NVS: A Substrate for Virtualizing Wireless Resources in Cellular Networks

Ravi Kokku, *Member, IEEE, ACM*, Rajesh Mahindra, Honghai Zhang, *Member, IEEE*, and Sampath Rangarajan, *Senior Member, IEEE*

***Abstract*—This paper describes the design and implementation of a network virtualization substrate (NVS) for effective virtualization of wireless resources in cellular networks. Virtualization fosters the realization of several interesting deployment scenarios such as customized virtual networks, virtual services, and wide-area corporate networks, with diverse performance objectives. In virtualizing a base station's uplink and downlink resources into slices, NVS meets three key requirements—isolation, customization, and efficient resource utilization—using two novel features: 1) NVS introduces a provably optimal slice scheduler that allows existence of slices with bandwidth-based and resource-based reservations simultaneously; and 2) NVS includes a generic framework for efficiently enabling customized flow scheduling within the base station on a per-slice basis. Through a prototype implementation and detailed evaluation on a WiMAX testbed, we demonstrate the efficacy of NVS. For instance, we show for both downlink and uplink directions that NVS can run different flow schedulers in different slices, run different slices simultaneously with different types of reservations, and perform slice-specific application optimizations for providing customized services.**

***Index Terms*—Cellular networks, flow scheduling, network slicing, programmability, spectrum sharing, virtualization, wireless resource management.**

I. Introduction

NETWORK virtualization enables deploying customized services and resource management solutions in isolated slices on a shared physical network. In the context of cellular networks, wireless resource virtualization can benefit several interesting deployment scenarios.

1) *Mobile virtual network operators (MVNOs):* Over the last decade, MVNOs have been increasingly establishing themselves as strong players in the mobile network market to provide enhanced services to focused customers [8]. These services include VoIP, video telephony, live streaming, etc., along with traditional voice services. MVNOs do not own the wireless network infrastructure themselves, but lease it from mobile network operators (MNOs) [8]. This model is argued to be a win–win situation for both MVNOs and MNOs since MVNOs help MNOs attract/retain greater number of customers [30], [33], [50].
2) *Corporate bundle plans:* As revenue from voice services is decreasing rapidly, data services are receiving increased focus from network operators [33]. To date, several sophisticated data plans for revenue generation have emerged and are constantly evolving [1]. Some corporate data plans allow bandwidth to be dynamically shared across employees within a corporation.
3) *Controlled evaluation of innovations:* Virtualization can help MNOs isolate partial wireless resources to deploy and test new ideas while continuing to run operational networks. This reduces the requirement for rebooting/restarting base stations or extra experimental base stations for field testing of upgrades or new solutions. More generally, a virtualized base station can help research efforts like Global Environment for Network Innovations (GENI) [5] to enable a general virtualized environment for studying innovative technologies in large-scale real-life scenarios.
4) *Services with leased networks (SLNs):* With the evolution of Internet services on which users are increasingly dependent, and the increased use of wireless networks on the last mile, in the future we envision application service providers (such as Google and Amazon) paying the wireless network operators on behalf of the users to enhance users' quality of experience.

All the above scenarios, although different on the user or the service side, impose the same requirements on network resource sharing, as we illustrate in Fig. 1. Fig. 1(a) depicts cases 1 and 2, where the requirement is to isolate network resources across user groups; application or service-specific isolation may not be present. Within the group, the network resources are shared according to the policies imposed/requested by the MVNO or the corporation. Fig. 1(b) shows complete isolation of users, networks, and services, a scenario that allows controlled experimentation and also well-isolated enterprise Intranets (a special case of corporate bundle plans). Finally, Fig. 1(c) shows network resources partitioned across services. In all the scenarios, the requirement from the network operator level is to provide virtualized network resources to the different groups.

Problem Definition and Challenges: Enabling network virtualization primarily requires addressing three requirements: 1) maintain resource isolation across groups or *slices*; 2) enable customization within individual slices; and 3) achieve efficient resource utilization. In the context of cellular networks, virtualization involves meeting the above requirements on both the core network and the access network components. While most

Manuscript received January 14, 2011; revised August 14, 2011; accepted November 08, 2011; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Walid. An earlier version of this paper was presented at the ACM International Conference on Mobile Computing and Networking (MobiCom), Chicago, IL, September 20–24, 2010.

R. Kokku is with IBM Research, Bangalore 560045, India (e-mail: ravi.kokku@acm.org).

R. Mahindra, H. Zhang, and S. Rangarajan are with NEC Laboratories America, Princeton, NJ 08540 USA (e-mail: rajesh@nec-labs.com; honghai@nec-labs.com; sampath@nec-labs.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2011.2179063

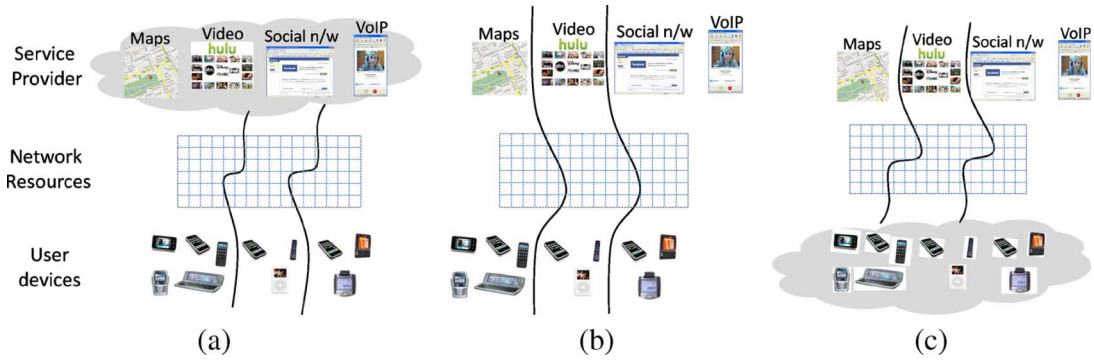


Fig. 1. Virtualization deployment scenarios. (a) MVNO/corporate: Resources are isolated across user groups, and not across services. (b) End-to-end: Resources are isolated end-to-end from services to users. (c) SLN: Resources are isolated across services, and not across users.

of these components can borrow virtualization solutions from the wired network [5], [15], [24], [28], [31], [43], [51] and desktop/server [14], [16] domains, the access network has to additionally deal with wireless resource virtualization, which is the focus of this paper.

In particular, several unique challenges get introduced in the virtualization of wireless resources.

1) Unlike wired networks, coexistence of slices with bandwidth-based and resource-based reservations is not straightforward since the bandwidth achieved by a slice from a given amount of resources varies with the channel quality of the users and the flow scheduling policies.
2) The access network has to consider resource sharing for uplink traffic; while resources are allocated to clients by the base station in the access network, traffic actually originates at the clients, thereby making it harder to achieve the two conflicting goals of isolation and efficient resource utilization across slices.
3) Wireless networks often incur considerable overheads due to signaling and retransmissions that have to be properly accounted for; ignoring such overheads affects users with better channel quality, thereby also hurting base station utilization. On the other hand, enabling customization requires striking the right tradeoffs to accommodate diverse slice requirements. While exposing too little flexibility would hinder innovation and differentiation, too much flexibility might result in unnecessary complexity for the slice owners.

Contributions: In this paper, we describe the design and implementation of a network virtualization substrate (NVS) that addresses the above challenges effectively. This paper brings forth three novel contributions. First, NVS is a simple and lightweight solution for optimal virtualization of broadband wireless resources across different slices, which is attractive to base station equipment manufacturers for rapid implementation. While we implement and evaluate NVS on a PicoChip-based [11] 802.16e WiMAX testbed, the design is general and can be easily adapted to several cellular technologies with similar characteristics (such as LTE [44] and IEEE 802.16m [6]). Second, NVS fosters innovation and differentiation among network operators by enabling customizations on base stations; otherwise, base stations from different equipment vendors have relatively few differences between them due to their adherence to standards (such as IEEE 802.16e).

Third, through NVS, we introduce a powerful testbed for researchers across the board to evaluate and deploy innovative flow management solutions on cellular base stations that are otherwise closed for experimentation. To the best of our knowledge, this is the first detailed design, implementation, and evaluation of flow-level virtualization of wireless resources on base stations, which highlights several conceptual and engineering tradeoffs in realizing the system. Our experiments demonstrate the efficacy in isolation and resource utilization and the flexibility of running different customized schedulers within slices on the same shared network equipment. We demonstrate virtualization of resources in both downlink and uplink directions.

Paper Overview: The rest of this paper is organized as follows. Section II presents a brief background of WiMAX and the design considerations for virtualization of a WiMAX base station. Section III describes the design and implementation of NVS, and Section IV provides detailed experimental evaluation. In Section V, we discuss extensions of slice admission control for better utilization of base station resources. We discuss limitations of NVS and future work in Section VI, related work in Section VII, and conclude in Section VIII.

II. Problem Formulation

In this section, we provide a brief background of the relevant features of WiMAX,[1] and then outline the requirements and design considerations for NVS.

A. WiMAX Background

Fig. 2 shows a simplified logical representation of the WiMAX network architecture [18], mainly consisting of three parts: the connectivity services network (CSN), the access services network (ASN), and the subscribers. Logically, the CSN and subscribers are owned by network service providers (NSPs), and the ASN is owned by network access providers (NAPs). NSPs and NAPs can be the same physical business entities or different entities depending on the deployment models. For instance, NSPs such as MVNOs often deploy their own CSN, but lease ASN resources from a NAP. The CSN provides functionalities such as IP connectivity, authentication, authorization, and accounting (a.k.a., AAA server functionality), and support

[1]We point the interested reader to a detailed survey [47] on WiMAX network resource management.

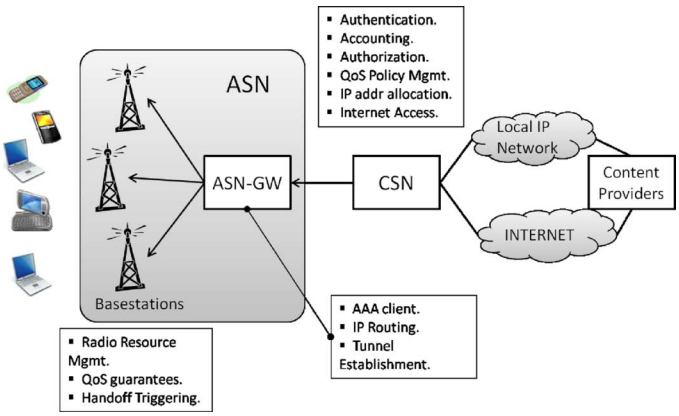


Fig. 2. Simplified WiMAX network architecture.

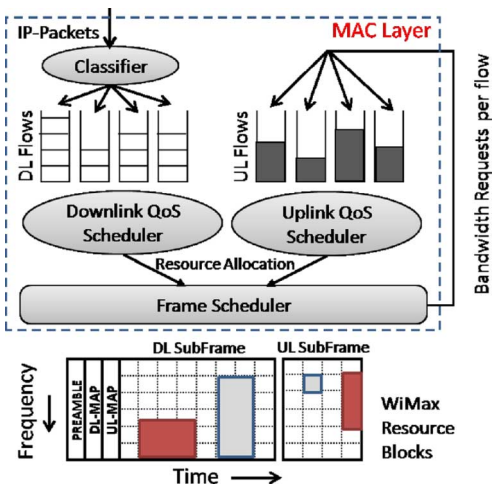


Fig. 3. Typical 802.16e MAC module.

for roaming. The ASN includes ASN gateways and base stations. The ASN gateways perform AAA client functionality, tunnel establishment with base stations, and routing to CSNs. The base stations mainly manage wireless resource allocations across users, quality-of-service (QoS) policy enforcement, and handoff triggering.

A mobile WiMAX (IEEE 802.16e) compliant base station uses orthogonal frequency division multiple access (OFDMA) frame structure for scheduling downlink and uplink transmissions between the base station and subscriber stations (clients). OFDMA enables each frame to be viewed as a set of 2-D slots of certain size on the frequency and time axes. Fig. 3 shows a sample MAC frame structure that the base station transmits periodically (typically every 5 ms). All clients are synchronized with the base station frame structure using a preamble. A downlink map informs the clients of which slots they will receive downlink data in, and an uplink map informs the clients of which slots they can transmit in. Such centralized scheduling maximizes both the client and network throughput.

WiMAX (802.16e) supports five different QoS classes for flows—UGS, eRTPS, RTPS, nRTPS, and BE—that are in the decreasing order of priority. UGS and eRTPS classes are suitable for VOIP traffic, RTPS for video traffic, nRTPS for ftp and file sharing, and BE for Web traffic. These classes differ in the resource request/grant method and QoS parameter settings. For instance, UGS enables carrying voice flows through a minimum guaranteed allocation of slots (downlink and uplink) periodically without request. RTPS allows through explicit bandwidth requests a minimum reserved rate, a maximum sustained rate, and a maximum delay on packets to carry variable bit-rate video flows. The minimum reserved rate parameter also allows admission control to avoid overcommitting resources. BE flows make explicit bandwidth requests with no minimum reservation or maximum limit on bandwidth and delay.

To help with the management of end-to-end connections, WiMAX defines the notion of a *service flow* between the base station and a user device. A service flow is a unidirectional flow (either uplink or downlink) of packets with a particular set of QoS parameters. A user's end-to-end connections are mapped to one or more of her service flows. The setup of service flows is left as a policy specification to the network operators.

For efficient resource allocation, the base station includes a collection of schedulers (Fig. 3). A *DL flow scheduler* determines the sequence of packets to be transmitted in the downlink direction based on flow priorities and other QoS parameters. Similarly, a *UL flow scheduler* determines uplink slot allocation based on the bandwidth requests from clients, channel quality, and QoS. These schedulers then invoke the *frame scheduler* that maps the packets and uplink resource allocations to specific slots in each MAC frame.

B. WiMAX Network Virtualization

To enable sharing WiMAX network resources across multiple business entities, network virtualization would require effective slicing of the CSN and the ASN resources. Fortunately, most of the CSN and the ASN components are built out of wired networks and server-grade machines, which can be virtualized with techniques drawn from related research and development efforts [5], [15], [24], [28], [31]. However, the ASN has to additionally address wireless resource sharing, a problem unique to the wireless domain and relatively less explored. Moreover, virtualization is most crucial on the scarce (bottleneck) resource in a system; in cellular networks, wireless resources are the most scarce and cannot be easily replicated like processors and memory. This paper focuses on this exact problem and builds NVS, a substrate on base stations that enables effective virtualization of the wireless resources.

The slotted OFDMA structure, the notion of service flows, and centralized uplink and downlink scheduling make WiMAX networks amenable to virtualization; NVS *builds upon these mechanisms* to consider flows in groups or slices (i.e., each slice consists of all the flows of an entity that requests virtualized resources) and meet the following requirements.

1) *Isolation:* Primarily, NVSshould ensure isolation of network resources across slices. Isolation means that *any* change in one slice due to new users, mobility of users, fluctuating channel conditions, etc., should not lead to reduction in resource allocation for other slices.
2) *Customization:* Different slices may prefer to handle flow scheduling differently depending on the services provided to achieve greater service differentiation against competitors. Hence, NVS should provide simple and appropriate

programming interfaces to enable customized solutions within the slices; specifically, QoS management across flows should be left to the slices.

3) *Resource utilization:* Especially in the wireless domain, network operators often attempt to maximize their revenue by keeping the scarce wireless channels occupied as much as possible. Hence, NVS should perform dynamic adaptation of resources across slices to maximize the benefit of both MNOs and slice owners.

C. Design Tradeoffs

The design of NVS requires us to understand the tradeoffs in the following two aspects.

1) Level of Isolation: Virtualization may be done at different levels such as at flow level, at subchannel or time-slot level, or even at the lowest level of hardware components [13] (such as antennas and signal processors). Virtualization at a higher level leads to better multiplexing of resources across slices (and hence increased utilization with fluctuating traffic) and simplicity of implementation, but can reduce the efficacy of isolation and the flexibility of resource customization, whereas virtualization at a lower level leads to the reverse effect. We resolve this tradeoff using three observations. First, the deployment scenarios we consider (SLNs, corporate bundle plans, and MVNOs) often require only flow-level isolation; many of them only care about flow-level/application-level QoS. Second, a large number of research efforts focus on developing innovative flow scheduling techniques, for which lower level virtualization is not required. Finally, exposing too much flexibility through lower level virtualization would expose unnecessary complexity to slice owners. Consequently, NVS provides flow-level virtualization to foster a broad set of deployment scenarios as mentioned above, while achieving better resource utilization.

2) Slice Provisioning: For performance predictability, supporting richer QoS and effective admission control of flows within a slice, resources need to be reserved on a per-slice basis. Such provisioning may be requested in two ways.

1) *Resource-based provisioning* defines resource allocation to a slice in terms of a fraction of the base station's resource slots per OFDMA frame. For instance, an MNO $\mathcal{A}$ may want to become an MVNO on another MNO $\mathcal{B}$'s network in geographical areas where $\mathcal{A}$ does not have coverage or own the wireless spectrum. In this case, $\mathcal{A}$ may request a slice in terms of percentage of base station resources and manage allocation across its flows much like its own network; the MNO $\mathcal{A}$ essentially gets extended coverage through these *virtual* base stations.
2) *Bandwidth-based provisioning* defines resource allocation to a slice in terms of the aggregate throughput (in megabits per second) that will be obtained by flows of that slice. For instance, an application service provider providing VoIP services, and intending to support mobility, may request a fixed amount of bandwidth to support a certain number of simultaneous calls for its mobile users, irrespective of the channel quality perceived by the users.

To understand the tradeoff in choosing between the two forms of provisioning, we consider the conceptual graph in Fig. 4. On the x-axis, we plot the resource slots allocated, which is also the cost incurred by a slice. On the y-axis, we plot the corresponding utility (bandwidth) achieved. The three lines represent different effective transmission rates (i.e., average bytes per slot) of a slice that depend on the combination of flows scheduled and their channel quality. With resource-based provisioning, the vertical dotted line in Fig. 4 shows that depending on channel conditions, fixed resource blocks result in different amounts of bandwidth, and hence different utility. Consequently, a slice must be more sophisticated to dynamically admit and schedule flows (much like traditional cellular base stations) and plan for the consequences of variable achieved bandwidth. Resource-based provisioning keeps pricing simple between the slice owner and the network operator since only a fixed amount of resources is allocated. On the other hand, bandwidth-based provisioning incurs a variable amount of base station resources depending on the clients' channel conditions, as shown by the horizontal dotted line in Fig. 4. Hence, a variable pricing model between the slice owner and the network operator may be more appropriate. However, it makes the design of flow scheduling and admission control within a slice easier since the bandwidth achieved, and hence the utility, is fixed as long as the base station is not overloaded (much like wired network flow scheduling).

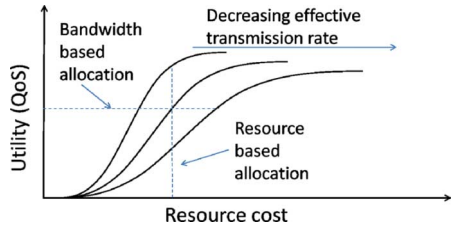


Fig. 4. Bandwidth versus resource provisioning.

Since both provisioning approaches have distinct advantages and disadvantages and satisfy different objectives, they are equally viable alternatives for slice requesters. Hence, NVS accommodates both forms of provisioning simultaneously.

III. NVS Design and Implementation

To meet the three requirements of virtualization, NVS decouples the flow scheduling problem from the slice scheduling problem. Such decoupling enables two advantages: 1) it provides greater control of customization to slices in both uplink and downlink directions and fosters the deployment of even proprietary and closed-source flow scheduling approaches; and 2) it makes slice provisioning, pricing between the slice owners and the MNO, and slice scheduling simpler. Despite the decoupling, NVS retains the notion of uplink and downlink service flows as in a traditional WiMAX base station scheduler,[2] and maintains per-flow queues. However, each flow is additionally tagged with the slice ID. Fig. 5 shows a schematic representation of NVS.

NVS operates at MAC-frame granularity to deliver packets or uplink resource allocations to the frame scheduler. For each frame, in each direction, NVS chooses the slices whose utility

[2]Observe that maintaining flow granularity allows using different uplink channel access methods for the different QoS classes, as is done in traditional WiMAX schedulers.

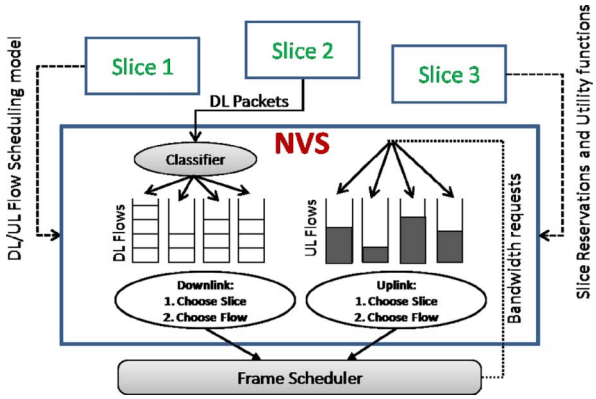


Fig. 5. NVS design.

is maximum if selected at that instant. The slice utility functions are generally agreed upon between the slice owners and the MNO such that maximizing the total utility of the slices directly maximizes the revenue of the MNO. Once a slice is chosen, NVS chooses a flow within the slice. To facilitate a variety of flow schedulers while achieving efficient resource utilization, we develop a generic framework for letting each slice direct NVS to *emulate* the flow scheduling policy chosen by the slice within the base station.

A. NVS *Slice Scheduler*

The goal of the slice scheduler (Step 1 of choosing a slice in Fig. 5) is to maximize the base station revenue while meeting the individual slice requirements. In what follows, we derive the solution by defining a specific class of utility functions that enable coexistence of bandwidth- and resource-based slice provisioning, modeling the problem in a utility-optimization framework, and then describing an optimal resource allocation algorithm.

1) Model: In either the uplink or the downlink direction, we consider two groups: Slices in group G_1 require bandwidth-based provisioning, and slices in group G_2 require resource-based provisioning. For a slice g in G_1, let $\tilde{r}_g^{\text{rsv}}$ denote the minimum reserved cumulative rate (or bandwidth) requested by a slice. Let r_g^{eff} denote the average long-term effective transmission rate of a slice g in the slots allocated to it; the slice scheduler maintains this quantity as a running average over a moving time window and updates it every MAC frame. Note that r_g^{eff} depends on the channel quality of the active users as well as the user scheduling policy in slice g. To account for the variation of r_g^{eff} and restrict its effect on other slices, we introduce the notion of a reference effective rate $\tilde{r}_g^{\text{eff}}$ for each slice, above which its reservation is honored, and below which its reservation is proportionally scaled down. In particular, we assume that an MNO offers the following *contingent bandwidth service level agreement (SLA)* to slice owners requesting bandwidth based provisioning: If the effective rate r_g^{eff} of a slice is at least $\tilde{r}_g^{\text{eff}}$, the minimum reserved rate that the slice scheduler attempts to provide—r_g^{rsv}—is set to $\tilde{r}_g^{\text{rsv}}$. Otherwise, r_g^{rsv} is scaled down proportional to r_g^{eff}. Thus

$$r_g^{\text{rsv}} = \tilde{r}_g^{\text{rsv}} \cdot \min\left(1, \frac{r_g^{\text{eff}}}{\tilde{r}_g^{\text{eff}}}\right). \tag{1}$$

The parameter $\tilde{r}_g^{\text{eff}}$ is the reference rate that can be used to instantiate variable pricing for bandwidth guarantees. For example, a particular slice owner may choose a lower value of $\tilde{r}_g^{\text{eff}}$ in an SLA and pay more to the MNO, and thereby achieve stricter bandwidth guarantees. Finally, let r_g represent the cumulative rate achieved by a slice till any instant of time, and $\mathcal{U}_g(r_g)$ be the utility (or revenue) for the slice g if a rate of r_g is achieved for the slice g.

For a slice g in G_2, let t_g^{rsv} and t_g be the minimum reserved fraction of resources (slots) and the achieved cumulative fraction of resources (slots) for the slice, and $\mathcal{V}_g(t_g)$ be the utility (or revenue) for the slice g if a fraction t_g of total resources is allocated to the slice g. Without loss of generality, we assume that the fractions of allocations of slices sum up to 1.

In general, the utility functions can be arbitrary depending on the business models between MNOs and slice owners. Since wireless resource virtualization is still at the nascent stage, no knowledge exists yet on how the utility functions will be defined. Hence, in this paper, we choose a plausible set of utility functions based on two observations. First, to enable coexistence of bandwidth- and resource-based provisioning, $\mathcal{U}_g(r_g)$ and $\mathcal{V}_g(t_g)$ should be comparable. Second, building on knowledge from the wired network domain [45], and assuming that admission control is employed to restrict both the number of slices and the number of non-best-effort flows within slices, $\mathcal{U}_g(r_g)$ and $\mathcal{V}_g(t_g)$ can be modeled as concave functions with respect to r_g and t_g, respectively. With this definition, the marginal utility of a slice decreases as the achieved rate r_g or the time allocation t_g increases. The following utility functions display such properties:

$$\begin{aligned} \mathcal{U}_g(r_g) &= \frac{r_g^{\text{rsv}}}{r_g^{\text{eff}}} \log(r_g) \\ \mathcal{V}_g(t_g) &= t_g^{\text{rsv}} \log(t_g). \end{aligned} \tag{2}$$

For $\mathcal{U}_g$, if a slice g requests higher minimum reserved rate, it should be charged more. Therefore, there is a linear factor r_g^{rsv} in the utility function. Higher effective rate r_g^{eff} indicates that the slice g needs fewer resources to achieve the reserved rate r_g^{rsv}. Therefore, $\mathcal{U}_g$ is proportional to the inverse of r_g^{eff}. The logarithm function of the achieved rate $\log(r_g)$ satisfies the concavity requirement. Similar rationale applies to the utility function $\mathcal{V}_g$. The following lemma shows that the two utility functions are equivalent, and hence comparable.

Lemma 1: Maximizing the utility functions $\mathcal{U}(r_g)$ and $\mathcal{V}(t_g)$ is equivalent if we define $t_g^{\text{rsv}} = r_g^{\text{rsv}}/r_g^{\text{eff}}$, $g \in G_1$.

Proof: As the average achieved rate r_g of a slice is its effective average transmission rate r_g^{eff} multiplied by the fraction of time t_g it is allocated

$$\begin{aligned} \mathcal{U}_g(r_g) &= \frac{r_g^{\text{rsv}}}{r_g^{\text{eff}}} \log(r_g) \\ &= t_g^{\text{rsv}} \log(r_g^{\text{eff}} t_g) \\ &= t_g^{\text{rsv}} \log(r_g^{\text{eff}}) + \mathcal{V}_g(t_g). \end{aligned} \tag{3}$$

Since the first term in (3) is fixed, maximizing $\mathcal{U}_g(r_g)$ and $\mathcal{V}_g(t_g)$ is equivalent. ■

Now, the objective of the slice scheduler is to

$$\begin{aligned} \text{maximize} \quad & \sum_{g\in G_1} \mathcal{U}_g(r_g) + \sum_{g\in G_2} \mathcal{V}_g(t_g) \\ \text{subject to} \quad & r_g \geq r_g^{\text{rsv}} \text{ if } g \in G_1 \\ & t_g \geq t_g^{\text{rsv}} \text{ if } g \in G_2 \\ & \sum_{g\in G_1} \frac{r_g}{r_g^{\text{eff}}} + \sum_{g\in G_2} t_g \leq 1 \end{aligned} \tag{4}$$

where the last constraint is the total resource constraint (r_g/r_g^{eff} is essentially the fraction of resources/time allocated to slice g in G_1).

2) Admission Control: To strictly satisfy the SLA for both types of slices, we should ensure that (4) is feasible. The following lemma shows a sufficient condition for the feasibility of (4).

Lemma 2: Problem (4) is feasible if

$$\sum_{g\in G_1} \frac{\tilde{r}_g^{\text{rsv}}}{\tilde{r}_g^{\text{eff}}} + \sum_{g\in G_2} t_g^{\text{rsv}} \leq 1. \tag{5}$$

Proof: Assuming condition (5) is satisfied, we construct a feasible solution as follows. Let $r_g = r_g^{\text{rsv}}$ for $g \in G_1$ and $t_g = t_g^{\text{rsv}}$ for $g \in G_2$. We have

$$\begin{aligned} \sum_{g\in G_1} \frac{r_g}{r_g^{\text{eff}}} + \sum_{g\in G_2} t_g &= \sum_{g\in G_1} \frac{r_g^{\text{rsv}}}{r_g^{\text{eff}}} + \sum_{g\in G_2} t_g^{\text{rsv}} \\ &\leq \sum_{g\in G_1} \frac{\tilde{r}_g^{\text{rsv}}}{\tilde{r}_g^{\text{eff}}} + \sum_{g\in G_2} t_g^{\text{rsv}} \\ &\leq 1 \end{aligned}$$

where the first inequality follows from (1). Therefore, all constraints in (4) are satisfied. ■

The admission control policy for strict isolation across slices would ensure that (5) is always satisfied. We comment that when $r_g^{\text{eff}} \leq \tilde{r}_g^{\text{eff}}$, the condition in (5) is also necessary and thus tight. When $r_g^{\text{eff}} > \tilde{r}_g^{\text{eff}}$, the condition is not tight. In this case, we can apply the technique in Section V to avoid wasting radio resources caused by admission control.

3) Slice Scheduling Algorithm: We now develop the following simple, online solution to solve (4). At each time instant j, define an exponential weighted moving average rate $r_{g,j}^{\text{exp}}$ and time allocation $t_{g,j}^{\text{exp}}$ for slices in G_1 and G_2, respectively

$$\begin{aligned} r_{g,j}^{\text{exp}} &= (1-\beta) r_{g,j-1}^{\text{exp}} + \beta r_{g,j}^{\text{inst}} I(s(j) == g) \\ t_{g,j}^{\text{exp}} &= (1-\beta) t_{g,j-1}^{\text{exp}} + \beta I(s(j) == g) \end{aligned} \tag{6}$$

where β is a small positive constant, $r_{g,j}^{\text{inst}}$ is the instantaneous transmission rate at slice g, $s(j)$ is the slice that is scheduled in time j, and I is an indicator function. The moving averages also account for the signaling overheads (in the appropriate units) incurred by flows of each slice for downlink and uplink maps, uplink bandwidth requests, and channel feedback. Define $w_{g,j}$ at time instant j as

$$w_{g,j} = \begin{cases} \frac{r_g^{\text{rsv}}}{r_{g,j}^{\text{exp}}}, & \text{if } g \in G_1 \\ \frac{t_g^{\text{rsv}}}{t_{g,j}^{\text{exp}}}, & \text{if } g \in G_2. \end{cases} \tag{7}$$

The scheduling algorithm then *schedules the slice with the maximum weight $w_{g,j}$ at each time instant j*. Intuitively, the weight represents the marginal utility of each slice with respect to t_g; this can be easily shown by considering the derivatives of $\mathcal{U}_g(r_g)$ and $\mathcal{V}_g(t_g)$ with respect to t_g, and using the chain rule for the former.

This solution is simple to implement, and solves (4) optimally, as we show in the following theorem.

Theorem 1: The above scheduling algorithm (i.e., scheduling the slice with the highest $w_{g,j}$) converges to the optimal solution for (4) if the problem is feasible.

Proof: When the system converges, the exponential moving average converges to the long-term average, implying $t_{g,j}^{\text{exp}} \to t_g$ and $r_{g,j}^{\text{exp}} \to r_g$. Moreover, upon convergence, the following condition is satisfied:

$$\frac{r_g^{\text{rsv}}}{r_g} = \frac{t_{g'}^{\text{rsv}}}{t_{g'}} = \text{constant for all } g \in G_1, g' \in G_2. \tag{8}$$

In other words, the weights $w_{g,j}$ of all slices converge to the same value. This is because if a slice has a larger weight than other slices, it will be allocated more bandwidth, which increases its moving-average data rate $r_{g,j}^{\text{exp}}$ or time allocation $t_{g,j}^{\text{exp}}$, and thus decreases its weight $w_{g,j}$. Similarly, if a slice has a smaller weight than other slices, it will not be allocated bandwidth, which decreases its moving-average data rate $r_{g,j}^{\text{exp}}$ or time allocation $t_{g,j}^{\text{exp}}$, and thus increases its weight $w_{g,j}$. As a result, all slices converge to the same weight $w_{g,j}$.

Using Lemma 1, (4) can be simplified as

$$\begin{aligned} \text{maximize} \quad & \sum_{g\in G} t_g^{\text{rsv}} \log(t_g) \\ \text{subject to} \quad & t_g \geq t_g^{\text{rsv}} \qquad \text{for all } g \in G \\ & \sum_{g\in G} t_g \leq 1 \end{aligned} \tag{9}$$

where $G = G_1 \cup G_2$. Clearly, the problem is feasible if and only if $\sum_{g\in G} t_g^{\text{rsv}} \leq 1$. Therefore, if the problem is feasible, we can write $\tau_g = t_g - t_g^{\text{rsv}}$, and (9) can be transformed to

$$\begin{aligned} \text{maximize} \quad & \sum_{g\in G} t_g^{\text{rsv}} \log(\tau_g + t_g^{\text{rsv}}) \\ \text{subject to} \quad & \tau_g \geq 0 \qquad \text{for all } g \in G \\ & \sum_{g\in G} \tau_g \leq 1 - \sum_g t_g^{\text{rsv}}. \end{aligned} \tag{10}$$

To solve problem (10), we can apply [21, Proposition 2.1.2] to obtain the following necessary and sufficient condition for

the optimal solution $(\tau_g, g = 1, \ldots, G)$ due to the concavity of the objective function:

$$\tau_g > 0 \Rightarrow \frac{\partial f}{\partial \tau_g} \geq \frac{\partial f}{\partial \tau_{g'}} \qquad \text{for any } g' \tag{11}$$

where $f = \sum_{g \in G} t_g^{\text{rsv}} \log(\tau_g + t_g^{\text{rsv}})$ is the objective function in (10). Hence, (11) is equivalent to

$$\tau_g > 0 \Rightarrow \frac{t_g^{\text{rsv}}}{t_g^{\text{rsv}} + \tau_g} \geq \frac{t_{g'}^{\text{rsv}}}{t_{g'}^{\text{rsv}} + \tau_{g'}} \qquad \text{for any } g'. \tag{12}$$

We next prove by contradiction that

$$\frac{t_g^{\text{rsv}}}{t_g^{\text{rsv}} + \tau_g} = \text{constant for all } g. \tag{13}$$

Supposing (13) is not satisfied, assume that

$$\frac{t_g^{\text{rsv}}}{t_g^{\text{rsv}} + \tau_g} < \frac{t_{g'}^{\text{rsv}}}{t_{g'}^{\text{rsv}} + \tau_{g'}} \tag{14}$$

which implies $\tau_g = 0$ from (12). However, if $\tau_g = 0$

$$\frac{t_g^{\text{rsv}}}{t_g^{\text{rsv}} + \tau_g} = 1 \geq \frac{t_{g'}^{\text{rsv}}}{t_{g'}^{\text{rsv}} + \tau_{g'}}$$

which contradicts (14).

Finally, notice that (8) is equivalent to (13), which is a necessary and sufficient condition for the optimal solution. Therefore, the algorithm defined by scheduling slices with the maximum $w_{g,j}$ converges to the optimal solution to (4). ■

B. Flow Scheduling Framework

To ensure that each slice can employ custom flow scheduling policies, NVS lets each slice determine the order in which packets are to be sent in the downlink direction, and resource slots are allocated in the uplink direction. This can be done in several modes, all of which have distinct advantages and disadvantages.

1) *Scheduler selection:* In this mode, NVS provides several generic schedulers that a slice can choose from. This mode is attractive to MNO's customers such as MVNOs with no expertise in the wireless domain and corporate Intranets and SLNs, who only want to reserve wireless resources and prefer relying on the MNO for scheduling. This mode, however, is not suitable for evaluating innovative technologies.
2) *Model specification:* In this mode, NVS provides an interface to specify, on a per-flow basis, a weight distribution as a function of the average rate already achieved and the modulation and coding scheme (MCS). This mode is appropriate for scenarios in which the flow scheduling policy can be represented as a series of weight computations. The weight distribution is sent as a set of discrete tuples that are stored in a table in the base station, which NVS looks up during flow scheduling. Once a slice is chosen, NVS emulates flow scheduling by choosing the flow(s) of the slice in the decreasing order of the weights. If multiple flows have the same maximum weight, NVS chooses the flow with lower average rate achieved. This mode is lightweight since no continuous communication is necessary between a slice and NVS.
3) *Virtual Time Tagging:* In this mode, NVS provides flow-level feedback to a slice to perform flow scheduling itself. NVS tags each packet that arrives into the per-flow queues maintained within NVS with a monotonically increasing virtual time. NVS then uses this time as a measure of weight to select packets from the heads of flow queues of the slice. This mode enables arbitrary flow schedulers to be defined by the slice owners and can be realized using several alternative implementations: 1) the slice resides in a virtual machine external to the base station either on the ASN gateway, a node within the ASN but external to the ASN gateway, or in the CSN; 2) the slice may be implemented within the base station as a virtual machine. In the former alternative, customized uplink flow scheduling cannot be performed efficiently since per-MAC-frame resource requests from clients need to be sent to the slices. The latter alternative can support both uplink and downlink flow scheduling, but requires significantly more modifications to the base station hardware and software.

```
ALGO flow_sched
1. compute_wts ()
2. while space avail in MAC frame
3.    select pkt. from flows i
        with maximum w_i

ALGO model_compute_wts
1.  w_i = Table [i] [avgRate] [MCS]

ALGO vtag_compute_wts
1. w_i = - (head_queue_tag(i))
```

Fig. 6. NVS flow scheduling emulation.

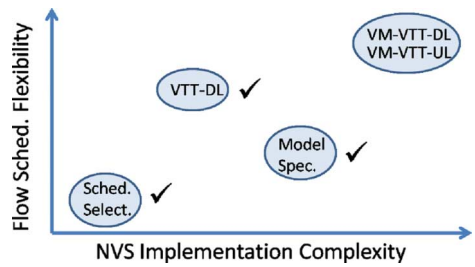


Fig. 7. NVS flow scheduling taxonomy.

Fig. 6 shows a summary of the model specification and virtual time tagging modes, and Fig. 7 compares the relative implementation complexity and flow scheduling flexibility of the various emulation modes, including variants of virtual time tagging: VTT-DL (downlink) outside the base station, and VM-VTT-DL (downlink) and VM-VTT-UL (uplink) within the base station as a virtual machine. Tick marks represent the modes we implemented in our prototype.

C. Flow Specification and Admission Control

In traditional WiMAX schedulers, service flow setup for users is left to the network operators. In a similar way, NVS encompasses a slice manager that enables slice owners to set up flows; additionally, the slice manager tags each flow with the slice ID during setup. NVS assumes that the slice owners implement flow-level admission control to ensure that flows with critical resource requirement do not suffer, and perform graceful degradation in the event of overload. Leaving flow

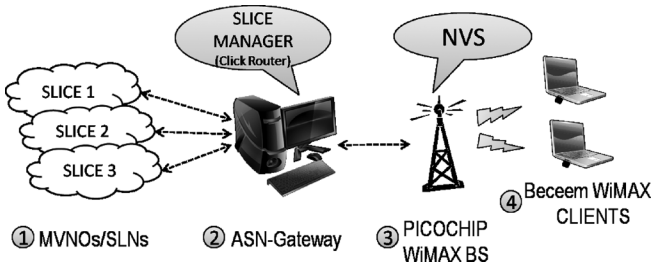


Fig. 8. WiMAX prototype platform.

admission control to slices will also ensure that researchers can evaluate new admission control policies.

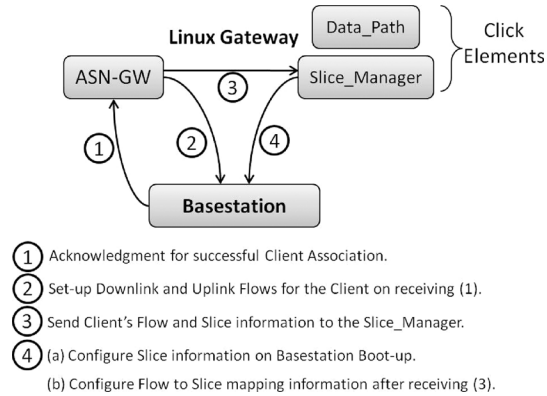


Fig. 9. Implementation details.

D. Prototype

We instantiate NVS on a WiMAX network platform consisting of a PicoChip [11] WIMAX base station (IEEE 802.16e compliant), a WiMAX Profile-C ASN gateway, and several Beceem [2] PCMCIA and USB clients (see Fig. 8). NVS is implemented as a part of the MAC task on the PicoChip base station, which includes about 500 lines of extra C code along with a few modifications to the existing MAC code. The base station contains an ARM 200-MHz processor and 64 MB memory, on which the MAC task executes. The MAC task is triggered every 5 ms with an interrupt from the PHY layer to prepare the next frame to transmit. NVS is invoked by the MAC task for scheduling the packets to be sent to the frame scheduler. The frame scheduler also executes as a part of the MAC task, after NVS. The frame scheduler sends the filled frame to the physical layer to be transmitted. The lightweight design of NVS enables us to achieve the above functionality within the 5 ms. If the MAC task does not return within 5 ms, the base station and clients would lose synchronization and disconnect. Note that the overhead for slice selection only involves computing the weight in (7) on a per-frame basis, which scales with the number of active slices, and hence is minimal. We update $r_{g,j}^{\exp}$ and $t_{g,j}^{\exp}$ once per MAC frame with $\beta = 0.1$ [(6)].

We implement the Slice Manager functionality on the PicoChip ASN gateway. The Picochip ASN gateway executes on a commodity PC running Ubuntu Linux. The original ASN gateway implementation handles control and data path between the base station and the outside network. The slice manager is implemented as an element in the Click Router Framework [3] in the Linux gateway itself. As shown in Fig. 9, we implement the necessary interfaces between the Slice Manager, the ASN gateway, and the base station. We also move the data path functionality of the ASN gateway to another element in click (i.e., *Data_Path*). The slice manager in our current implementation takes slice information as a configuration file that includes the type of provisioning and the reservation per slice in terms of either bandwidth or percentage of slots. The configuration file also defines the flow scheduler mode used. On base station bootup, the slice manager configures the above-mentioned per-slice information in the base station. The ASN gateway provides an interface to the base station for setting up service flows in the downlink and the uplink direction for each client. The service flow information is then passed to the Slice Manager that utilizes it to configure flow-to-slice mapping in the base station. We implement the three flow scheduler specification modes: scheduler selection, model specification, and virtual time tagging for downlink.

For model specification, we define a simple API (to the slice manager) for a slice requester to construct the table with a sequence of messages from each slice to NVS, containing [FlowID, MCS, avgRate, weight]. For each flow, the table is stored in NVS as a sorted array to facilitate binary search, given a current achieved avgRate. The closest avgRate match in the table that is less than the achieved avgRate is chosen for setting the weight for the flow. If flow ID is sent as zero, we use the table for all flows of the slice.

For the virtual tagging approach, we implement the individual slice functionality within the *Data_Path* Element in the Click Router Framework itself assuming that the ASN gateway has enough resources to support the slices without becoming the bottleneck, and hence does not require isolation. Ideally, each slice should be implemented within virtual machines either on the gateway or on a node behind the gateway. Such a framework is already implemented in the context of GENI [22] and can be easily integrated with NVS in the future. Note that both scheduler selection and model specification approaches do not need explicit creation of such slices.

IV. Experimental Evaluation

We now present several groups of experiments that rigorously evaluate NVS in meeting the requirements laid out—isolation, customizations and efficacy in resource utilization, and its support for different forms of provisioning. Table I summarizes the experiments. We perform all the measurements in an indoor office testbed, primarily because the PicoChip [11] base station we have access to is designed for providing Femto-cell coverage. Nevertheless, the testbed is sufficient to highlight the virtualization features provided by NVS, which will remain equally valid on Macro-cell base stations. All experiments are done with over-the-air transmissions between the base station and the clients on the 2.585–2.595-GHz channel (10 MHz). To emulate clients at different channel qualities (i.e., different MCSs), we place client laptops in different cubicle locations in the office.

TABLE I
SUMMARY OF EXPERIMENTS

Expt	Demonstrates
1	Isolation of slice throughput
2	Short timescale isolation
3,4,5	Downlink/uplink flow scheduling customization
6	Service customization within slices
7	Efficacy of flow scheduler specification modes
8	Uplink resource utilization
9,10	Co-existence of various forms of slice provisioning

The clients use Beceem chipset-based [2] USB and PCMCIA WiMAX interfaces.

The experiments from Sections IV-A to IV-C are done with bandwidth-based slice provisioning such that (5) is satisfied. The results will be similar for resource-based provisioning since bandwidth-based and resource-based provisioning are equivalent when the MCS of the clients is the same. Furthermore, unless specified, the experiments are done with the scheduler selection approach to minimize the number of varying parameters when demonstrating the behavior of NVS. We later compare the performance tradeoffs of model specification and virtual time tagging.

A. Isolation

We first demonstrate NVS's efficacy in providing slice isolation. We also provide a comparison to results obtained with similar experiments on a *Vanilla* base station that does not perform virtualization.

Experiment 1: For this experiment, we consider two groups—the first group with three clients (Slice 1), and the second with two clients (Slice 2). The clients are placed at similar distances from the base station with an MCS of QPSK-1/2. Both the slices reserve 1.7 Mb/s as aggregate bandwidth requirement. When all the clients are at QPSK-1/2, the total downlink base station capacity is 3.4 Mb/s. Each client sets up a downlink nRTPS (non-real time) flow with a minimum reservation of 0.5 Mb/s. All flows are backlogged with UDP packets generated by Iperf [7]. With such a setup, Fig. 10(a) shows the individual flow throughput with NVS. NVS ensures that each of the two slices meets the reservation of 1.7 Mb/s. Since Slice 2 has only two flows, the residual bandwidth after meeting the flows' minimum reservations is redistributed within the slice. On a Vanilla base station [see Fig. 10(b)], the scheduler gives equal bandwidth to all the flows. This is because the Vanilla base station does not have any notion of slices and uses a proportional fair flow scheduler that allocates equal resources across all flows since the channel quality is similar for all the clients, thereby violating group requirements; group 2 receives only 1.36 Mb/s. When we stop traffic for one of the flows of group 2, Fig. 10(c) shows that NVS redistributes additional resources within Slice 2, whereas the Vanilla base station scheduler [Fig. 10(d)] redistributes the additional resources to all other flows equally, thereby bringing down group 2's aggregate throughput to 850 kb/s.

Experiment 2: We now study the short-timescale isolation provided by NVS. This experiment shows that the delay experienced by interactive traffic of a slice is low as long as its *offered load* is within the reserved rate of the slice. For this experiment,

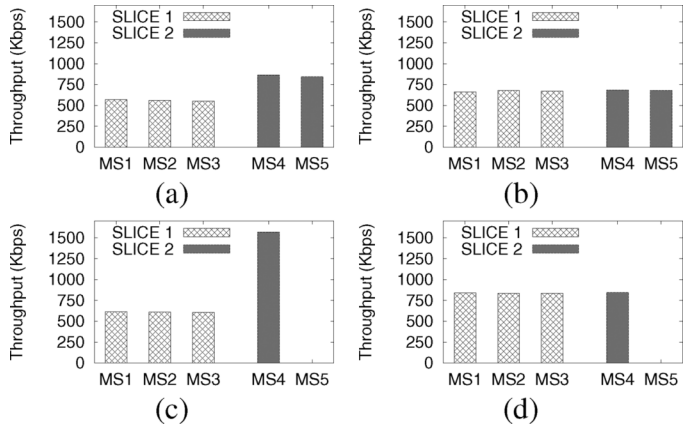


Fig. 10. Isolation across flow groups. (a), (c) NVS. (b), (d) Vanilla.

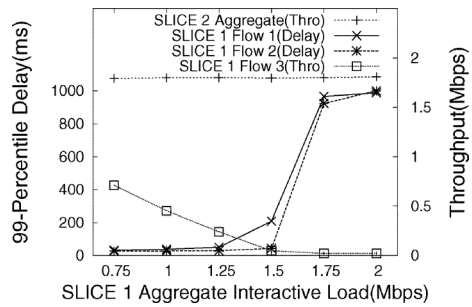


Fig. 11. Short timescale isolation.

Slice 1 has two time-sensitive flows and one ftp connection with an aggregate reservation of 1.5 Mb/s, while Slice 2 has two ftp connections with an aggregate reservation of 1.9 Mb/s. The time-sensitive flows are configured as RTPS QoS class on the base station, while the ftp connections use the nRTPS QoS class. We use D-ITG [20] to generate UDP traffic with different interpacket spacing (using exponential distribution) for the two time-sensitive flows. The ftp flows run backlogged UDP traffic. We measure one-way delay at the receiver that is synchronized with the transmitter using NTP [10] on a wired connection.

Fig. 11 shows that the 99th-percentile delay for the interactive flows is low as long as the total slice load is less than 1.5 Mb/s, which is the reserved rate for Slice 1. This is true even when the flows of Slice 2 are constantly backlogged. Observe that the throughput of the ftp connection of Slice 1 falls to negligible values to give way to the higher-priority interactive flows. Hence, isolation is guaranteed across slices. However, with offered load above 1.5 Mb/s within Slice 1, the delay for the interactive flows increases significantly due to higher queuing delays. This intraslice problem would be solved by a properly designed flow scheduling or admission control policy by the slice owner.

B. Customization

In this section, we first demonstrate that NVS allows slices to run custom flow schedulers simultaneously. Then, we experiment with an end-to-end setup, in which we enable application-level enhancements within one slice to demonstrate NVS's ability to foster the creation of enhanced services by

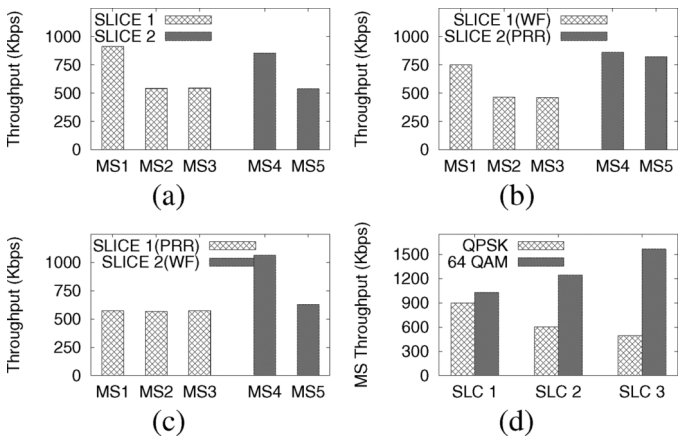


Fig. 12. Customized downlink schedulers. (a) Vanilla (WF). (b) NVS (WF/PRR). (c) NVS (PRR/WF). (d) NVS (GWF).

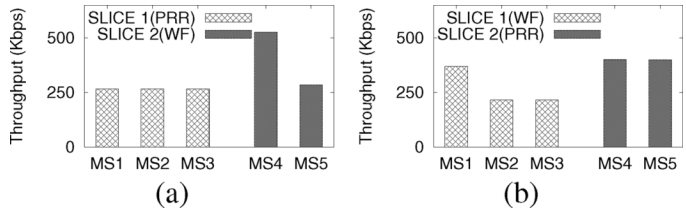


Fig. 13. Customized uplink schedulers.

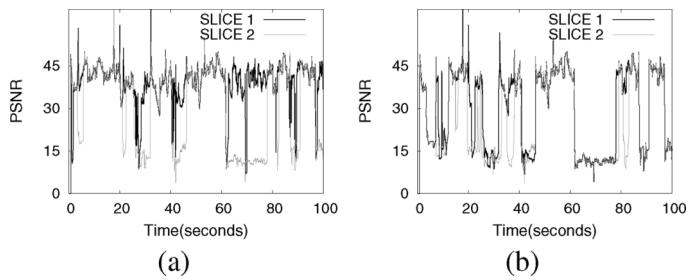


Fig. 14. Customized services within slices. (a) NVS. (b) Vanilla.

slice owners. Finally, we compare the different flow scheduling emulation approaches in the NVS framework.

To demonstrate the flow scheduler customization, we conduct experiments for uplink and downlink directions with two slices running different kinds of flow schedulers. For brevity, we demonstrate the results of three flow schedulers: a weighted-fair scheduler (WF) that allocates bandwidth to flows in proportion to their minimum reserved rates, a Priority with Round Robin (PRR) scheduler that allocates resources in the priority order of QoS classes and uses RR policy for flows within the same class, and finally a Generalized Weighted Fairness (GWF) scheduler [48].

Experiment 3: We consider three nRTPS flows for the first slice and two nRTPS flows for the second, and the minimum reserved rates of the first flow of each of the two slices is set at 1 Mb/s, while that of the remaining flows is kept at 0.5 Mb/s. For this experiment, all clients use the same MCS. For the downlink case, the aggregate bandwidth reservation of the two slices is equal to 1.7 Mb/s. Fig. 12(a) shows the Vanilla base station's allocation to flows with a WF scheduler; the first and fourth flows get double the throughput of the other flows. With NVS, Fig. 12(b) shows that Slice 1 runs a WF flow scheduler while Slice 2 runs a PRR scheduler, and Fig. 12(c) shows that Slice 1 runs a PRR flow scheduler while Slice 2 runs a WF scheduler. The graphs demonstrate that NVS allows running different flow schedulers in different slices on the same base station, whereas a Vanilla base station can run only a single flow scheduler at a time across all flows.

Experiment 4: We now instantiate three versions of the GWF [48] flow scheduler in three slices. Each slice has two nRTPS flows at MCS of QPSK-1/2 and 64QAM-1/2, respectively, with backlogged traffic. The GWF scheduler allocates resources to flows in proportion to a weighted sum of slots S_i and bytes B_i already allocated: $f = w*S_i+(1-w)*(B_i/M)$, where M is the number of bytes per slot for the maximum MCS. We set $w = 0$, $w = 0.5$, and $w = 1$ for Slices 1, 2, and 3, respectively. Fig. 12(d) shows that each slice allocates resources differently across its flows. Furthermore, observe that in Slice 1, the bandwidth is almost equal across flows since $w = 0$ provides bandwidth fairness, whereas $w = 1$ in Slice 3 provides slot fairness resulting in a bandwidth allocation proportional to the MCS of flows.

Experiment 5: In the uplink direction, we set the aggregate bandwidth reservation of the two slices to 750 kb/s. Fig. 13 shows that we make similar observations as in Experiment 3.

Experiment 6: We now demonstrate that NVS enables customized application optimizations within slices. We consider a setup with two slices with one RTPS flow each, where Slice 1 additionally incorporates extensions for enhanced video delivery to users of the slice, while Slice 2 does not perform any such extensions. Each RTPS flow carries a video in the downlink direction. This setup may be considered to emulate an MVNO or an SLN (in Slice 1) providing such enhanced service to all videos delivered to its users. The flow scheduling extensions provided by Slice 1 assume that video packets are prioritized based on the content they carry (for example, packets of H.264/AVC coded videos may be tagged as I, P, or B frames in the decreasing order of priority). The extensions include dropping enough low-priority frames when the available channel capacity fluctuates below the video requirement, so that the resulting video gets distorted minimally.

The displayed video quality is measured in terms of PSNR—a standard metric [4] that is a function of the mean square error between the transmitted video and that displayed at the receiver. Without the extensions, the video gets backlogged when the channel capacity is not sufficient and leads to a significant number of undisplayed frames at the client, thereby reducing the PSNR of the video. Fig. 14(a) shows this exact behavior. While the flow in Slice 2 observes significant drop in PSNR for long periods of time, the flow in Slice 1 observes fewer such instances. We also perform the same experiment on a Vanilla base station that does not provide any video extensions, and show in Fig. 14(b) that both flows suffer significant drop in PSNR.

Experiment 7: In this experiment, we compare the different flow scheduling specification modes. Specifically, we compare the short-term performance of the flows with different modes.

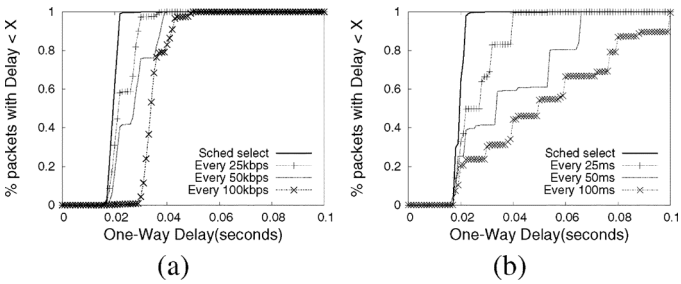


Fig. 15. Effect of model specification and virtual tagging on flow-delay for WF scheduling. (a) Model spec. (b) VTT-DL.

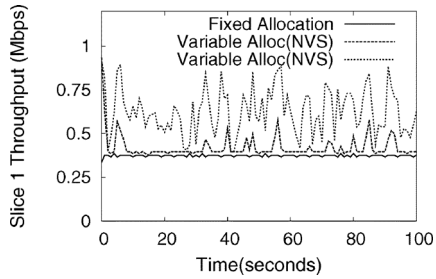


Fig. 16. Efficacy of utilization with NVS.

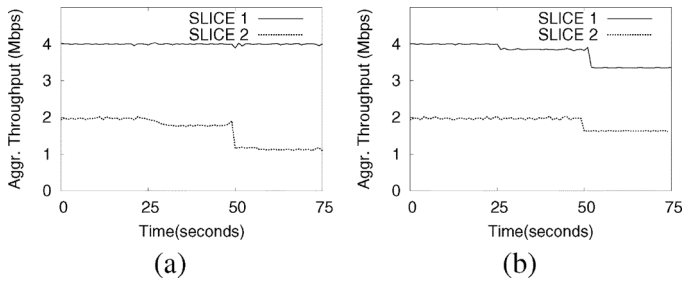


Fig. 17. Isolation across slices with bandwidth based reservations. (a) NVS. (b) NVS-variant.

We setup two flows of a slice such that the first flow reserves 500 kb/s while the second flow reserves 2.5 Mb/s. We run backlogged UDP traffic for the second flow and measure the per-packet delay for the first flow. The offered load for the first flow is set such that it operates just below saturation.

For model specification, we use a weight distribution of the WF scheduling algorithm. Due to the discretization of the weights, the error in the average achieved rate may cause variations in the per-packet one-way delay of a flow. We measure the per-packet delay for three different weight distributions, sampled at avgRate of 25, 50, and 100 kb/s, respectively, and compare with scheduler selection (i.e., with the WF scheduler within the base station). Fig. 15(a) shows as expected that as the sampling becomes coarser, the one-way delay increases. This graph allows both an MNO and a slice owner to strike the right tradeoff between the number of samples sent and the increase in delay. We could also consider interpolation for calculating intermediate weights, although this approach needs further exploration.

For virtual time tagging (VTT-DL), we consider a similar setup, with the feedback of achieved rate given to the flow scheduler at different time intervals. Fig. 15(b) shows that as the feedback interval increases, packets observe higher delays. This graph again allows both an MNO and a slice owner to strike the right tradeoff between the feedback interval and the observed increase in delay.

C. Utilization

Due to centralized scheduling at MAC frame granularity, NVS can reallocate bandwidth between lightly loaded and heavily loaded slices in the downlink and uplink directions at fine timescales. We now demonstrate this efficacy for uplink.

Experiment 8: In this experiment, we consider two slices: Slice 1 with uplink backlogged BE flows, and Slice 2 with uplink RTPS flows carrying video. Video traffic fluctuates with time, thereby requiring different amounts of bandwidth at different times. Fig. 16 shows the aggregate throughput of Slice 1, whose BE flows occupy as much bandwidth as available. When the reservation is fixed for Slice 2, Slice 1 gets a constant 350 kb/s, whereas when NVS is allowed to reallocate Slice 2's unused bandwidth to Slice 1, the graph shows for two different video rates that the bandwidth achieved by Slice 1 is much higher than 350 kb/s, while causing minimum distortion to the videos.

D. Slice Provisioning

We now show that NVS ensures effective coexistence of slices with bandwidth-based provisioning irrespective of fluctuations within slices and a mixture of slices with bandwidth-based and resource-based provisioning.

Experiment 9: The main goal of this experiment is to show the importance of (1) in ensuring isolation, specifically with slices requesting bandwidth-based provisioning. We consider two slices with 4 and 2 Mb/s reservation. Each slice has two clients initially at an MCS of 16QAM-1/2. When admitted, the base station capacity is sufficient to meet the reservation of both slices (i.e., (5) is feasible, and $\tilde{r}_g^{\text{eff}}$ is set at the transmission rate corresponding to 16QAM-1/2 for both slices). After the experiment begins, the clients of Slice 2 are moved away from the base station such that their MCS drops to QPSK-3/4 and QPSK-1/2 at 25 and 50 s, respectively. Fig. 17 shows the actual throughput of the slices with NVS and with a variant of NVS that does not scale down the slice reservation using (1). Clearly, Slice 1's throughput does not get affected in the former case, but does reduce in the latter case; this shows the lack of isolation to other slices without scaling down the slice reservation using (1).

Experiment 10: We now evaluate NVS's behavior with a mix of resource-based and bandwidth-based provisioning. We consider two slices SLC1 and SLC2 with reservations as 600 kb/s and 1.2 Mb/s, and two slices SLC3 and SLC4 with reservations of 15% and 30% of slots. Each slice has two clients running downlink nRTPS flows with backlogged traffic generated by Iperf. We consider two scenarios, when all clients are at MCS of QPSK-1/2 and 16QAM-3/4, respectively. For illustrating the difference in allocation by NVS, just for this experiment, we prevent NVS from giving bandwidth to slices beyond their reservations. For Slices 1 and 2, Fig. 18(a) and (b) shows that when the channel quality improves, the bandwidth

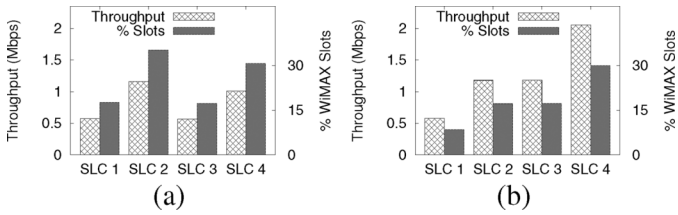


Fig. 18. Coexistence of slices with bandwidth- and resource-based provisioning. (a) QPSK-1/2. (b) 16QAM-3/4.

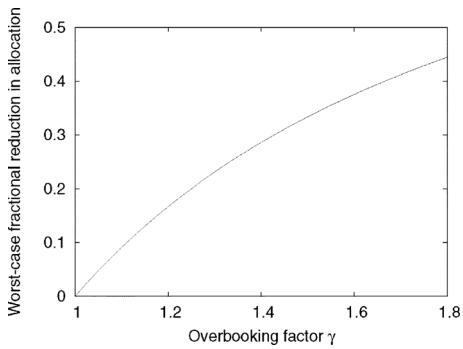


Fig. 19. Fractional rate/resource reduction as a function of γ.

requirements are met while taking fewer slots. Here, NVS focuses on meeting the bandwidth requirements. Comparatively for Slices 3 and 4, when the channel quality improves, the number of resource slots are kept the same, thereby achieving increased throughputs. Here, NVS focuses on meeting slot requirements. The graphs demonstrate that NVS can simultaneously meet the requirements of slices with bandwidth-based and resource-based reservations.

V. Slice Admission Control Extension

In practice, network operators may choose to relax the admission control constraint (5) in Section III-A slightly to allow overbooking and occasional violation of SLAs, which may bring higher revenues to network operators because typically not all slices occupy all their reserved bandwidth or resources. As a result, a general admission control policy is to admit a new slice if

$$\sum_{g\in G_1} \frac{\tilde{r}_g^{\mathrm{rsv}}}{\tilde{r}_g^{\mathrm{eff}}} + \sum_{g\in G_2} t_g^{\mathrm{rsv}} \le \gamma$$

where $\gamma \ge 1$ is the overbooking factor.

With an overbooking factor $\gamma > 1$, when every slice has enough traffic to send at its reserved rate (or radio resource), the base station cannot fulfill all SLAs. In this case, our online slice scheduling algorithm ensures that every slice gets a fraction of its reserved rate r_g^{rsv} (or radio resource t_g^{rsv}). In particular, slice g will receive a rate of $r_g^{\mathrm{rsv}}/\gamma$ (or $t_g^{\mathrm{rsv}}/\gamma$), resulting in a reduction of $\frac{\gamma-1}{\gamma}$ fraction of its reservation. Fig. 19 shows the fractional reduction versus the overbooking factor. For instance, if the maximum tolerable reduction for a slice is 30% of the reserved rate, an MNO may choose $\gamma \le 1.43$. Note that the reduction plotted in Fig. 19 is the worst-case scenario wherein all slices have sufficient traffic to send at their reservations. In practical systems, due to fluctuating traffic arrivals, the actual reduction could be much lower than the worst-case scenario. The graph, along with the estimation of practical traffic arrival patterns, can provide useful insight for setting up reasonable SLAs between the MNOs and slice owners.

VI. Limitations and Discussion

NVS encounters one limitation with uplink flows. For efficiency, the IEEE 802.16e mobile WiMAX standard allows a flow of a client to steal bandwidth allocated to another flow of the same client. This can violate isolation across flows if the two flows belong to different slices. Fixing this problem would require modifying the client WiMAX drivers. NVS is limited to the isolation possible at the base-station uplink scheduling level.

NVS currently virtualizes wireless resources on a per-base-station basis. For some deployment scenarios, virtualization may be required across a network of base stations. For instance, a corporate data plan may involve a certain resource allocation across the entire network, irrespective of the location of clients. We plan to extend NVS in this direction in our future work.

Deployment scenarios such as GENI and corporate Intranets demand end-to-end isolation [17], which requires virtualization on all wired and wireless network components between servers and clients. NVS addresses only the wireless access aspect and requires integration with complementary technologies [22], [28] for end-to-end isolation.

While the design and implementation of NVS is in the context of a time division duplex (TTD)-based OFDMA system, we note that the design is equally applicable to a frequency division duplex (FDD)-based system since NVS abstractly considers wireless resources as slots of a certain channel width and time.

VII. Related Work

A. Wireless Network Virtualization

Several GENI design documents describe proposals and issues of wireless network virtualization [37], [39]. In this context, Bhanage *et al.* [22] focus on virtualizing WiMAX network resources from the ASN gateway, with the base station as a black-box. This approach, however, cannot support uplink virtualization and will only result in downlink virtualization at a coarse granularity. Although several commercial deployments of MVNOs exist worldwide [8], [33], little information is publicly available on the resource virtualization aspects, if any. Hardware virtualization solutions [13] are more appropriate for traditional MNOs attempting to cut hardware costs. In contrast, NVS virtualizes resources at the flow-level, which is appropriate for enabling the deployment scenarios such as MVNOs, corporate data plans, and SLNs. In the WiFi domain, Smith *et al.* [46] discuss virtualization on 802.11 hardware for testbed sharing and experiment repeatability, and MadWiFi's Virtual AP [9] and MultiNet [23] enable virtualization of 802.11 network interfaces. The challenges and solutions are specific to 802.11.

B. Wired Network Virtualization

Resource virtualization in the wired domain has received significant focus in the past years [5], [12], [15], [26]–[28], [49]. The most relevant related work in this domain deals with hierarchical link-sharing strategies across multiple agencies [26], [27], [49], while considering both guaranteed real-time services and best-effort services. These works, however, do not completely decouple flow selection from slice selection in the interest of packet-level isolation and QoS guarantees. NVS, in contrast, enables greater level of customization to empower slice owners to use differentiating flow management techniques. Recently, He *et al.* [28] designed an adaptive substrate network that supports multiple virtual networks running customized protocols. Such solutions are complementary to NVS from the end-to-end solution perspective. Furthermore, wired domain solutions do not have to address the challenges unique to the wireless domain such as uplink support, channel variations, and types of provisioning.

C. QoS Scheduling in Wireless Networks

Scheduler design in wireless networks, especially in WiMAX networks [47], is a hot topic and very relevant to NVS. The works we discuss are some of the flow scheduling proposals that have been evaluated theoretically or in simulations. NVS enables these proposals to be evaluated on real network testbeds with realistic traffic, as we demonstrated by implementing the GWF scheduler [48]. Various versions of Deficit Round Robin scheduling [25], [41], [42] with dynamic weights have been explored to provide bandwidth and delay guarantees. Iera *et al.* [29] propose a slight modification of Weighted Fair Queuing across the different WiMAX classes. Ryu *et al.* [40] propose an urgency- and efficiency-based (UEPS) algorithm that maximizes the throughput of non-real-time traffic while maintaining the QoS of real-time traffic. Liu *et al.* [34] propose to dynamically adapt flow priority based on its channel and received service status. Andrews [19] proposes an algorithm to maximize total throughput of the system such that the minimum reserved rate of all the users is met.

Fair queuing in wireless networks received significant research attention in the past. However, these solutions do not consider scheduling across slices or groups of flows. For instance, Lu *et al.* [35] describe an ideal wireless fair-scheduling algorithm for flows that provides a packetized implementation of a modified version of fluid fair queuing (FFQ). Ramanathan *et al.* [38] propose an approach in which long-term fairness is guaranteed to flows that have not received satisfactory service in the short term due to poor channel conditions. Nandagopal *et al.* [36] discuss a unified architecture for design of wireless fair queuing algorithms for fair service across flows.

VIII. Conclusion

This paper describes a network virtualization substrate (NVS) for effective virtualization of wireless resources in a WiMAX network. NVS aims to strengthen the role of cellular networks in providing enhanced experience to users, facilitates providing greater differentiation of services among content providers and MVNOs, and opens up several new research opportunities. Through a systematic design, prototype implementation and detailed evaluation on a WiMAX testbed, we demonstrate that NVS achieves the above goals.

References

[1] "AT&T available data plans," AT&T, Dallas, TX, 2011 [Online]. Available: http://www.wireless.att.com

[2] "Beceem WiMAX chips for mobile applications," Broadcom, Irvine, CA, 2009 [Online]. Available: http://www.beceem.com/

[3] "Click modular router," 2011 [Online]. Available: http://read.cs.ucla.edu/click/

[4] "EvalVid," Technical University of Berlin, Berlin, Germany, 2011 [Online]. Available: www.tkn.tu-berlin.de/research/evalvid/

[5] "GENI," [Online]. Available: http://www.geni.net/

[6] "IEEE 802.16m," 2011 [Online]. Available: http://www.ieee802.org/16/tgm/

[7] "Iperf," 2011 [Online]. Available: http://sourceforge.net/projects/iperf/

[8] "List of MVNOs and their specializations," 2011 [Online]. Available: http://www.mobileisgood.com/mvno.php

[9] "Madwifi virtual access point," 2011 [Online]. Available: http://madwifi-project.org/wiki/VAP

[10] "Network Time Protocol," 2011 [Online]. Available: http://www.ntp.org

[11] "Picochip femtocell solutions," Picochip, Bath, U.K., 2011 [Online]. Available: http://www.picochip.com/

[12] "Planetlab testbed," 2007 [Online]. Available: http://www.planet-lab.org

[13] "Vanu networks," Vanu, Inc., Cambridge, MA, 2011 [Online]. Available: http://www.vanu.com/

[14] "VMware," VMware, Inc., Palo Alto, CA, 2011 [Online]. Available: http://www.vmware.com

[15] "VMware infrastructure in a Cisco network environment," Cisco Systems, Inc., San Jose, CA, 2008 [Online]. Available: http://www.cisco.com/application/pdf/en/us/guest/netsol/ns304/c649/ccmigration_09186a00807a15d0.pdf

[16] "Xen," 2011 [Online]. Available: http://www.xen.org

[17] "Geni design principles," *Computer*, vol. 39, no. 9, pp. 102–105, 2006.

[18] J. G. Andrews, A. Ghosh, and R. Muhamed, *Fundamentals of WiMAX*. Upper Saddle River, NJ: Prentice-Hall, 2008.

[19] M. Andrews, L. Qian, and A. Stoylar, "Optimal utility based multi-user throughput allocation subject to throughput constraints," in *Proc. IEEE INFOCOM*, 2005, vol. 4, pp. 2415–2424.

[20] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, "D-ITG distributed internet traffic generator," in *Proc. QEST*, Sep. 2004, pp. 316–317.

[21] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 2004.

[22] G. Bhanage, I. Seskar, R. Mahindra, and D. Raychaudhuri, "Virtual basestation: Architecture for an open shared WiMAX framework," in *Proc. ACM SIGCOMM VISA Workshop*, 2010, pp. 1–8.

[23] R. Chandra, "A virtualization architecture for wireless network cards," Ph.D. dissertation, Cornell Univ., Ithaca, NY, 2006.

[24] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: An overlay testbed for broad-coverage services," *Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, 2003.

[25] C. Cicconetti, L. Lenzini, and E. Mingozzi, "Quality of service support in IEEE 802.16 networks," *IEEE Netw.*, vol. 20, no. 2, pp. 50–55, Apr. 2006.

[26] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 4, pp. 365–386, Aug. 1995.

[27] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 690–704, Oct. 1997.

[28] J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, and M. Chiang, "DaVinci: Dynamically adaptive virtual networks for a customized internet," in *Proc. ACM CoNEXT*, 2008, Article no. 15.

[29] A. Iera, A. Molinaro, and S. Pizzi, "Channel-aware scheduling for QoS and fairness provisioning in IEEE 802.16/WiMAX broadband wireless access systems," *IEEE Netw.*, vol. 21, no. 5, pp. 34–41, Sep. 2007.

[30] A. Kiiski, "Impacts of MVNOs on mobile data service market," in *Proc. 17th Eur. Regional ITS Conf.*, Aug. 2006.

[31] R. Kokku, "SHARE: Run-time system for high-performance virtualized routers," Ph.D. dissertation, University of Texas, Austin, TX, 2005.

[32] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: A virtualization substrate for WiMAX networks," in *Proc. ACM MobiCom*, 2010, pp. 233–244.

[33] G. Lenahan, "With the right support, MVNOs can enrich network operators with innovation differentiation and market share," Telcordia, Piscataway, NJ, Whitepaper, 2008 [Online]. Available: http://www.telcordia.com/library/whitepapers/mvno_mvne.jsp

[34] Q. Liu, X. Wang, and G. B. Giannakis, "A cross-layer scheduling algorithm with QoS support in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 55, no. 3, pp. 839–847, May 2006.

[35] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 473–489, Aug. 1999.

[36] T. Nandagopal, S. Lu, and V. Bharghavan, "A unified architecture for the design and evaluation of wireless fair queueing algorithms," in *Proc. ACM MobiCom*, 1999, pp. 132–142.

[37] S. Paul and S. Seshan, "Virtualization and slicing of wireless networks," GENI, Design Doc. 06-17, 2006.

[38] P. Ramanathan and P. Agrawal, "Adapting packet fair queueing algorithms to wireless networks," in *Proc. ACM/IEEE MobiCom*, Oct. 1998, pp. 1–9.

[39] D. Raychaudhuri and M. Gerla, "New architectures and disruptive technologies for the future Internet: The wireless, mobile and sensor network perspective," GENI, Tech. Rep. 05-04, 2005.

[40] S. Ryu, B.-H. Ryu, H. Seo, M. Shin, and S. Park, "Wireless packet scheduling algorithm for OFDMA system based on time-utility and channel state," *ETRI J.*, vol. 27, pp. 777–787, Dec. 2005.

[41] A. Sayenko, O. Alanen, and T. Hamalainen, "Scheduling solution for the IEEE 802.16 basestation," *Int. J. Comput. Telecommun. Netw.*, vol. 52, pp. 96–115, Jan. 2008.

[42] A. Sayenko, O. Alanen, J. Karhula, and T. Hamalainen, "Ensuring the QoS requirements in 802.16 scheduling," in *Proc. Workshop Model. Anal. Simul. Wireless Mobile Syst.*, Oct. 2006, pp. 108–117.

[43] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, "Network virtualization architecture: Proposal and initial prototype," in *Proc. 1st ACM VISA*, 2009, pp. 63–72.

[44] S. Sesia, I. Toufik, and M. Baker*, LTE, The UMTS Long Term Evolution: From Theory to Practice*, ser. Interscience. Hoboken, NJ: Wiley, 2009.

[45] S. Shenker, "Fundamental design issues for the future Internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 7, pp. 1176–1188, Sep. 1995.

[46] G. Smith, A. Chaturvedi, A. Mishra, and S. Banerjee, "Wireless virtualization on commodity 802.11 hardware," in *Proc. WinTECH*, 2007, pp. 75–82.

[47] C. So-In, R. Jain, and A.-K. Tamimi, "Scheduling in IEEE 802.16e mobile WiMAX networks: Key issues and a survey," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 2, pp. 156–171, Feb. 2009.

[48] C. So-In, R. Jain, and A.-K. A. Tamimi, "Generalized weighted fairness and its application for resource allocation in IEEE 802.16e mobile WiMAX," in *Proc. 7th Annu. IEEE Consumer Commun. Netw. Conf.*, 2010, pp. 784–788.

[49] I. Stoica, H. Zhang, and T. S. E. Ng, "A hierarchical fair service curve algorithm for link-sharing, real-time, and priority services," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 185–199, Apr. 2000.

[50] D. Varoutas, D. Katsianis, T. Sphicopoulos, K. Stordahl, and I. Welling, "On the economics of 3G mobile virtual network operators (MVNOs)," *Wireless Pers. Commun.*, vol. 36, no. 2, pp. 129–142, 2006.

[51] Y. Zhu, R. Zhang-Shen, S. Rangarajan, and J. Rexford, "Cabernet: Connectivity architecture for better network services," in *Proc. ACM CoNEXT ReArch Workshop*, 2008, Article no. 64.

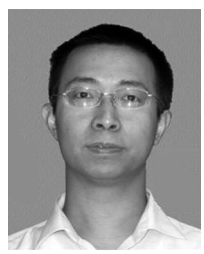
Ravi Kokku (M'11) received the M.S. and Ph.D. degrees in computer science from the University of Texas at Austin in 2005.

He is a Researcher with IBM Research, Bangalore, India. From 2005 to 2011, he was a Research Staff Member with NEC Laboratories America, Inc., Princeton, NJ. His research interests include virtualization, resource management and novel service enablement in wireless networks, and wide area data replication.

Dr. Kokku is a member of the Association for Computing Machinery (ACM). He was a recepient of the IBM Ph.D. fellowship from 2001 to 2004.

Rajesh Mahindra received the Master's degree in electrical and computer engineering from Rutgers University, New Brunswick, NJ, in 2007.

He is an Associate Research Staff Member with NEC Laboratories America, Inc., Princeton, NJ. His research interests include video streaming over wireless, wireless network virtualization, and wireless resource management.



Honghai Zhang (M'08) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1998, and the Ph.D. degree from the University of Illinois at Urbana–Champaign in 2004, both in computer science.

He is currently a Research Staff Member with NEC Laboratories America, Inc., Princeton, NJ. Prior to his current position, he was with the Wireless Advanced Technology Laboratory, Alcatel-Lucent, Bell Labs, Whippany, NJ. His research interests include scheduling, interference mitigation, and video streaming in mobile wireless networks and cognitive radio networks.

Dr. Zhang was a recipient of Vodafone Fellowship during his Ph.D. study.

Sampath Rangarajan (M'91–SM'05) received the M.S. degree in electrical and computer engineering and Ph.D. degree in computer science from the University of Texas at Austin in 1987 and 1990, respectively.

He heads the Mobile Communications and Networking Research Department, NEC Laboratories America, Princeton, NJ. Previously, he was with the Networking Research Center, Bell Laboratories, Holmdel, NJ. Prior to that, he was a cofounder and Vice President of Technology with Ranch Networks, Morganville, NJ, a venture-funded startup in the IP networking space. Earlier, he was a Researcher with the Systems and Software Research Center, Bell Laboratories, Murray Hill, NJ. Before joining Bell Laboratories, he was an Assistant Professor with the Electrical and Computer Engineering Department, Northeastern University, Boston, MA. His research interests span the areas of mobile communications, mobile networks, and distributed systems.

Dr. Rangarajan has been on the Editorial Boards of the IEEE TRANSACTIONS ON COMPUTERS and *Mobile Computing and Communications Review*. He is currently a member of the Editorial Board of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.