Cellular Wireless Resource Slicing for Active RAN Sharing

Ravi Kokku*, Rajesh Mahindra, Honghai Zhang and Sampath Rangarajan
NEC Laboratories America Inc., Princeton, USA
*Currently at IBM Research, Bangalore, India

ABSTRACT

We present the design and implementation of CellSlice, a novel system for slicing wireless resources in a cellular network for effective Radio Access Network (RAN) sharing. CellSlice is a gateway-level solution that achieves the slicing without modifying the basestations' MAC schedulers, thereby significantly reducing the barrier for its adoption. Achieving slicing with a gateway-level solution is challenging, however, since resource scheduling decisions occur at the basestations at fine timescales, and these decisions are not visible at the gateways. In the uplink direction, CellSlice overcomes the challenge by *indirectly constraining* the uplink scheduler's decisions using a simple feedback-based adaptation algorithm. For downlink, we build on the technique used by NVS, a native basestation virtualization solution, and show that effective downlink slicing can be easily achieved without modifying basestation schedulers. We instantiate a prototype of CellSlice on a Picochip WiMAX testbed. Through both prototype evaluation and simulations, we demonstrate that CellSlice's performance for both remote uplink and remote downlink slicing is close to that of NVS. CellSlice's design is access-technology independent, and hence can be equally applicable to LTE, LTE-Advanced and WiMAX networks.

Keywords

Cellular Networks, RAN Sharing, Active Sharing, Resource Management

1. INTRODUCTION

Recent years are witnessing an unprecedented surge in data traffic on cellular networks, forcing mobile network operators (MNOs) to constantly increase network provisioning, which increases capital and operational costs correspondingly. At the same time, as cellular networks evolve from being voice-dominated to being data-dominated, the revenue generated per unit wireless resource is reducing significantly. This trend of decreasing revenues and increasing costs is already making MNOs consider radical changes to the way networks are deployed and operated.

Active RAN Sharing: One radical change that is receiving considerable attention recently is active RAN (Radio Access Network) sharing [1, 12, 17]. A study by ABI research [4] indicates that operators could save upto $60 Billion through active RAN sharing in a period of 5 years. RAN sharing enables significant reduction in equipment in low traffic areas and results in atleast 100% increased rollout speed with a given cost [9].

The Radio Access Network typically involves the basestations and the immediately connected gateways that manage the basestations. For instance, in a 3G network, the NodeBs and Radio Network Controllers (RNCs) comprise the RAN. In a WiMAX network, the RAN consists of basestations and ASN Gateways [19]. Active RAN Sharing involves sharing the gateways and basestations across multiple entities (e.g. network operators) with either separate spectrum resources for each entity or shared spectrum resources through spectrum pooling [1, 33]. Active RAN sharing can also enable several new interesting deployment scenarios. For example, one mobile network operator (MNO) may become a virtual operator (MVNO) [25, 27] on another MNO's network during traffic surges, which reduces cost for the first MNO, while generating revenue for the second. Or, a service provider (e.g. Youtube) may lease resources from an MNO (e.g. Verizon) to improve service quality to its customers or to share costs for enabling video services to mobile users [18]; this model generates new revenue streams for the MNO.

Our Approach: CellSlice focuses on deployments with shared-spectrum RAN sharing. We believe that such deployments will be more prevalant in the future. This is because studies show that spectrum resources owned by a given operator are often underutilized on an average [28], and hence regulatory bodies are increasingly considering spectrum sharing for better statistical multiplexing and improved utilization [1]. In this paper, we focus on the problem of managing the wireless resources across multiple entities for enabling effective RAN sharing. We define a *slice* as a group of user flows that belong to a single entity such as an MVNO, and require a chosen fraction of spectrum resources to

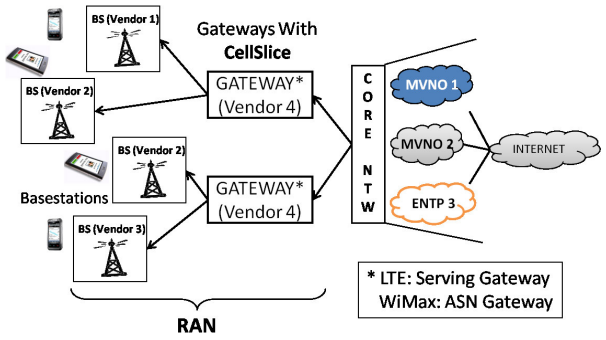


Figure 1: Deployment Point of CellSlice.

be allocated for satisfying the SLAs; the term *slicing* indicates meeting the resource allocation on an aggregate for the group of flows of the slice. While the entities co-hosted on the physical network contend for several resources on gateways and basestations (such as processors, memory and backhaul wired links), wireless (spectrum) resources are typically the most scarce and expensive resources in cellular networks, and their effective slicing is crucial for successful active RAN sharing. The other components can be shared by adopting past well known solutions [16, 23, 24, 31].

Specifically in this paper, we ask the following practical question: *can we design a RAN sharing solution that requires minimal modifications to existing basestations, such that the solution is attractive for immediate and broader deployment?* While wireless resource slicing may be better achieved by natively modifying the basestation schedulers [26], achieving it remotely with minimal modifications to basestations has significant advantages (See Figure 1). Firstly, the solution can be more easily deployed at gateways that manage 100s of basestations, or transparently as middleboxes in the mobile core network. Such "minimum footprint" solutions typically encounter fewer hurdles for adoption.

Secondly, remote slicing from higher levels in the mobile network hierarchy makes network-wide resource allocation across a set of basestations for different entities significantly simpler. Finally, network operators often use basestations from multiple vendors. For instance, Bharti Airtel selected Huawei, NSN and Ericsson as its vendors for its 3G RAN [6] while KDDI obtained basestations from Samsung, Motorola and NEC for its LTE deployments [10]. In such scenarios, some vendors may not support resource slicing due to intellectual property protection by other vendors, or may require significant standardization efforts for inter-working, which in itself can hinder deployability. Also, due to the different scales of operation and competition in the market, vendors prefer cost reduction measures on basestations, and feature enhancements on gateways.

Performing wireless resource slicing remotely such that a *group of flows* achieve a given fraction of resources is a harder problem than per-flow allocation. This is because wireless resource allocation in cellular networks is centrally done on a per-flow basis by the basestation schedulers at fine timescales; these basestation scheduling decisions are not visible at the gateways and can violate slice specific reservations and isolation. The problem is especially challenging in the uplink direction, since traffic originates at the clients and the gateways can only observe the traffic *after* the resource allocation has already been done.

Contributions: To overcome the above challenge and achieve uplink resource slicing, our primary contribution involves the design, implementation and analysis of a simple, yet effective algorithm for *indirectly constraining* the scheduling decisions taken by the basestation's uplink scheduler. CellSlice leverages access to a flow shaping parameter that is commonly used in basestations [19, 29] to restrict the resources allocated to a given flow. CellSlice adapts this parameter for flows of each slice dynamically such that (1) the resulting allocation by the basestation meets the slice requirements, while providing isolation to other slices, (2) flows within a slice converge to a fair allocation of resources, even when they arrive and depart dynamically during system operation, and (3) resource utilization is maximized.

Our second contribution involves (1) moving the downlink slice scheduling functionality of NVS (our recently proposed native basestation virtualization solution [26]) into CellSlice on the gateway, and (2) *controlling* the number of packets sent to the basestation at any point of time. Such control avoids queue buildup at the basestation, thereby enforcing scheduling decisions made at the gateway. The design of CellSlice is oblivious to a particular cellular technology and is equally applicable to LTE, LTE-Advanced and WiMAX.

We implement and evaluate a prototype of CellSlice on a Picochip WiMAX (IEEE 802.16e) network testbed. We compare the efficacy of CellSlice to NVS [26] in terms of inter-slice isolation, intra-slice flow convergence, and wireless resource utilization. For instance, our experiments show in both uplink and downlink directions that the basestation utilization with CellSlice is 90% of that with NVS more than 90% of the time, while providing similar isolation as NVS.

Paper organization: Section 2 provides background, related work and design considerations for remote resource slicing. Section 3 presents the design, analysis and prototype implementation of uplink slicing in CellSlice, and Section 4 presents the corresponding evaluation. Section 5 presents the design and evaluation of downlink resource slicing. Section 6 discusses the limitations and future work. Section 7 concludes.

2. PROBLEM FORMULATION

In this section, we present a background on cellular networks and active RAN sharing across multiple enti-

ties. We then discuss the key requirements to be satisfied while slicing wireless resources to enable effective active RAN sharing, related work, and design considerations for remotely slicing resources from gateways.

2.1 Background

RAN sharing has been receiving significant interest in recent times due to the inherent cost-saving benefits it provides [33, 9]. As cellular networks evolve towards supporting data-dominated traffic, trends show that the revenue per byte is decreasing, whereas capital and operational costs are increasing significantly with the rapidly increasing traffic; active RAN sharing promises to reduce these costs significantly. Active RAN sharing across multiple entities can be done in several ways. (1) In the MORAN (Multi Operator RAN) approach, the entities share RAN gateways (e.g. RNC in 3G, or S-Gateway in LTE or ASN Gateway in WiMAX) and some parts of the basestations such as processing elements, whereas the radio and power amplifiers remain physically independent to enable the entities to use their different assigned frequencies. (2) In the MOCN (Multi Operator Core Network) approach, which has been specified since 3GPP Rel. 6 (3GPP TS 23.251) [2], operators share both the RAN gateways and the basestations, and may also pool their frequencies. In either approach, RAN sharing can also happen with virtual network operators (MVNOs) or application service providers leasing resources from physical network operators. Several network sharing projects are underway across the world using both the MOCN and MORAN approaches [9], although we are not familiar with the technical details. Our focus in this paper is on the MOCN approach.

A key component of active RAN sharing is dividing the wireless resources such that each entity receives its share of allocation. Such slicing can be done with either a basestation-level solution or a gateway-level solution. As discussed earlier, compared to basestation-level solutions, remotely slicing wireless resources makes the solution easily deployable, enables easier network-wide resource reservations for slices, and enables operation with basestations from multiple vendors, some or all of which may not support native virtualization. Nevertheless, either of the approaches has its own advantages, and which solution appeals to a network equipment vendor or cellular network operator is at best speculative at this point. We instead focus on the technical merit of the solutions in our research and explore a gateway-level solution in this paper.

For efficient resource allocation, basestations incorporate uplink and downlink schedulers that allocate wireless resources across multiple flows. We can abstract the scheduling process as a task of filling "resource slots" in each MAC frame; these can be resource blocks in LTE terminology or slots in the WiMAX terminology. Figure 2 shows an overview of the basestation schedulers. The schedulers use several parameters (including QoS parameters and channel quality) to determine the schedule. CellSlice's goal will be to remotely control these scheduling decisions to ensure that each entity (i.e. its flows as an aggregate) receive their share of the wireless resources.

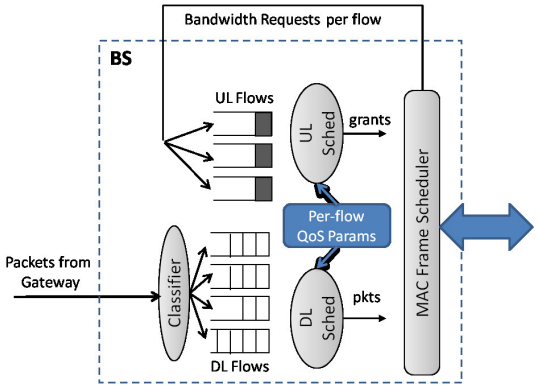


Figure 2: Basestation Scheduler Overview.

2.2 Solution Requirements

We consider groups of flows as slices and manage resource allocation across the slices such that each slice's resource reservation is satisfied. In meeting the goal, a solution is expected to satisfy the following three requirements across the different slices:

1. **Inter-slice Isolation**, i.e., any change in one slice such as change in channel conditions or user mobility should not affect the resource allocation for other slices.
2. **Customizable Intra-slice Resource Allocation**, i.e., inter-flow resource allocation within each slice should converge to a ratio defined by a slice owner's custom policy, and
3. **Efficient Radio Resource Utilization**, i.e. resources unutilized by one slice may be allocated to other slices that can utilize them.

2.3 Related Work

Recently, there have been several efforts on wireless network virtualization, some of which target resource slicing across multiple entities [26, 20], while others target infrastructure and control plane support [13, 21, 34]. In what follows, we discuss these works and differentiate them from our work.

The system NVS [26] enables wireless resource slicing at a cellular basestation by modifying the MAC schedulers within the basestation. This solution provides fine-timescale isolation and achieves high basestation utilization since resource allocation is determined at

MAC frame granularity. However, the solution faces high barrier for deployment since it requires modifications to the schedulers; basestation vendors are often sensitive about schedulers since schedulers are the main differentiating components in an otherwise-standards-compliant product.

Another recent effort led to VNTS [20], a solution for remote slicing of downlink wireless resources at basestations through traffic shaping. This approach has several drawbacks: (1) The traffic shaping rate on each client is set assuming that all clients of the slice have backlogged traffic. If some clients do not have traffic, a slice cannot meet its reservations even if its other clients have backlogged traffic. (2) If a slice does not have traffic, the spare resources are not utilized for other slices, leading to basestation under-utilization. (3) VNTS does not currently support uplink resource slicing, which is crucial for commercial deployments. With increasing uplink traffic due to video uploads, social networking applications, video calls, etc., isolation in the uplink direction will be crucial for effective RAN sharing; otherwise, a slice with a large number of flows can lead to starvation for other slices.

vBTS [15] is a virtualized basestation solution that enables sharing radio access network components at the hardware level and running multiple basestation protocol stacks in software. This solution is more appropriate for sharing hardware infrastructure between native MNOs with separate dedicated spectrum resources (i.e. for the MORAN approach). CellSlice operates at a higher level of abstraction for simplicity, easier deployability and even enables entities such as application service providers or MVNOs to lease network resources.

Finally, several solutions target problems complementary to wireless resource slicing in cellular networks. OpenRoads (OpenFlow wireless) [34] virtualizes flow address space within a mobile network and enables interworking between multiple wireless technologies for simultaneously enabling the deployment of novel protocols. The *virtual basestation* work [13, 21] focuses on design considerations for slice monitoring and management, isolation on wired networks and virtual machine support on the gateways. Solutions in the WiFi domain [11, 22, 30] mainly address deployment challenges specific to WiFi networks, whereas other related efforts focus on open shared hardware platforms [32] to co-host multiple wireless standards.

2.4 Design Considerations

CellSlice focuses on wireless resource slicing both in the uplink and the downlink directions without directly modifying the basestation schedulers, but instead by *constraining* the basestation's scheduling decisions from a remote gateway. Remote uplink slicing is especially challenging since wireless resource reservation requests from the clients for enabling uplink transmissions terminate at the basestations and are not visible at the gateways. Hence, controlling the resource allocation decided upon by the basestations such that slice isolation is achieved in the uplink direction is not straight-forward; this forms the major focus of this paper. For downlink resource slicing, while all traffic flows through the gateways and facilitates imposing slice-specific reservations, it is important to regulate the traffic sent to the basestations. Sending less traffic than the wireless channel capacity at a basestation causes underutilization. Whereas, sending more traffic than the capacity leads to increased queues at the basestation, beyond which resource allocation is performed according to the basestation scheduling policy that can override the slice specific allocations and violate slice isolation. Overcoming this challenge is the second focus of the paper.

In addressing the above problems, our goal is to require (or introduce) minimum additional communication between gateways and basestations, and minimal support from the basestations. To achieve the same, we assume that minimal "sensor" and "actuator" elements are exposed by the basestations.

- On the sensing side, we assume (1) periodic feedback of resource (slot) utilization that is defined as the ratio of the slots used for transmission and the total available slots in the period, and (2) average per-flow MCS (modulation and coding scheme) that indicates how efficiently a given slot is utilized for a user. Note that this information is also a part of the GENI WiMAX API [8]. Utilization should be readily available at standards-compliant LTE basestations for easy exposition [3], and MCS is already maintained on basestations for aiding MAC scheduling.

- On the actuator side, we assume a single shaping parameter (hereafter called maximum sustained rate parameter) per flow; the gateways can configure this parameter, and the basestations ensure that each flow does not receive more throughput than the maximum sustained rate. The WiMAX standard already has such a parameter for each QoS service class.

3. UPLINK RESOURCE SLICING

At the core of CellSlice's approach is the consideration of groups of flows into slices, with each slice belonging to an independent entity and requiring a given resource allocation. We define a flow as a stream of packets of a user on which a particular set of QoS parameters are applied by the operator's network. For instance, a flow maps to a bearer in LTE terminology and a service flow in WiMAX terminology. Each entity (slice) co-hosted

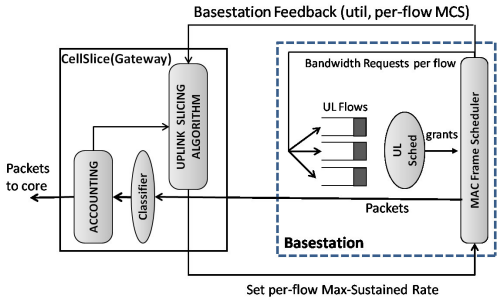


Figure 3: CellSlice's Uplink Slicing.

on the physical network may have multiple users, and each user may have multiple flows simultaneously.

Each flow belongs to only one slice, although different flows of a user can belong to different slices at the same time; this association can be established during flow setup, depending on the slice specification for the given deployment scenario. For instance, all flows with source or destination IP matching `youtube.com`'s IP addresses may be categorized as one slice.

3.1 Design

Figure 3 shows the overview of the software architecture of uplink slicing, as a part of the gateway. Mainly, the solution includes an uplink slicing algorithm that determines the appropriate setting of maximum sustained rate parameter for each flow periodically.

3.1.1 Model

CellSlice allows slices to make reservations in terms of *fraction of resources* desired per unit time, i.e., a slice may request 20% of basestation resources. Let us represent the slot allocation reserved by a slice owner for slice g as t_g^{rsv}, which is a fraction of the total resource slots in a unit time interval. For feasibility of meeting every slice's reservation without running the basestation into an over-utilized state, a slice admission control procedure ensures that $\sum_g t_g^{rsv} \leq \mathcal{U}_H$. $\mathcal{U}_H (< 1)$ is the threshold on the wireless resource utilization above which the basestation is considered to be over-utilized. Although slices make specific resource reservations, we assume that the slices that have enough traffic can make use of additional resources beyond their reservations to improve end-user experience. Hence, an MNO may attempt to maximize the basestation utilization when a subset of the slices do not use their reserved resources.

With such resource-based reservations, depending on the channel conditions of users within the slice and the flow scheduling policy used within the slice, the effective bandwidth achieved by a slice is different at different times. The solution can also be to allow slices to make bandwidth-based reservations, i.e., a slice may request 1 Mbps slice throughput always irrespective of the channel conditions of the users. Although we do not discuss the extension in this paper to simplify the presentation of the main idea, we note that the extension can be done easily as shown in [26], which demonstrates that bandwidth-based reservations can be equivalently converted to resource-based reservations.

Now, let r_{gi}^{max} represent the maximum sustained rate for a flow i of slice g. The eventual goal of CellSlice is to dynamically tune this parameter. We assume that the basestation allocates resources to each uplink flow such that its average rate (bandwidth) r_{gi} is upper-bounded by r_{gi}^{max}. In wireless networks, the rate obtained by a user for a unit resource is dependent on the MCS (modulation and coding scheme) used, which depends on the channel quality of the user. Hence, r_{gi}^{max} can be equivalently related to the fraction of resources required t_{gi}^{max} as

$$r_{gi}^{max} = t_{gi}^{max}.\mathcal{R}_{gi} \tag{1}$$

where $\mathcal{R}_{gi}$ is the effective bitrate (corresponding to the effective MCS) of the user. Let us assume that the basestation sends feedback every τ units of time of the following parameters: utilization u and per-flow effective bitrate $\mathcal{R}_{gi}$.

3.1.2 Uplink Slicing Algorithm

The goal of the slicing algorithm now is to adapt r_{gi}^{max} to ensure that the reservation of each slice is satisfied in a unit time interval τ, i.e.

$$\forall g,\ t_g \ \geq \ t_g^{rsv} \tag{2}$$

where t_g is the allocation for slice g in that time interval. Since t_{gi}^{max} and r_{gi}^{max} are interchangeable as given by Equation 1, the algorithm can adapt either parameter. For simplicity of explanation, we represent the algorithm in terms of t_{gi}^{max}. Algorithm 1 shows the basic procedure for adaptation of t_{gi}^{max} to achieve uplink slicing across the slices. The algorithm involves a simple feedback-based discovery mechanism that (1) sets/resets the parameter t_{gi}^{max} for each flow when the basestation utilization exceeds a threshold, such that new settings of t_{gi}^{max} ensure isolation across slices, and (2) incrementally increases the parameter t_{gi}^{max} for all flows when the uplink resources are underutilized.

Specifically, when the basestation utilization exceeds the threshold $\mathcal{U}_H$, the parameter t_{gi}^{max} of each flow is reset to a value that is a fraction (α_{gi}) of the slice reservation t_g^{rsv} (Line 3 of the algorithm), which ensures slice isolation at that instant. The flag `STRICT` in Line 2 of the algorithm is **false** by default. When the basestation is under-utilized, t_{gi}^{max} for each flow is incremented such that the total utilization would increase proportional to the current under-utilization (Line 5 of the algorithm). This ensures that CellSlice harvests any spare resources aggressively for use by other slices/flows; the parameter γ_{gi} allows tuning the aggressiveness on a per-flow basis.

Algorithm 1 Uplink Slicing

1: Every τ units of time
2: **if** $(u > \mathcal{U}_H \parallel$ `STRICT`$)$ **then**
3: $\quad t_{gi}^{max} \leftarrow \alpha_{gi}.t_g^{rsv}$
4: **else**
5: $\quad t_{gi}^{max} \leftarrow t_{gi}^{max} + (1-u).\gamma_{gi}$
6: **end if**

Figure 4(a) shows the behavior of the increment procedure; it increases rapidly when u deviates significantly from $\mathcal{U}_H$, and less rapidly as it approaches $\mathcal{U}_H$.

In some scenarios (such as for experiment repeatability in GENI testbeds [21]), an operator may wish to provide strict isolation across slices and tradeoff basestation utilization. To achieve this with CellSlice, an operator can simply set `STRICT` to **true**, which ensures that a slice g does not receive resources beyond that allocated in Line 3 of Algorithm 1.

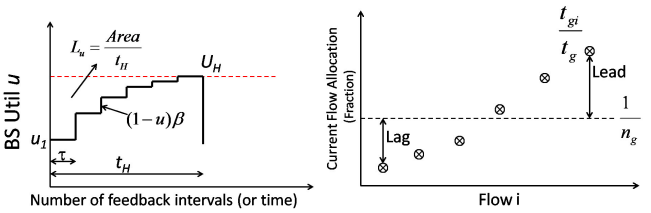
While the overall idea is simple and attractive to a gateway manufacturer for quick implementation, the efficacy of CellSlice relies on how well the quantities α_{gi} and γ_{gi} are set, which we discuss next.

3.1.3 Setting α_{gi}

The parameter α_{gi} for each flow within slice g is set such that $\sum_{i \in g} \alpha_{gi} = 1$, which ensures that the slice on an aggregate receives t_g^{rsv} fraction of resources. CellSlice's current design attempts to equalize the slot allocation across flows of a slice if every flow has traffic to send; i.e. if n_g is the number of flows in the slice and every flow has backlogged traffic, then $\alpha_{gi} = 1/n_g$. However, it is often not the case that all flows have backlogged traffic. Since one of the goals of CellSlice is to maximize the utilization of the basestation, we instead set α_{gi} as

$$\alpha_{gi} = \begin{cases} \frac{t_{gi}}{t_g}(1-\delta) + \frac{\delta}{n_g} & \text{if } t_g > 0 \\ \frac{1}{n_g} & \text{otherwise} \end{cases} \quad (3)$$

If none of the flows of a slice has traffic (i.e. $t_g = 0$), we set $\alpha_{gi} = 1/n_g$ to ensure that when the traffic starts, the basestation will be able to allocate resources for the flows, while ensuring that even in the worst case of all flows beginning simultaneously, the resulting slice's allocation does not exceed t_g^{rsv}. If a flow in a slice has traffic, α_{gi} is set by the first expression in Equation 3, which is a linear combination of the contribution of the flow's traffic to the total slice's traffic, and the ideal share of the flow (ie. $1/n_g$) within the slice when every flow is active. The contribution of a flow to the slice's total traffic is estimated by monitoring the uplink traffic and computing total per-flow and per-slice slot usage t_{gi} and t_g respectively; the slots used for a packet is given by the ratio of the packet size and the $\mathcal{R}_{gi}$ of the flow.



(a) (b)

Figure 4: (a) Behavior of Algorithm 1 and (b) Setting α_{gi}.

The linear combination in Equation 3 achieves intra-slice flow convergence (See Figure 4(b)). Re-writing the first expression in Equation 3 and substituting it into Line 3 of Algorithm 1, we obtain

$$t_{gi}^{max} = t_g^{rsv} \cdot \left(\frac{t_{gi}}{t_g} + \delta\left(\frac{1}{n_g} - \frac{t_{gi}}{t_g}\right)\right). \quad (4)$$

For a given flow that is *leading* its ideal allocation (i.e. $\frac{t_{gi}}{t_g} > \frac{1}{n_g}$), the higher the value of t_{gi}/t_g the higher is the decrease in t_{gi}^{max} due to the term $\delta(\frac{1}{n_g} - \frac{t_{gi}}{t_g})$, thereby yielding to the lagging flows (i.e. $\frac{t_{gi}}{t_g} < \frac{1}{n_g}$). Similarly for the lagging flows, the lower their current allocation t_{gi}/t_g, the higher is its new setting t_{gi}^{max} due to the same term. Hence, leading flows aggressively backoff and lagging flows aggressively take up resources when active. In the backlogged case in which every flow has traffic, Equation 3 ensures that α_{gi} for each flow converges to $1/n_g$. Equation 4 shows that the convergence speed is controlled by the parameter δ.

The quantity δ also helps *bound* the amount of resource allocation received by every slice. Regardless of whether a slice is active or not, the design choice of α_{gi} in Equation 3 indicates $\sum_{i \in g} \alpha_{gi} = 1$. As a result, for every slice g,

$$\sum_{i \in g} t_{gi}^{max} = t_g^{rsv}.$$

This ensures slice isolation as long as $\sum_g t_g^{rsv} \leq 1$.

3.1.4 Setting γ_{gi}

The setting of α_{gi} ensures that every time the basestation utilization exceeds the threshold $\mathcal{U}_H$, resource allocation is reset to ensure isolation across slices. From this point, the quantity γ_{gi} ensures that any unused resources (due to inactive flows or flows with limited traffic) are incrementally utilized by the active flows. To achieve this, we set γ_{gi} as follows:

$$\gamma_{gi} = \begin{cases} \frac{\beta}{\mathcal{N}} & \text{if } t_{gi} > \rho.t_{gi}^{max} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Firstly, the algorithm should not increment t_{gi}^{max} if the flow is currently not reacting to the previous incre-

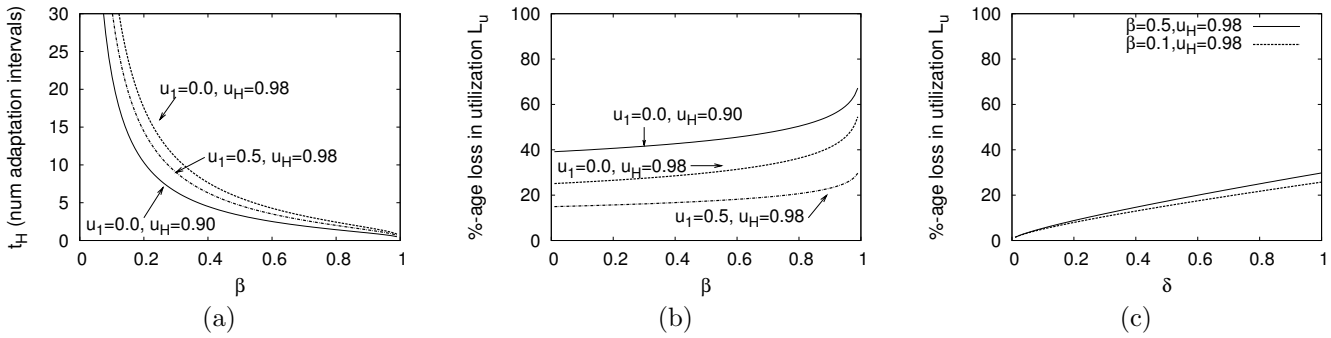


Figure 5: Tradeoff Analysis.

ment, or if it is lagging behind. This can happen either because the application on the client itself does not have traffic to send, or because the intermediate transport protocol (e.g. TCP) is still ramping up. Hence, γ_{gi} is set to zero if the flow's current allocation t_{gi} is less than or equal to a factor ρ of t_{gi}^{max}. This minimizes the violation of isolation if the flow's traffic suddenly surges. Secondly, if the current allocation is greater than ρ of t_{gi}^{max}, the latter quantity is incremented by $\frac{\beta}{\mathcal{N}}$, where $0 < \beta < 1$, and $\mathcal{N}$ is the estimated number of flows in the system across all slices that are actively reacting to increments, i.e. for which $t_{gi} > \rho.t_{gi}^{max}$. Equation 5 ensures that the utilization in the next interval increases by $(1-u)\beta$ if every flow reacts to the increment, where u is the utilization in the current interval.

Note that the increment increases aggressively when u is small, and becomes more conservative as it approaches 1; this behavior attempts to aggressively reduce under-utilization, while conservatively avoiding over-utilization.

Summary: CellSlice's uplink algorithm allows an MNO to balance the three goals of (1) providing slice isolation, (2) maximizing resource utilization, and (3) ensuring intra-slice (flow) convergence. The parameter α_{gi} achieves slice isolation, β opportunistically improves basestation utilization, and δ ensures intra-flow convergence.

3.2 Analysis

Since the algorithm dynamically discovers the appropriate setting for t_{gi}^{max} for each flow using an increment/decrement process, it leads to some loss in both utilization and isolation. The algorithm resets the value of t_{gi}^{max} every time the current utilization exceeds $\mathcal{U}_H$, to ensure isolation across slices; this reset operation, however, *reduces the overall basestation utilization.* Then, the algorithm increases the utilization of the basestation incrementally; hence, if a new flow arrives and *leads to violation of isolation* across the slices, CellSlice does not correct it unless the utilization reaches $\mathcal{U}_H$, and t_{gi}^{max} is reset again. We now derive expressions for the loss in utilization (represented by L_u), and the maximum time t_H before isolation across slices is restored after a violation. Figure 4(a) represents L_u and t_H pictorially.

For analytical tractability, we assume that each flow once active has backlogged traffic (at least till the time the basestation utilization reaches $\mathcal{U}_H$). The worst-case scenario that leads to maximum value of t_H occurs when a new slice becomes active just after a reset operation for the existing slices. For this scenario, we state the following theorem.

Theorem 1. *Under the backlogged traffic model for flows that become active, we have*

$$L_u = \frac{(\mathcal{U}_H - u_1 + \beta(1-\mathcal{U}_H))\log(1-\beta)}{\beta \log\left(\frac{(1-\beta)(1-\mathcal{U}_H)}{1-u_1}\right)} \tag{6}$$

$$\textit{and} \qquad t_H = 1 + \frac{\log\left(\frac{1-\mathcal{U}_H}{1-u_1}\right)}{\log(1-\beta)} \tag{7}$$

where u_1 is the initial basestation utilization when t_{gi}^{max} is reset.

The proof of the theorem is deferred to the Appendix; we instead focus here on the implications of the theorem. We use the analytical expressions to understand the dependence of L_u and t_H on the parameters β and δ, the initial basestation utilization u_1, and the utilization threshold $\mathcal{U}_H$; this understanding helps tune the parameters β and δ appropriately in the system.

Figures 5(a) and 5(b) plot the expressions t_H and L_u for different values of the parameters. Figure 5(a) shows that small values of β lead to significantly increased time for restoring isolation, since isolation, once violated, is restored again only when the basestation utilization reaches the threshold $\mathcal{U}_H$. On the other hand, Figure 5(b) shows that large values of β lead to increased loss in utilization, since the algorithm makes the basestation utilization reach the threshold more frequently, leading to a flip-flop behavior at short timescales. *Hence, the choice of β leads to a tradeoff between isolation and utilization (i.e. increasing β decreases t_H, but increases L_u, and vice versa)*; in practice, the choice of β depends on the isolation requirement of the slices.

The other lines in the graphs show the sensitivity to other parameters: the starting point u_1 and the utilization threshold $\mathcal{U}_H$. As expected, t_H is lower if either staring point is higher or utilization threshold is lower. However, contrary to the above trend, L_u is lower if either starting point is higher or utilization threshold is higher. Since the utilization threshold makes more dramatic impact on L_u, *the graphs suggest maintaining as high a starting point (during the reset) as possible, and as high a utilization threshold as possible.*

The worst case impact of δ on L_u occurs in the following scenario: consider one slice with a large number of flows n_g such that $\frac{\delta}{n_g} \rightarrow 0$ and $t_g^{rsv} = \mathcal{U}_H$. Let one flow be already active such that $\frac{t_{gi}}{t_g} = 1$ and all other flows are inactive. In this case, $u_1 = (1-\delta)\mathcal{U}_H$ using line 3 of Algorithm 1 and Equation 3. We substitute the value of u_1 in Equation 6 to derive L_u, and plot it in Figure 5(c). At low values of δ, the loss in utilization is lower, and is less dependent on β. However, in practice, choosing higher values of δ is desirable because it would ensure quicker intra-slice (inter-flow) convergence. We also demonstrate this trend in simulations later. *Hence, the choice of δ leads to a tradeoff between utilization and intra-slice convergence.*

3.3 Optimization

Although slices make reservations, it is possible that some slices may not have traffic during certain durations. To increase basestation utilization in such durations, CellSlice can redistribute t_g^{rsv} of inactive slices to other slices (i.e., increase u_1) by simply changing the right hand side of line 3 in Algorithm 1 to $c.\alpha_{gi}.t_g^{rsv}$, where c is set as

$$c = \frac{\mathcal{U}_L}{\sum_g t_g^{rsv}.I(t_g > 0)} \quad (8)$$

The indicator function I is 1 if $t_g > 0$ and is 0 otherwise. The parameter $\mathcal{U}_L$ is set such that $\mathcal{U}_L < \mathcal{U}_H$ to ensure that at least one increment with γ_{gi} is required to reach the threshold $\mathcal{U}_H$, which enables inter-flow convergence within a slice under diverse traffic patterns.

Note that this optimization also subsumes the case in which all slices are active, but their cumulative reservation is below the basestation capacity, i.e. $\sum t_g^{rsv} < \mathcal{U}_H$.

3.4 Prototype

Our prototype consists of a Picochip [14] WiMAX basestation (IEEE 802.16e compliant), an Access Service Network (ASN) gateway and several USB [5] clients (See Figure 6). The ASN gateway functionality is deployed on a standard dual-core Linux machine connected to the base station with a 100 Mbps link. We also implement a Slice Manager as part the ASN gateway to configure the slice parameters. As shown in Figure 7, we implement the necessary interfaces between

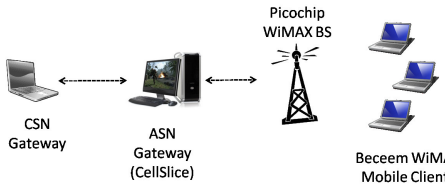


Figure 6: CellSlice **prototype on WiMAX testbed.**

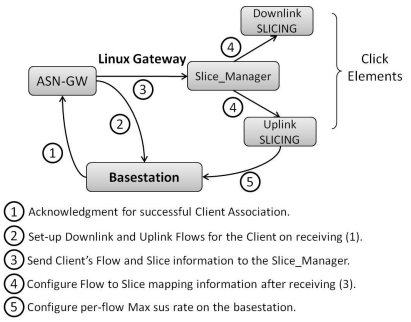


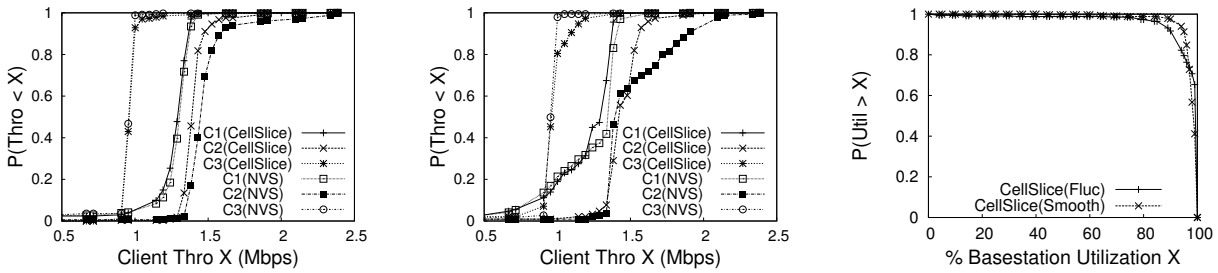
Figure 7: CellSlice **implementation details.**

the Slice Manager, the uplink slicing functionality, the ASN gateway and the basestation. The Slice Manager in our current implementation takes slice information as a configuration file that includes the reservation per slice in terms of percentage of slots. The ASN gateway provides an interface to the basestation 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 uplink and downlink slicing functionalities.

We implement CellSlice's uplink slicing functionality as a user-level Click module [7] that intercepts all data packets from the basestation in the uplink direction. This lets us estimate the average number of slots t_{gi} allocated to each uplink flow. The Picochip basestation provides feedback on the average basestation utilization and the average MCS per client to the CellSlice instance every τ units of time. We set $\tau = 300$ ms, $\beta = 0.5$, $\delta = 0.3$, $\mathcal{U}_H = 0.98$ and $\rho = 0.9$, unless explicitly varied.

Based on the feedback, and the estimates of t_{gi}, CellSlice computes new t_{gi}^{max} for each flow, converts it to r_{gi}^{max} using Equation 1 and sends a message to the basestation containing the r_{gi}^{max} value for each flow. We implemented this message exchange between the uplink slicing functionality and the basestation.

For creating and managing slices in a real deployment, and providing appropriate interfaces to slice owners (to configure slice reservations and monitor resource allocations for the slices), CellSlice requires a more sophisticated management plane. This is an independent piece of work in itself, and is being explored by other research efforts [21].



(a) (b) (c)

Figure 8: (a) Isolation (C1: *Smooth*) (b) Isolation (C1: *Fluc*) and (c) Utilization.

4. EVALUATION

In this section, we evaluate the efficacy of CellSlice's uplink slicing on the WiMAX platform shown in Figure 6. We evaluate CellSlice's efficacy in providing isolation across slices and maximizing wireless resource utilization. To consider diverse and large scale scenarios, we use both prototype evaluation and simulations; we explain the particular setup while describing each experiment. All experiments are done with over-the-air transmissions between the basestation and the clients on the 2.585-2.595 GHz channel (10MHz). We place client laptops in different cubicle locations in the office.

We compare the performance of CellSlice's uplink slicing with NVS [26]—a solution that achieves fine-timescale slicing by modifying basestation's MAC schedulers. We perform similar comparison for the downlink slicing also later in the paper.

4.1 Metrics

Apart from throughput and wireless resource utilization, we use two additional metrics, introduced below.

Slice Satisfaction Index: We use a slice satisfaction index S_I to represent how well the reservations of slices are satisfied (much like Jain's fairness index): $S_I = \frac{(\sum_g x_g)^2}{G\sum_g (x_g)^2}$, where $x_g = \frac{t_g}{t_g^{rsv}}$, and G is the number of slices. The value of S_I varies from $1/G$ to 1 for least satisfaction to maximum satisfaction among all slices respectively. The index is applicable for all slices with $t_g^{rsv} > 0$.

Intra-slice Convergence: For flow convergence within each slice, we compute the Jain's Fairness Index f_g for each slice g in terms of the slots allocated t_{gi} to each flow of the slice: $f_g = \frac{(\sum_i t_{gi})^2}{n_g \sum_i (t_{gi})^2}$.

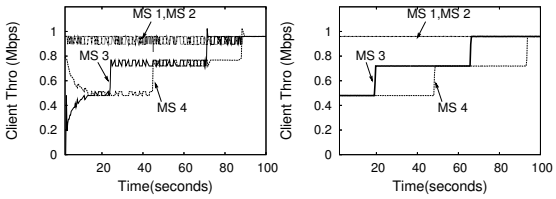
4.2 Prototype Evaluation

We first demonstrate that the prototype provides isolation and maximizes utilization in a small-scale setup; the Picochip platform is a femto-cell basestation, and hence can only support up to 6 clients.

Experiment 1: We set up two slices, the first slice has two clients C1 and C2 and the second slice has one client C3. All clients are placed at similar locations from the basestation such that their MCS is fixed at QPSK-1/2. Slice 1 has a reservation of 75%, whereas slice 2 has a reservation of 25% of the basestation resources. Client C1 streams a VBR video in the uplink direction to a gateway while clients C2 and C3 have backlogged UDP traffic generated by Iperf. The video (*Smooth*) has a peak-to-average frame size ratio of 2. We plot the CDF of the throughput obtained by each of the three clients with CellSlice and compare it with NVS in Figure 8(a). The graph shows that despite fluctuations in traffic due to the VBR video in slice 1, client C3 of slice 2 receives its reserved allocation. However, the throughput of client C2 with CellSlice is slightly lower than the case with NVS. As NVS is implemented within the basestation, it can efficiently allocate resources unused by client C1 to C2 within the same slice at fine timescales.

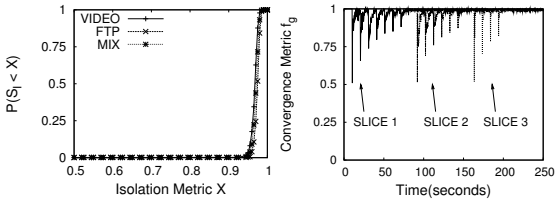
Experiment 2: We repeat the above experiment with a highly fluctuating version of the same video (*Fluc*) ; while the average video rate remains the same, the peak to average frame size ratio is 6. In Figure 8(b), we again plot the CDF of the throughput for all three clients with client C1 streaming the highly fluctuating video. Clearly, NVS is very effective in providing isolation across the slices. However, with CellSlice, client C3 of slice 2 receives greater than its allocation about 20% of the time, even though the client C2 of slice 1 is backlogged. This is a limitation in CellSlice due to the delay in feedback and the uniform increment operation to t_{gi}^{max} in the interest of utilization. Secondly, we observe that C2 achieves lower throughput with CellSlice than with NVS. Again, being a gateway-level design, CellSlice is limited by the delay in feedback, and is also unable to utilize spare capacity due to traffic fluctuations that occur within the feedback interval.

Finally, we plot the complementary CDF of the utilization at the basestation for CellSlice in each of the above experiments. Figure 8(c) shows that the utiliza-



(a) CellSlice (b) NVS

Figure 9: Isolation with capacity changes.



(a) Inter-slice (b) Intra-slice

Figure 10: (a) Slice satisfaction, and (b) Intra-slice convergence with CellSlice.

tion is 90% more than 90% of the time, even with significant traffic fluctuation. The utilization with NVS is 100% in both cases, since slices have backlogged traffic.

Experiment 3: This experiment shows the efficacy of slice isolation with CellSlice, when the basestation capacity changes due to improvement in channel quality for some users. We set up two slices with two clients each. Both slices have a reservation of 50% of the basestation resources. All clients have backlogged UDP traffic generated by Iperf. Both clients MS1 and MS2 of slice 1 transmit continuously at an MCS of 16QAM-1/2. Clients MS3 and MS4 of slice 2 start sending traffic at an MCS of QPSK-1/2. The MCS of client MS3 is upgraded (to QPSK-3/4 followed by 16QAM-1/2) twice at about 20 seconds and 70 seconds respectively. Similarly, the MCS of client MS4 is upgraded at 45 seconds and 90 seconds. We perform a similar experiment with NVS. Figure 9 shows that the throughput of MS1 and MS2 does not change as the MCS of clients changes in the other slice, whereas clients in slice 2 are able to achieve higher throughput from the slots allocated to them, as their MCS improves. The behavior is comparable to that of NVS, except for the small fluctuations caused by the adaptation steps in CellSlice.

4.3 Simulation Study

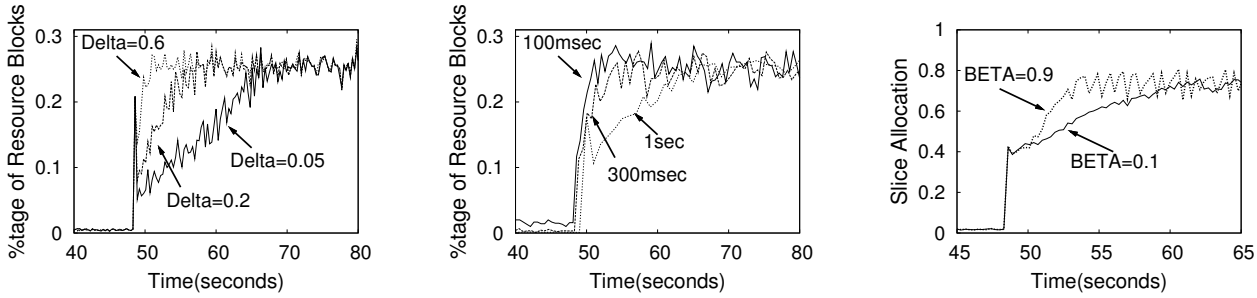
We now consider several larger-scale scenarios that we could not create in the prototype. We simulate these scenarios with an inhouse OFDMA system-level simulator that simulates remote slicing in the same way as it would function in the prototype. The simulator also simulates fast fading on the wireless channel, user arrival and departure, variable amounts of traffic and different types of traffic (Video, FTP and VoIP).

Experiment 4: In this experiment, we show the efficacy of slice isolation with large number of slices and clients. We setup the simulation with 10 slices and 20 clients. The clients are randomly assigned to the different slices; each slice has 1, 2, 3 or 4 clients. We run the simulation for a mix of traffic. The first run involves all clients running FTP traffic, the second run involves all clients running video traffic, and the third run includes a mix of FTP, video and VoIP flows. For each run, we compute the CDF of the slice satisfaction index S_I (computed every second) achieved by CellSlice. Figure 10(a) shows the CDF, and indicates that CellSlice is effective in allocating resources to the slices proportional to their reservation in the presence of multiple clients with fluctuating traffic of different types.

Observe that CellSlice cannot identify the resources allocated by the basestation for retries (HARQ) due to fluctuating channel conditions. However, Figure 10(a) shows that this is not too much of a concern since well-implemented HARQ and rate adaptation within the basestation minimizes the retries.

Experiment 5: We now demonstrate the convergence of resource allocation across flows within each slice with CellSlice even with their dynamic arrival. In this experiment, we consider three slices with 8, 7 and 5 clients. Every 10 seconds, an FTP session for a client is started (beginning with all clients of the first slice, followed by the other slices). We measure the intra-slice convergence metric f_g every second and plot it in Figure 10(b). As shown in the figure, f_g drops when a new flow arrives, but converges to 1 soon after. As expected, the drop is higher when the slice has fewer clients.

Experiment 6: To explore the effect of δ on flow convergence, we consider two slices with 2 and 3 clients respectively. Slice 1 and 2 have a reservation of 20% and 80% respectively. We initiate video traffic for each client with staggered start time. We only plot the allocation achieved by the 5th client in Figure 11(a). As the figure shows, a higher delta results in quicker convergence of flows within a slice. However, recall from the analysis (Section 3.2) that higher value of δ results in higher loss in utilization. We also repeat the above experiment to show the sensitivity of flow convergence to the feedback interval τ from the basestation. Figure 11(b) shows that with coarser feedback interval, flow convergence takes much longer. Also, CellSlice's uplink slicing works effectively only for fluctuations at a coarser timescale than the feedback interval τ; the shorter the feedback interval the better is CellSlice's effectiveness. However,



(a) δ (b) Feedback Intvl τ (c) β

Figure 11: Sensitivity analysis.

shorter feedback interval increases the messaging overhead between the basestations and gateways; this exposes a tradeoff for exploration by the MNO.

Experiment 7: Finally, we explore the sensitivity to β. We consider three slices with 1, 2 and 22 clients respectively. The flows of slice 1 and 2 run video. Slice 3 has 12 video flows and 10 VoIP flows. The resource reservations for slice 1, 2 and 3 are 10%,10% and 80% respectively. Traffic for clients of slice 2 is introduced at 25 seconds and traffic for clients of slice 3 is started at around 50 seconds. When slice 3 starts, slices 1 and 2 already receive higher than their reserved resources; the allocation is reset only when the utilization reaches $\mathcal{U}_H$, which depends on β. We plot in Figure 11(c) the resource allocation for slice 3 for $\beta = 0.1$ and $\beta = 0.9$. The graph shows that β=0.1 results in a much longer time to converge to the slice's reservation once it becomes active, although β=0.1 results in higher utilization (Section 3.2).

5. DOWNLINK SLICING

We now focus on achieving efficient remote downlink slicing that is robust to fluctuating traffic, user arrival and departure, and fluctuating availability of wireless resources. Unlike the uplink direction, all traffic is available at the gateway in the downlink direction, and hence can be scheduled appropriately to satisfy slice specific reservations. To achieve this, we borrow the idea of NVS [26], which includes a hierarchical packet scheduler within the basestation for scheduling slices first to satisfy their reservations, and then schedule flows within each slice to achieve flow convergence according to the slice-specified policy. This idea can also be readily implemented within CellSlice at the gateway. However, the challenge that comes with borrowing NVS's idea is that of avoiding queue buildup at the basestation; such queue buildup can override the scheduling decisions taken by CellSlice at the gateway.

5.1 Design

To avoid queue buildup at the basestation, we design

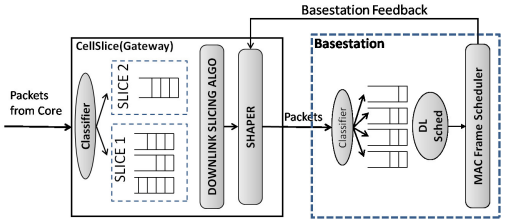


Figure 12: CellSlice's **Downlink Slicing.**

an adaptive shaper that estimates the effective resource slots available for downlink traffic at the basestation, and sends only enough amount of traffic to match the slots. Figure 12 shows the software architecture of CellSlice's downlink slicing. Most importantly, full control of downlink traffic enables separating functionality between the slice scheduler and the shaper: the slice scheduler focuses on intra-slice convergence and inter-slice isolation, leaving only utilization maximization for the shaper.

Shaper: The shaper is invoked every time the utilization feedback from the basestation is received (i.e. every τ units of time). To estimate the effective available slots for downlink data transmission at the basestation, we consider the following model for the shaper. Let $\tilde{S}_\tau$ be the actual slots available (and not directly known to the gateway) in τ units of time, and S_τ be the shaper's estimate of the slots at any point of time. If u is the utilization of the basestation downlink slots at any instant of time, then

$$\tilde{S}_\tau = \frac{S_\tau}{u} \quad (9)$$

The goal of the shaper now is to increase S_τ to approach $\tilde{S}_\tau$. Unfortunately, we cannot set the new value of S_τ to $\tilde{S}_\tau$ directly, since (1) u is an approximate measure, and depends on the update policy (such as running average, interval-based average, etc.) on the basestation, and hence may lead to over-estimation of slots, (2) once the basestation queue builds-up, the utilization will be equal to 1 (and not >1) and does not account for the

Algorithm 2 Downlink Shaper

```
On utilization feedback from basestation
if (u < U_H && GW-backlogged) then
    S_τ ← S_τ + λ
else if (u == 1) then
    S_τ ← S_τ(1 − δ′)
end if
```

queue buildup, which again results in over-estimation of slots.

To overcome this problem and avoid any oscillations due to over-estimation, we take multiple steps to conservatively approach $\mathcal{U}_H\tilde{S}_\tau$. The approach is shown in Algorithm 2. When the utilization is less than $\mathcal{U}_H$, the algorithm checks if the gateway was backlogged in the previous interval with packets for this basestation (the flag `GW-backlogged` is set to **true** by the slice scheduler if the queues destined for the basestation are not empty by the end of the interval τ). If the gateway was also backlogged, the slots are incremented by λ that is proportional to the deficit in slot estimate:

$$\begin{aligned} \lambda &= (\mathcal{U}_H\tilde{S}_\tau - S_\tau)\beta' \\ &= S_\tau(\frac{\mathcal{U}_H}{u} - 1)\beta' \end{aligned} \quad (10)$$

Not incrementing when the gateway is not backlogged with packets for the basestation, avoids overestimation of S_τ. The parameter λ leads to a similar increment behavior as in the uplink direction: the increment is higher at lower values of u, and becomes negligible as it approaches $\mathcal{U}_H$, in order to avoid exceeding $\mathcal{U}_H$. Hence, the value of S_τ stabilizes at a value close to the actual value $\tilde{S}_\tau$, unless there is a variation in the number of actual available slots itself.

When the utilization is 1 (indicating possible queue buildup), S_τ is reduced multiplicatively to accommodate changes in the number of available slots and rediscover the right value of S_τ. Note that the parameter δ' also leads to a similar decrement behavior as in the uplink direction. Hence, our downlink prototype uses $\beta' = \beta = 0.5$ and $\delta' = \delta = 0.3$, and $\mathcal{U}_H = 0.98$.

Slice Scheduler: The slice scheduler is invoked periodically (multiple times within the time interval τ) to transmit enough packets, such that the number of slots used is S_τ in τ units of time. The choice of the invocation period strikes a tradeoff between the scheduler invocation overhead and the increased queue buildup due to bursty transmission to the basestation. The period is chosen to be 5 ms in our current prototype, to minimize queue buildup and maximize isolation.

Once invoked, the slice scheduler schedules the order of packet transmissions from the different slices, such that the slice reservations are satisfied, and isolation across slices is maintained, while also implementing the slice-specified flow scheduling policy within each slice. Given the reservations t_g^{rsv} for each slice, the slice scheduler continuously (i.e., at per-packet granularity) computes a weight $w = t_g/t_g^{rsv}$ for each slice, where t_g is a moving average of the current resource allocation in terms of resource slots that the slice g received. The number of slots each packet occupies is computed as the ratio of packet size and the bitrate (corresponding to the MCS) of the link to the user. At each packet scheduling instant, the scheduler selects the slice with the maximum weight w, and then selects a packet from within the slice. Much like NVS, packet scheduling order can be determined by customized flow schedulers within each slice that are specified by the respective slice owners. We do not expand on the flow scheduling functionality here for brevity and lack of innovation over NVS.

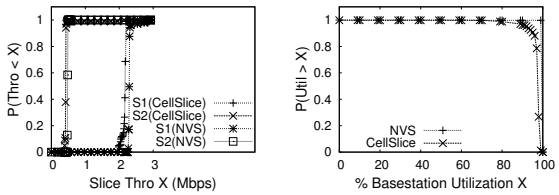
5.2 Prototype Evaluation

We implement a prototype of CellSlice's downlink slicing functionality on the Picochip WiMAX ASN gateway as another user level Click module. All packets entering the gateway from the core are intercepted by this module for downlink scheduling. Figure 6 shows the experimental setup.

Experiment 9: In this experiment, we compare the efficacy of CellSlice to NVS in the downlink direction. We setup two slices such that slice 1 reserves 20% of the downlink resource slots while slice 2 reserves 80%. We run backlogged UDP traffic for the slices. The feedback interval for this experiment is τ =300ms. Figure 13(a) shows the CDF of the per-second throughput for the slices with both CellSlice and NVS. CellSlice causes marginal decrease in throughput compared to NVS for both slices, and each slice receives its respective share of resources in each second even if the other slice is backlogged, hence demonstrating CellSlice's isolation at per-second granularity. Figure 13(b) shows the complementary CDF of the basestation utilization measured for the same experiment, and shows the drop caused because of the shaper. However, the utilization is 90%, more than 95% of the time; we believe that the benefits of remote slicing outweigh this small drop in utilization. Figure 13(c) shows the behavior of the shaper with time, which is as expected from the design. The shaper approaches towards the correct setting of the number of slots in proportion to the deficit in utilization, and hence leads to high basestation utilization.

6. DISCUSSION AND LIMITATIONS

CellSlice's design minimizes the dependence on any one particular access technology to enable it to work for basestations of different technologies like WiMAX, LTE etc. It is also designed as a standalone solution and, for



(a) Isolation (b) Utilization

Figure 13: Comparison with NVS.

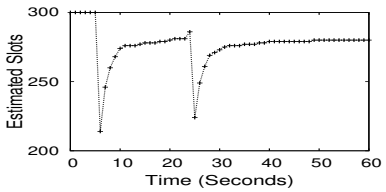


Figure 14: Slots estimation with CellSlice.

instance, can be deployed as either a part of the gateway or a dedicated middlebox between the gateway and the basestation. CellSlice requires minimal support from basestations, which some vendor basestations already provide; such minimal requirements from basestations significantly reduces the barrier for deployability.

With current capacities of basestations (∼10s of Mbps) and the corresponding number of users supported (<200 per basestation), both the feedback from the basestation and the shaping parameter update message to the basestation can be aggregated into a maximum of two TCP packets in each direction every adaptation interval τ. While this is negligible overhead on the basestations, a gateway that handles 100s of basestations requires increased message processing capacity. However, since gateways are usually built with server-grade hardware, we believe that the nominal increased cost is acceptable to equipment vendors for the novel capabilities remote slicing enables.

CellSlice's uplink slicing works effectively only for fluctuations at a coarser timescale than the feedback interval τ; the shorter the feedback interval the better is CellSlice's effectiveness. However, shorter feedback interval increases the messaging overhead between the basestations and gateways. Hence, remote slicing may require significant resource provisioning or may not be as effective in highly fluctuating environments (e.g. with high user mobility). CellSlice also cannot identify the resources allocated by the basestation for retries (HARQ) due to fluctuating channel conditions by just monitoring the traffic, and hence may not be as accurate as NVS; however, if the rate adaptation algorithm within the basestation is well-implemented, the inaccuracy would be insignificant.

CellSlice currently does not support per-slice customization of flow management in the uplink direction. Incorporating such extensions is interesting future work, and would require slice-specific modifications to α_{gi} and γ_{gi}.

The selection of β and δ is currently based on trade-off analysis and intuition. A more systematic solution would require precise formulation of SLAs and utility functions of the MNOs and slice owners, and the penalty functions that represent the penalty for not meeting a slice's reservations or not providing isolation. This is an interesting avenue for future research.

7. CONCLUSION

The relatively high operational costs and low profit margins of cellular networks have traditionally kept them closed from any radical changes beyond providing basic connectivity. We believe that such opaqueness is not sustainable as we move from a voice-dominated to a data-dominated mobile Internet, and that cellular networks will be compelled to support infrastructure for RAN sharing to achieve cost savings, and rapid service innovation and differentiation. CellSlice can be a key component of the such an infrastructure. The primary challenge of remotely slicing wireless resources on a basestation, without modifying the basestation schedulers, is to override the scheduling decisions taken by the basestations in order to impose slice-specific resource allocation. Through a proof-of-concept prototype design and implementation, we showed that it is possible to achieve slicing of wireless resources remotely from gateways with simple algorithms in both uplink and downlink directions.

8. REFERENCES

[1] Telecom policy to allow spectrum sharing. *http://www.indianexpress.com/news/telecom-policy-to-allow-spectrum-sharing/858068/.*

[2] 3GPP TS 23.251: Network sharing; Architecture and functional description. *ftp://ftp.3gpp.org/Specs/html-info/23251.htm.*

[3] 3GPP TS 32.425: Telecommunication management; Performance Management (PM); Performance measurements Evolved Universal Terrestrial Radio Access Network (E-UTRAN) . *http://www.3gpp.org/ftp/Specs/html-info/32425.htm.*

[4] Active RAN Sharing Could Save $60 Billion for Operators. *http://www.cellular-news.com/story/36831.php.*

[5] Beceem WiMAX Chips. *http://www.beceem.com/.*

[6] Bharti adds Huawei alongwith NSN and Ericsson for 3G RAN. *http://articles.economictimes.indiatimes.com/2010-09-21/news/27600378_1_ericsson-india-circles-chinese-equipment-major-huawei.*

[7] Click modular router. *http://read.cs.ucla.edu/click/.*

[8] GENI API Spec. *http://groups.geni.net/geni/attachment/wiki/WiMAX/API_wimax_v1.0-3.pdf.*

[9] Infrastructure Sharing in Practice–Nokia Siemens Networks. *http://www.itu.int/ITU-D/asp/CMS/ASP-CoE/2010/InfraSharing/S12.pdf.*

[10] KDDI selects NEC, Motorola and Samsung for LTE RAN. *http://www.japantoday.com/category/technology/view/kddi-selects-nec-motorola-for-lte-mobile-network* and *http://www.totaltele.com/view.aspx?ID=465683.*

[11] MERU Networks Wireless LAN Virtualization. *http://www.merunetworks.com/technology/wlan/index.php.*

[12] Network Sharing in Asia. *http://www.omnixsoftware.com/uploadedFiles/Resources/Thought_Leadership/090601 - Connect -World - Network Sharing in Asia.pdf*.
[13] Open Virtualized WiMAX Base Station Node for GENI Wide-Area Wireless Deployments. *http://groups.geni.net/geni/wiki/WiMAX*.
[14] Picochip femtocell solutions. *http://www.picochip.com/*.
[15] Vanu Networks. *http://www.vanu.com/*.
[16] VMware Infrastructure in a Cisco Network Environment. *http://www.cisco.com/application/pdf/en/us/guest/netsol/ns304/c649/ccmigration_09186a00807a15d0.pdf*.
[17] Yota Wireless: Russian telcos in LTE network sharing deal. *http://www.yota.ru/en/info/massmedia/details/?ID=284877*.
[18] YouTube in Network Deal Talks With Operators, Manufacturers. *http://www.bloomberg.com/news/2011-06-08/youtube-in-talks-about-network-deal-with-operators-handset-manufacturers.html*.
[19] J. G. Andrews. *Fundamentals of WiMAX*. Prentice Hall, 2008.
[20] G. Bhanage, R. Daya, I. Seskar, and D. Raychaudhuri. VNTS: A Virtual Network Traffic Shaper for Air Time Fairness in 802.16e. In *ICC*, 2010.
[21] G. Bhanage, I. Seskar, R. Mahindra, and D. Raychaudhuri. Virtual basestation: Architecture for an open shared wimax framework. In *ACM SIGCOMM VISA Workshop*, 2010.
[22] G. D. Bhanage, D. Vete, I. Seskar, and D. Raychaudhuri. Splitap: Leveraging wireless network virtualization for flexible sharing of wlans. In *GLOBECOM*, pages 1–6, 2010.
[23] P. Goyal, X. Guo, and H. M. Vin. A hierarchical cpu scheduler for multimedia operating systems. In *OSDI*, 1996.
[24] 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 *ACM CoNEXT*, 2008.
[25] A. Kiiski. Impacts of mvnos on mobile data service market. In *17th European Regional ITS Conf.*, Aug. 2006.
[26] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan. NVS: A Substrate for Virtualizing WiMAX Networks. In *ACM MobiCom.*, Sept 2010.
[27] G. Lenahan. With the right support, mvnos can enrich network operators with innovation differentiation and market share. Telcordia whitepaper: *http://www.telcordia.com/library/whitepapers/mvno_mvne.jsp*.
[28] U. Paul, A. Subramanian, M. Buddhikot, and S. Das. Understanding Traffic Dynamics in Cellular Data Networks. In *IEEE Infocom*, 2011.
[29] S. Sesia, I. Toufik, and M. Baker. *LTE,The UMTS Long Term Evolution: From Theory to Practice*. Wiley Series ISBN: 978-0-470-69716-0.
[30] G. Smith, A. Chaturvedi, A. Mishra, and S. Banerjee. Wireless virtualization on commodity 802.11 hardware. In *WinTECH*, pages 75–82, New York, NY, USA, 2007. ACM.
[31] 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.*, 8(2), 2000.
[32] K. Tan, J. Zhang, J. Fang, H. Liu, Y. Ye, S. Wang, Y. Zhang, H. Wu, W. Wang, and G. M. Voelker. SORA: high performance software radio using general purpose multi-core processors. In *USENIX NSDI*, 2009.
[33] H. Vadada. RAN Sharing Options. *http://www.telecom-cloud.net/2011/03/29/radio-network-sharing-the-new-paradigm/*.
[34] K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown. Stanford OpenRoads Deployment. In *WINTECH*, 2009.

Appendix A: Proof of Theorem 1

Denote u_k as the utilization at the end of the kth adaptation interval. The increment operation (step 5) of Algorithm 1 at the beginning of the kth adaptation interval leads to an aggregate increase in utilization of $(1-u_{k-1})\beta$ by the end of the interval (i.e. when the next feedback of utilization is received), assuming that every flow immediately reacts to the increment in $t^{max}_{g(k-1)}$. Hence, the new utilization in the kth interval can be represented by the recursive equation:

$$u_k = u_{k-1} + (1-u_{k-1})\beta \tag{11}$$

Rewriting this equation, we have

$$\begin{aligned} u_k - 1 &= (u_{k-1}-1)(1-\beta) \\ &= (u_1-1)(1-\beta)^{k-1} \end{aligned} \tag{12}$$

Therefore,

$$\begin{aligned} u_k &= 1-(1-u_1)(1-\beta)^{k-1} \\ \Rightarrow k &= 1+\frac{\log\left(\frac{1-u_k}{1-u_1}\right)}{\log(1-\beta)} \end{aligned} \tag{13}$$

Recall that if isolation is violated at any instant, CellSlice restores isolation only after the utilization reaches $\mathcal{U}_H$. In the worst case (while still assuming that active slices have backlogged traffic), isolation can get violated just after the reset operation (i.e. when the utilization is reset to u_1). From this point, let t_H represents the number of adaptation intervals to restore isolation. Using Equation (13), the number of adaptation intervals required for the basestation to reach $\mathcal{U}_H$ from u_1 is

$$t_H = 1+\frac{\log\left(\frac{1-\mathcal{U}_H}{1-u_1}\right)}{\log(1-\beta)} \tag{14}$$

The average loss in utilization L_u is calculated from the area over the curve in Figure 4(a) as

$$\begin{aligned} L_u &= 1-\frac{1}{t_H}\sum_{k=1}^{t_H} u_k \\ &= 1-\frac{1}{t_H}\sum_{k=1}^{t_H}(1-(1-u_1)(1-\beta)^{k-1}) \\ &= \frac{1-u_1}{t_H}\sum_{k=1}^{t_H}(1-\beta)^{k-1} \\ &= \frac{1-u_1}{t_H}\left(\frac{1-(1-\beta)^{t_H}}{1-(1-\beta)}\right) \end{aligned}$$

Using Equation (14) and simplifying, we obtain

$$L_u = \frac{(\mathcal{U}_H - u_1 + \beta(1-\mathcal{U}_H))\log(1-\beta)}{\beta\log\left(\frac{(1-\beta)(1-\mathcal{U}_H)}{1-u_1}\right)} \tag{15}$$