Hotspot Slicer: Slicing virtualized Home Wi-Fi Networks for Air-Time Guarantee and Traffic Isolation

Sven Zehl, Anatolij Zubow and Adam Wolisz
{zehl, zubow, wolisz}@tkn.tu-berlin.de
Telecommunication Networks Group, Technische Universität Berlin, Germany

***Abstract*—Nowadays, the usage of multiple virtual wireless networks on top of one physical AP is very common in home Wi-Fi networks. Be it Hotspots of Internet Service Providers (ISPs) or Mobile Network Operators (MNOs) used for offloading their traffic, community networks or plain so-called "Guest-Networks". Whereas the home user (AP owner) in the best case should not even be aware that his network connection is shared. Most of the ISPs and MNOs are now trying to convince their customers to install an additional virtual wireless hotspot network on their home AP and offer them in return the free usage of all other available hotspots. But, currently most costumers are skeptical as the providers cannot guarantee a downlink slice of air-time or real separation in time on the wireless access network. In this paper we demonstrate by using a novel downlink slicing scheme applied on commercial off-the-shelf hardware, how slicing on MAC level can be applied to truly guarantee a fixed amount of air-time for the home user and provide complete traffic separation in time between the home and the hotspot network. Moreover, our demonstrator shows the benefits of the approach by comparing the quality of a high definition video stream with and without our MAC slicing approach.**

***Index terms*— 5G, home Wi-Fi networks, Medium Access Control (MAC) layer slicing, smart homes, traffic isolation, Wi-Fi Hotspots, wireless LAN, wireless networks.**

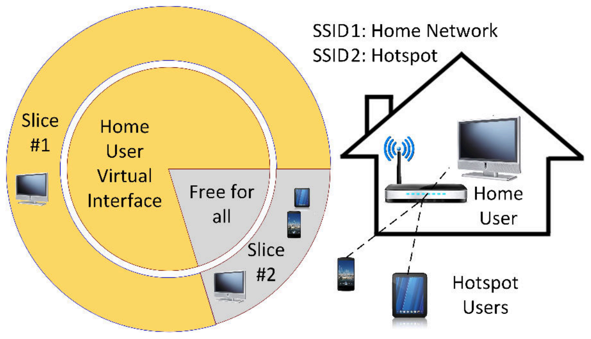


Fig. 1. Hotspot Slicer: Slicing virtualized Home Wi-Fi networks

I. Introduction

Using several virtual wireless networks, identified through different service set identifiers (SSIDs) and basic service set identifiers (BSSIDs), on one physical Wi-Fi Access Point (AP) is very common in today's home Wi-Fi networks. The first virtual network usually provides Internet connectivity for the resident or the owner of the AP. All further virtual networks are for example used for so-called "Guest Networks", Community Networks or Public Hot-spots of the Internet Service Provider (ISP) or a Mobile Network Operator (MNO). Albeit all of them are used by different target groups, all of them are used to provide Internet access to some kind of guest users. Whereas the home user (AP owner) in the best case should not even be aware that his network connection is shared. Generally, this has to be tackled on two points, first on the backhaul network (usually wired) and second on the WiFi access network. While current approaches focused on the back-haul network by applying network slicing [1] or traffic shaping with e.g. the aid of software defined networking (SDN) techniques [2], our approach slices the network directly before CSMA/CA is used to send the traffic over the wireless medium. Slicing the network on MAC level of the access network provides several advantages to efficiently control the resources of the wireless network. First, there is no need for extensive packet-capturing or air-time calculation. Second, even fine grained MAC slicing is possible, and third, real resource isolation in time can be achieved.

In this paper we demonstrate a solution that considers the needs of the residents or home AP owners and the characteristics of today's home wireless access networks to apply a novel slicing technology to guarantee air-time[1] and traffic isolation in time for the downlink of the Wi-Fi devices of the home AP owner. To show the effectiveness of MAC slicing, the demonstrator streams a high quality video to the home device. During the course of the demo, MAC slicing is turned off to show the audience the problem of the current state-of-the-art approach – significant juddering and artifacts even with only two additional hotspot users.

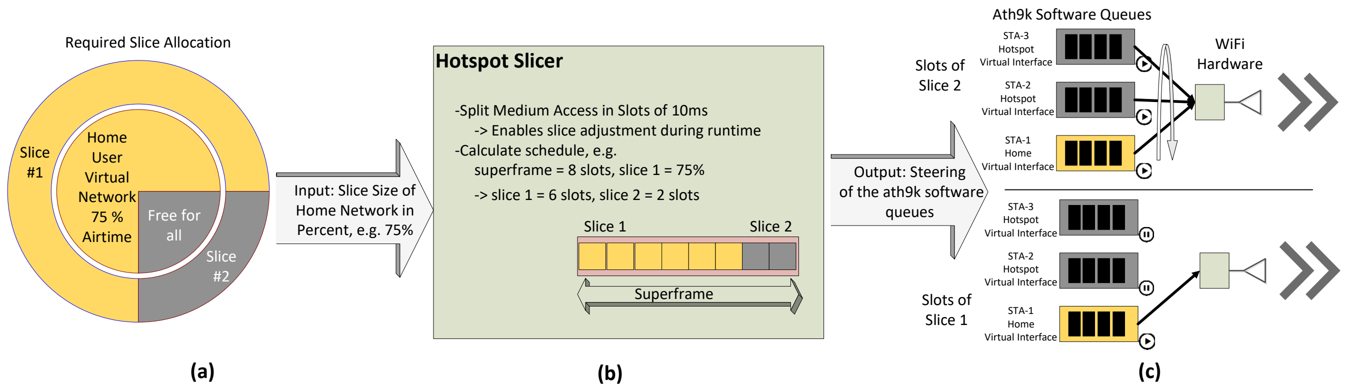
Finally, we provide the full source code of our demonstrator, which is primarily based on a modified version of the popular ath9k driver, to the research community [3].

II. Hotspot Slicer Demonstrator Building Blocks

A. ath9k driver

The ath9k driver is part of the so-called SoftMAC architecture of the Linux Wi-Fi stack. Originally, Wi-Fi MACs were completely implemented as FullMAC devices, in which all

[1]We define the term air-time as the time when the Wi-Fi chip is transmitting data via the wireless channel plus the time needed for medium access.



Required Slice Allocation
Slice #1
Home User Virtual Network 75 % Airtime
Free for all
Slice #2
(a)
Input: Slice Size of Home Network in Percent, e.g. 75%
Hotspot Slicer
-Split Medium Access in Slots of 10ms
-> Enables slice adjustment during runtime
-Calculate schedule, e.g.
superframe = 8 slots, slice 1 = 75%
-> slice 1 = 6 slots, slice 2 = 2 slots
Slice 1
Slice 2
Superframe
(b)
Output: Steering of the ath9k software queues
Ath9k Software Queues
Slots of Slice 2
STA-3 Hotspot Virtual Interface
STA-2 Hotspot Virtual Interface
STA-1 Home Virtual Interface
WiFi Hardware
Slots of Slice 1
STA-3 Hotspot Virtual Interface
STA-2 Hotspot Virtual Interface
STA-1 Home Virtual Interface
(c)

Fig. 2. Hotspot slicer implementation, (a) Input is size in percent of needed air-time for home network (slice 1), (b) medium access is partioned within small slots forming a super frame to enable fine grained slice adjustment, HotSpot slicer calculates needed slots to form slice of home-network (c) slicing is achieved by pausing and unpausing the ath9k software queues directly before the data is handed over to the hardware.

MAC layer functions are controlled by the individual hardware or firmware of the device. In contrast to FullMAC devices in a SoftMAC device the whole 802.11 frame management has to be done by a software module running on the host system.

The Linux Wi-Fi SoftMAC module consists of general hardware independent modules that are shared by all underlying Wi-Fi driver modules (cfg80211 and mac80211) and additional hardware dependent modules such as the ath9k driver module, cf. Fig. 3.

Communication from user-space to the kernel-space is enabled by the use of Netlink sockets and a set of predefined commands called NL80211. All commands are received by cfg80211, which exists as layer between user-space and mac80211. Downwards, the mac80211 module is communicating with the specific hardware driver, e.g. ath9k for Atheros cards or iwlwifi for Intel cards. All the kernel internal communication between cfg80211 and mac80211 and between mac80211 and the hardware driver is realized by callback functions.

B. ath9k Frame Queuing

In order to enable aggregation, block acknowledgement and power saving functionality, the ath9k driver is maintaining software queues for each link and each traffic identifier (TID). TID values between 0 and 7 are priorities that are mapped to the corresponding 802.11e traffic class hardware queues (background, best effort, voice and video). TID values between 8 and 16 are used to implement the 802.11e traffic specification (TSPEC) functionality.

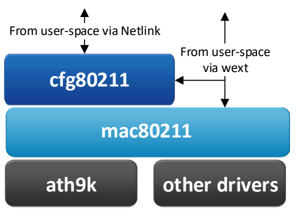


Fig. 3. Mac80211 Structure.

C. IEEE 802.11 Power Saving Functionality

The standard 802.11 power saving mechanism (PSM) enables client STAs to sleep when it has no frames to send and no frames destined to it are waiting to be received. In this doze state, the STA is able to save energy, but is not able to transmit or receiver any frames. If there are frames destined to a sleeping STA operating in infrastructure mode, the AP buffers these frames till the STA indicates that it is awake and ready to receive them. In the opposite direction, a STA is always able to transmit frames destined to the AP.

III. Hotspot Slicer Implementation

To enable slicing of the MAC layer, we utilize the existing ath9k driver power saving implementation to pause and unpause the traffic per link by steering the ath9k software queues, cf. Sec. II-B. As the power saving functionality is running inside the Linux kernel, no modifications to the registers of the Wi-Fi hardware need to be done, which preserves the standard MAC functionality such as CSMA/CA. For this reason, we do not introduce additional unfairness and are fully compliant with the IEEE 802.11 standard.

We implemented the Hotspot Slicer by adding two additional functions to the ath9k frame queuing functionality in the wireless driver similar to [4]. The first function can be used for pausing distinct software queues of the ath9k driver which are identified by the utilized virtual interface plus the receiver MAC address and traffic identifier (TID). The second function enables to continue distinct queues. For calling the functions we utilize the Netlink protocol and extended therefore the standard functionality of nl80211 to enable the call of these two functions from Linux user-space.

For enabling fine granular adjustment of the slice size during runtime, we additionally divide the medium access in small slots of 10ms size. Multiple slots are then combined to form a slice which is then assigned to distinct virtual interface, cf. Fig. 2b.

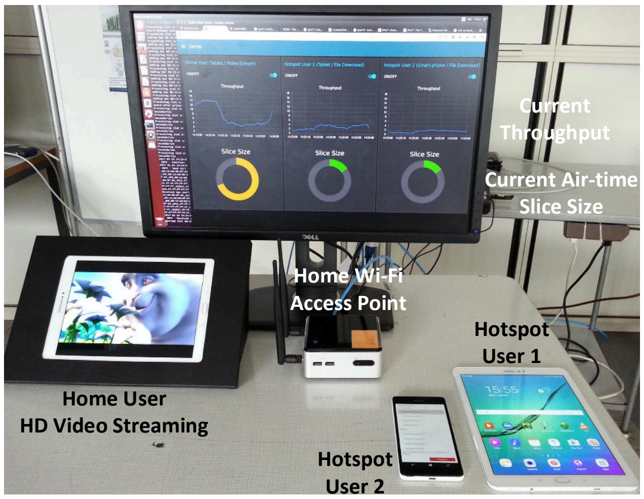


Fig. 4. Demonstrator Setup

Within user-space, a small C++ daemon equipped with a Netlink socket takes care of the slot scheduling. In every slot, the daemon is sending the current slot configuration, e.g. which queues should be paused and which not, directly via Netlink to the ath9k driver. The modified driver then pauses or un-pauses the transmission of frames belonging to the corresponding virtual interface, cf. Fig. 2c. Using this setup we are able to set fixed air-time slots for every single virtual interface and therefore to slice the downlink of the physical AP into as many slices as virtual interfaces are available, e.g. Hotspot ISP1, Hotspot MNO1, Guest-Network1, Home-Network, etc.

Moreover, to enable changing the slice size during runtime, the daemon is also equipped with a ZeroMQ[2] socket that allows sending of new slot schedules which are then applied immediately. With this enhancement we are in addition to guaranteeing air-time to the home network users, able to guarantee a fixed amount of bandwidth by incorporating the physical transmission rate and the busy time of the wireless channel and calculate the required number of slots needed for the slice of the home network to full-fill a required bandwidth. Both values, the currently used physical transmission rate and the time the channel is occupied by other devices, can be collected using the debugFS interface of the ath9k driver during run-time.

For future work, we plan to utilize the MAC slicer together with ResFi [5] to enable traffic separation of neighboring APs to solve hidden node problems.

IV. Demonstrator Setup

The demonstrator setup is shown in Fig. 4 it consists of a home AP represented by Intel NUC hardware with a Linux based system running the modified ath9k implementation described in Section III together with the Linux software AP implementation hostapd. Further, three different unmodified client STAs are used to represent one home device (Android Tablet, used to watch a video stream) and two guest devices (Android Tablet and Android smart-phone downloading data from the Internet). Using the open-source wireless experimenter software Uniflex [6] together with the Node Red visualization software [7] we show the current throughput plus the current size of the slice assigned to the down-link of the corresponding client STA, cf. Fig.4. In the course of the demo, we show the difference between the throughput without MAC slicing in which every client STA independently of their associated virtual Wi-Fi network (Home or Hotspot) will get the same amount of air-time between the case in which MAC layer slicing is applied to achieve air-time guarantee for the home virtual network. The demo mimics a use-case in which an Android tablet used by the AP owner to stream a high resolution video competes for down-link bandwidth with two hot-spot user devices which want to download data from the Internet. The demo shows that without MAC slicing it is impossible to achieve judder and artifact free streaming using the home device, cf. Fig.5(b), after MAC slicing is activated enough bandwidth will be available to enable high quality streaming, cf. Fig.5(a), while the throughput of the guest devices will drop, cf. Fig. 4. Moreover, during the demonstration, the audience will be able to switch client STAs either of the home or the guest network on and off and see the impact on the throughput either when the baseline approach is applied (no slicing) or when air-time is guaranteed to the home network.

Acknowledgment: This work has been supported by the European Union's Horizon 2020 research and innovation program under grant agreement No. 645274 (WiSHFUL project).

[2]http://zeromq.org/

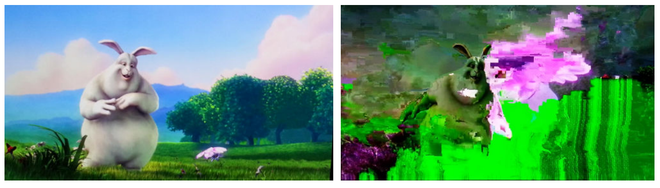


Fig. 5. Screenshot of the home device when (a) MAC layer slicing is applied, and (b) MAC slicing is disabled (baseline)

References

[1] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown, "Slicing home networks," in *Workshop on Home networks*. ACM, 2011.
[2] J. Lee, M. Uddin, J. Tourrilhes, S. Sen, S. Banerjee, M. Arndt, K.-H. Kim, and T. Nadeem, "mesdn: mobile extension of sdn," in *Iinternational workshop on Mobile cloud computing & services*. ACM, 2014.
[3] S. Zehl, "Hotspot slicer," https://github.com/szehl/hotspot-slicer, 2017.
[4] S. Zehl, A. Zubow, and A. Wolisz, "hmac: Enabling hybrid tdma/csma on ieee 802.11 hardware," *arXiv preprint arXiv:1611.05376*, 2016.
[5] S. Zehl, A. Zubow, M. Döring, and A. Wolisz, "ResFi: A Secure Framework for Self Organized Radio Resource Management in Residential WiFi Networks," in *IEEE WoWMoM 2016*, June 2016.
[6] P. Gawlowicz, A. Zubow, M. Chwalisz, and A. Wolisz, "UniFlex: A Framework for Simplifying Wireless Network Control," in *ICC 2017*, May 2017.
[7] N. O'Leary and D. Conway-Jones, "Node red - a visual tool for wiring the internet of things," https://github.com/node-red/node-red, 2017.