection. *Tridentcom*, 4:2731–2742, March 2006.

[32] Network time protocol daemon. $http://www.ntp.org//$.

[33] Bernard Aboba. Virtual access points, ieee document, ieee 802.11-03/154r1. *http://tinyurl.com/yjjkwpv*.

[34] Madwifi driver. $http://www.madwifi.org$.

[35] Sachin Ganu, Kishore Ramachandran, Marco Gruteser, Ivan Seskar, and Jing Deng. Methods for restoring mac layer fairness in 802.11 networks with physical layer capture. In *Proceedings of the REALMAN workshop*.

[36] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3), 2000.

[37] Kristina Knight. Jiwire: Wifi to become predominant connection for mobile users. *http://tinyurl.com/yjukaq9*.

[38] Web 2.0 framework. *http://tinyurl.com/dqt86*.

[39] Mitch Wagner. How iphone 3.0 may revolutionize the smartphone industry. *http://tinyurl.com/c4vkpa*.

[40] Rohan Murty, Jitendra Padhye, Ranveer Chandra, Alec Wolman, and Brian Zill. Designing high performance enterprise wi-fi networks. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 73–88, Berkeley, CA, USA, 2008. USENIX Association.

[41] DJ Leith, P Clifford, D Malone, and A Ng. Tcp fairness in 802.11 e wlans. *Communications Letters.*, Vol.9(11):964–966, 2005.

[42] Tarun Joshi, Anindo Mukherjee, Younghwan Yoo, and Dharma P. Agrawal. Airtime fairness for ieee 802.11 multirate networks. *IEEE Transactions on Mobile Computing*, 7(4):513–527, 2008.

[43] Chun-Ting Chou, Kang G. Shin, and Sai Shankar N. Contention-based airtime usage control in multirate ieee 802.11 wireless lans. *IEEE/ACM Trans. Networking*, 14(6):1179–1192, 2006.

[44] Albert Banchs, Pablo Serrano, and Huw Oliver. Proportional fair throughput allocation in multirate ieee 802.11e wireless lans. *Wireless Networks*, 13(5):649–662, 2007.

[45] Godfrey Tan and John Guttag. Time-based fairness improves performance in multi-rate wlans. In *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 23–23, Berkeley, CA, USA, 2004. USENIX Association.

[46] Rosario G. Garroppo, Stefano Giordano, Stefano Lucetti, and Luca Tavanti. Providing air-time usage fairness in ieee 802.11 networks with the deficit transmission time (dtt) scheduler. *Wirel. Netw.*, 13(4):481–495, 2007.

[47] Dong-Young KIM, Eun-Chan PARK, and Chong-Ho CHOI. Distributed access time control for per-station fairness in infrastructure wlans. *Transactions on Communications*, Vol.E89-B(9):2572–2579, 2006.

[48] Iperf traffic generator. *http://sourceforge.net/projects/iperf/.*

[49] Rajendra K. Jain, Dah-Ming W. Chiu, and William R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, DEC, September 1984.

[50] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862 – 876, 2010.

[51] Vanu multiran virtual basestation. $http://tinyurl.com/22lka2f$.

[52] The openbts project. $http://openbts.sourceforge.net/$.

[53] Ravi Kokku, Rajesh Mahindra, Honghai Zhang, and Sampath Rangarajan. Nvs: a virtualization substrate for wimax networks. In *Proceedings of the Mobicom*, pages 233–244, New York, NY, USA, 2010. ACM.

[54] Miguel Gómez Rodríguez, Fermín Galán Márquez, and Emilio 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.

[55] Thomas Anderson, Larry Peterson, Scott Shenker, and Jonathan Turner. Overcoming the internet impasse through virtualization. *Computer*, 38(4):34–41, 2005.

[56] VINI. `http://www.vini-veritas.net/`.

[57] Gautam Bhanage, Ivan Seskar, Yanyong Zhang, Dipankar Raychaudhuri, and Shweta Jain. Experimental evaluation of openvz from a testbed deployment perspective. In *6th international conference of Testbeds and Research Infrastructure (ICST Tridentcom)*, Berlin, Germany, 5 2010.

[58] G. Bhanage, I. Seskar, and D. Raychaudhuri. A service oriented experimentation framework for virtualized WiMAX systems. In *proceedings of 7th ICST Tridentcom conference, Shanghai, China*, 2011.

[59] VMWare player. *`http://www.vmware.com/products/player/`*.

[60] KVM. $http://www.linux-kvm.org/$.

[61] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warbeld. Xen and the Art of Virtualization. In *In Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP)*, oct 2003.

[62] A user-mode port of the kernel. *`http://user-mode-linux.sourceforge.net/`*.

[63] OpenVZ instruction manual. `http://wiki.openvz.org/`.

[64] Omf framework. $http://omf.mytestbed.net/$.

[65] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Netw.*, 1:344–357, June 1993.

[66] Paul Burke. The output of a queuing system. *Operations Research*, 4:699–704, 1956.

[67] Ieee mobile wimax standard. *http://standards.ieee.org/about/get/802/802.16.html*.

[68] Simple network management protocol. *http://www.snmp.org/*.

[69] Z. Lei and N.D. Georganas. Video transcoding gateway for wireless video access. In *Proceedings of IEEE CCECE*, volume 3, pages 1775–1778, 2003.

[70] A. Vetro, C. Christopoulos, and H. Sun. Video transcoding architectures and techniques: an overview. *Signal Processing Magazine, IEEE*, 20(2):18–29, 2003.

[71] S. Winkler and P. Mohandas. The evolution of video quality measurement: from PSNR to hybrid metrics. *Broadcasting, IEEE Transactions on*, 54(3):660–668, 2008.

[72] Hamed Soroush, Nilanjan Banerjee, Aruna Balasubramanian, Mark D. Corner, Brian Neil Levine, and Brian Lynn. DOME: A Diverse Outdoor Mobile Testbed. In *In proceedings of HotPlanet*, June 2009.

[73] E. Ertin, A. Arora, R. Ramnath, M. Nesterenko, V. Naik, S. Bapat, V. Kulathumani, M. Sridharan, H. Zhang, and H. Cao. Kansei: a testbed for sensing at scale. In *In proceedings of IPSN*, pages 399–406, 2006.

[74] Alexander Sayenko, Olli Alanen, Juha Karhula, and Timo Hämäläinen. Ensuring the qos requirements in 802.16 scheduling. In *In proceedings of ACM MSWiM*, pages 108–117, New York, NY, USA, 2006.

[75] J. Lakkakorpi, A. Sayenko, and J. Moilanen. Comparison of different scheduling algorithms for wimax base station: Deficit round-robin vs. proportional fair vs. weighted deficit round-robin. In *In proceedings of IEEE WCNC*, pages 1991–1996, April 2008.

[76] J. Sun, Yanling Yao, and Hongfei Zhu. Quality of service scheduling for 802.16 broadband wireless access systems. In *In proceedings of IEEE VTC*, volume 3, pages 1221–1225, May 2006.

[77] Yi-Neng Lin, Che-Wen Wu, Ying-Dar Lin, and Yuan-Cheng Lai. A latency and modulation aware bandwidth allocation algorithm for wimax base stations. In *In proceedings of IEEE WCNC*, pages 1408–1413, April 2008.

[78] Sheng-Tzong Cheng, Bo-Fu Chen, and Chih-Lun Chou. Fairness-based scheduling algorithm for tdd mode ieee 802.16 broadband wireless access systems. In *In proceedings of IEEE APSCC*, pages 931–936, Dec. 2008.

[79] Fen Hou, J. She, Pin-Han Ho, and Xuemin Shen. Performance analysis of weighted proportional fairness scheduling in ieee 802.16 networks. In *In proceedings of IEEE ICC*, pages 3452–3456, May 2008.

[80] Godfrey Tan and John Guttag. Time-based fairness improves performance in multi-rate WLANs. In *Proceedings of the 2004 USENIX Technical Conference*. USENIX Association, June 2004.

[81] Click modular router. $http://read.cs.ucla.edu/click/$.

[82] Robert Ricci Chris, Chris Alfeld, and Jay Lepreau. A solver for the network testbed mapping problem. *SIGCOMM Computer Communications Review*, 33:2003, 2003.

[83] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.

[84] W. Szeto, Y. Iraqi, and R. Boutaba. A multi-commodity flow based approach to virtual network resource allocation. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 6, pages 3004–3008 vol.6, Dec. 2003.

[85] Jinliang Fan and Mostafa H. Ammar. Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In *In Proc. IEEE INFOCOM*, 2006.

[86] Bernard Aboba. Virtual access points, ieee document, ieee $802.11 - 03/154r1$. *http://tinyurl.com/yjjkwpv*.

[87] Anand Rajaram and David Z. Pan. Meshworks: an efficient framework for planning, synthesis and optimization of clock mesh networks. In *ASP-DAC '08: Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, pages 250–257, Los Alamitos, CA, USA, 2008. IEEE Computer Society Press.

[88] Hideaki Takagi, Leonard Kleinrock, and Fellow Ieee. Throughput analysis for persistent csma systems. *IEEE Transactions on Communications*, 33:627–638, 1985.

[89] Nash equillibrium. $http://en.wikipedia.org/wiki/Nash\_equilibrium$.

[90] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In *WCNC*, Mar 2005.

[91] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. In *Proceedings of OSDI*, Boston, December 2002.

[92] Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, 41(3), 2007.

[93] S. Singhal, G. Hadjichristofi, I. Seskar, and D. Raychaudhuri. Evaluation of UML based wireless network virtualization. In *Proceedings of NGI*, Poland, Mar 2008.

[94] Tcp/udp traffic generation tool. `http://dast.nlanr.net/Projects/Iperf/`.

[95] Gautam Bhanage, Rajesh Mahindra, Ivan Seskar, and Dipankar Raychaudhuri. Implication of MAC frame aggregation on empirical wireless experimentation. In *IEEE Globecom 2009 Wireless Networking Symposium*, Honolulu, Hawaii, USA, Nov 2009.

[96] Benjamin Quetier, Vincent Neri, and Franck Cappello. Selecting a virtualization system for grid/p2p large scale emulation. In *Proceedings of EXPGRID workshop*, June 2006.

[97] Steffen Maier, Daniel Herrscher, and Kurt Rothermel. On node virtualization for scalable network emulation. In *Proceedings of SPECTS*, Philadelphia, PA, July 2005.

[98] V. Chaudhary, Minsuk Cha, J.P. Walters, S. Guercio, and S. Gallo. A comparison of virtualization technologies for hpc. *Proceedings of AINA*, March 2008.

[99] 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.

Curriculum Vitae

Gautam Bhanage

EDUCATION

2011 Ph.D. (Electrical & Computer Engineering), Rutgers – The State University of New Jersey, US

2006 M.S. (Electrical & Computer Engineering), Rutgers – The State University of New Jersey, US

2004 B.E. (Computer Engineering), Mumbai University, India

EMPLOYMENT/CAREER

2005–2010 Graduate Research Assistant, WINLAB, Rutgers University, NJ.

2010 Summer Intern, Raytheon BBN Labs, Boston MA.

2006 Summer Intern, SRI, Menlo Park CA.

2005 Summer Intern, Morgan Stanley, NY.

2004–2005 Graduate Teaching Assistant, Electrical & Computer Engineering Dept., Rutgers University, NJ.

SELECTED PUBLICATIONS Publications relevant to the Phd topic:

2011 G. Bhanage, I. Seskar, D. Raychaudhuri, "A Service Oriented Experimentation Framework For Virtualized WiMAX Systems", In proceedings of 7th *ICST Tridentcom conference*, Shanghai, China, 2011.

2010 G. Bhanage, D. Vete, I. Seskar, D. Raychaudhuri, Leveraging Wireless Virtualization For Flexible Sharing Of WLANs, In proceedings of IEEE Globecom next generation networks symposium, Miami, Florida, 2010.

2010 G. Bhanage, I. Seskar, R. Mahindra, and D. Raychaudhuri, "Virtual Basestation Architecture for an Open Shared Wimax Framework," In ACM SIGCOMM Workshop Virtualized Infastructure Systems and Architectures, New Delhi, India, 2010.

2010 G. Bhanage, R. Daya, I. Seskar, and D. Raychaudhuri, VNTS: a virtual network traffic shaper for air time fairness in 802:16e slices, In proceedings of IEEE ICC - Wireless and Mobile Networking Symposium, South Africa, May, 2010

2010 G. Bhanage, I. Seskar, Y. Zhang, D. Raychaudhuri, and S. Jain, "Experimental Evaluation Of OpenVZ From A Testbed Deployment Perspective," in the proceedings of 6th international conference of Testbeds and Research Infrastructure (ICST Tridentcom), Berlin, May, 2010.

2008 R. Mahindra, G. Bhanage, G. Hadjichristofi, I.Seskar, and D. Raychaudhuri, "Space Versus Time Separation For Wireless Virtualization on An Indoor Grid", in Proceedings of IEEE NGI, April 2008.

2008 R. Mahindra, G. Bhanage, G. Hadjichristofi, I.Seskar, and D.Raychaudhuri, "Integration of heterogeneous networking testbeds", in Proceedings of IEEE Tridentcom, March 2008.