A new RAN slicing strategy for multi-tenancy support in a WLAN scenario

Katerina Koutlia*, Anna Umbert*, Roberto Riggio†, Irene Vilà*, Ferran Casadevall*
*Dept. of Signal and Theory Communications (TSC)
Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
Email: {katkoutlia, annau, ferranc}@tsc.upc.edu, irene.vila.munoz@upc.edu
† FBK CREATE-NET, Trento, Italy
Email: rriggio@fbk.eu

***Abstract*—Radio Access Network (RAN) slicing is a key technology, based on Software Defined Networks (SDN) and Network Function Virtualization (NFV), which aims at providing a more efficient utilization of the available network resources and the reduction of the operational costs. In that respect, in this paper a novel virtualized WiFi network hypervisor is presented. This new hypervisor, based on a time variant radio resource sharing mechanism named Weighted Air-Time Deficit Round Robin (WADRR), is able to follow the dynamicity of the traffic variations seen by the different tenants located to the network Access Points (APs). It will be shown that the proposed WADRR hypervisor is able to dynamically assign the appropriate resources per tenant in every AP of the network, according to its specific traffic requirements. To this end, all the network APs are instructed by a controller which is responsible of guaranteeing, on average (long term perspective) and over the whole network, the accomplishment of the tenant's Service Level Agreement (SLA) target, while satisfying the short term traffic requests in the individual network APs. The correct behavior of the proposed algorithm has been validated through both simulations and in a real SDN-NFV platform build upon the 5G-EmPOWER test-bed.**

***Index Terms*—SDN-NFV; Virtualization; Hypervisor; RAN slicing; 5G-EmPOWER testbed;**

I. Introduction

The need for flexible management and efficient operation of the network, while maintaining the capital (CAPEX) and operating (OPEX) expenditures low, has opened the way for new designs and technologies, which will be able to cope with the aforementioned challenges. In that respect, virtualization has been established as a key technology since it allows decoupling software applications from the underlying hardware [1]. SDN and NFV are two promising approaches that when combined can provide the necessary flexibility in the network and resources management [2]. These advantages give us the possibility to share dynamically the available resources and slice the Radio Access Network (RAN) according to the specific requests.

RAN slicing concept is able to cope with the challenges imposed by the continuously increasing traffic demand in wireless networks. Moreover, in most of the cases, traffic varies with time and is not following a homogeneous pattern. As such, an important challenge lies in the fact that each Virtual Network Operator (VNO) or tenant can have different resource requests in a particular Access Point (AP) over time depending on the traffic variations on their network. Network slicing facilitates a cost-effective deployment and operation of multiple logical networks over a common physical network infrastructure, such that each network is customized to best serve the needs of specific applications and/or communications service providers. In this way, each tenant can have its own logically isolated slice of resources with its own desired set of services and a complete control of them. Slicing a RAN becomes particularly challenging due to the inherently shared nature of the radio channel and the potential influence that any transmitter may have on any receiver. In order to guarantee resource isolation between VNOs, a hypervisor must be introduced at the virtualization layer. Usually, a fixed sharing of the available resources between tenants in every AP is assumed, however this fixed sharing is not able to cope with the time-variant nature of the tenant traffic patterns.

One of the challenges of performing RAN Slicing in a Wireless LAN lies on the fact that the available resources are mainly shared in time. For this reason, existing algorithms in the literature, such us Round Robin (RR) or Deficit Round Robin (DRR) [3], cannot be considered as ideal candidates for the implementation of the hypervisor because they manage packets or transmitted bytes, respectively, instead of time. Moreover, to the best of our knowledge different solutions that have been presented in the literature focus in their majority on the adjustment of the Contention Window (CW) with the target to provide fairness among the different slices of the network. In addition, a vast amount of these solutions has been validated only using simulation environments [4]. For instance, in [5] the authors propose an algorithm that is based on the control theory and that adjusts the CW. Though simulations it is demonstrated that the solution offers fairness among the slices, however performance is not always guaranteed. Another approach of adjusting the CW is presented in [6], which is evaluated only through simulations. Moreover, the paper lacks of details related to the CW control mechanism. A real implementation is presented in [7] that apart from the CW it is also adjusted the transmission opportunity for each client. One of the main problems is that isolation is not provably guaranteed. An interesting solution has been presented in [8], where the authors propose a scheduling algorithm known as Air-Time DRR (ADRR) that is based on the principles of the DRR, with major difference that instead of considering bytes for each queue, they consider the Estimated Transmission Time (ETT) of a packet. In this work, we further modify the ADRR and present a new solution, named Weighted ADRR (WADRR). The target of this algorithm is to dynamically adapt the resource allocation (in the downlink direction) in each AP of the network as instructed by a central unit (Controller), with ultimate goal to preserve the agreed Service Level Agreement (SLA) of each tenant when considering all the APs. The logic behind the proposed WADRR algorithm is described in section III.

The contribution of this work is two-folded. First, with the use of a test-bed, we aim at demonstrating that the proposed hypervisor employing the WADRR, can dynamically allocate the available resources as desired depending on the traffic demand of each of the network tenants. Second, it is shown that a simulation environment for the validation and evaluation of the new algorithm can be a useful tool to study the behavior and performance of the proposed solution, before developing it on the real test-bed. In that respect, this work adopts the 5G-EmPOWER test-bed [9] with SDN-NFV capabilities for the implementation of the proposed WiFi network hypervisor. Moreover, it presents a simulator [10] of the 5G-EmPOWER test-bed that allows a fast development and validation of different algorithms with purpose the further implementation of them in the 5G-EmPOWER platform.

The rest of the paper is organized as follows. Section II presents the system architecture of both the test-bed and the simulator. Moreover, a discussion about the framework of the proposed solution is also given. In Section III, the proposed resource sharing algorithm is described in detail. The experimental set up and results, along with an analysis of them, are given in Section IV. Finally, the conclusions and the future work are presented in Section V.

II. System Architecture

The framework on which this work is based is as follows. We aim at considering a scenario where multiple APs are shared by several VNOs (tenants). In this context, every tenant has a SLA agreement with the owner of the infrastructure in order to access a set of wireless resources. The most simple approach is to assign resources (weights) according to the SLAs, in a static manner; that is, all the APs share the resources among the tenants assuming a traffic distribution as agreed in the SLA. However, this simple assumption cannot cope with the asymmetric traffic distribution over the whole network. In this context, it is possible for a given tenant that the traffic to be served by a certain AP exceeds the assumed one that derives by the corresponding SLA, whereas in another AP the traffic of this tenant could be less than the expected. Notice that the resource allocation must guarantee that on average the whole traffic is compliant with the SLA agreement. As such, by considering the same fixed weights in all the APs of the network is not a suitable implementation to manage

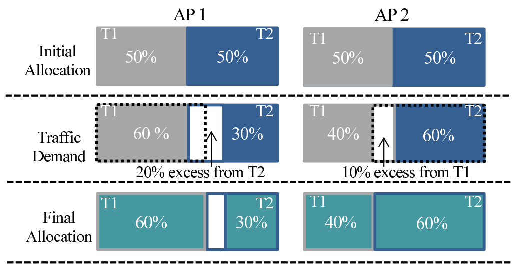

Fig. 1: Resource Allocation Example

the asymmetric traffic distribution usually observed in real networks. Based on the above, what we envisage is to perform a weight allocation following the traffic needs in every AP. That is, for the i^{th} tenant we wish to achieve:

$$\sum_{j=1}^{N} w_{i,j} Rb_j \equiv w_{SLA,i} \sum_{j=1}^{N} Rb_j \tag{1}$$

where N is the number of the network APs in which tenant i owns a slice, $w_{i,j}$ is the weight for tenant i in AP j, Rb_j is the mean throughput of AP j and $w_{SLA,i}$ is the percentage of global resources assigned to tenant i per SLA. Equation (1) must fulfill the constrain that for all the tenants (I) in a given AP:

$$\sum_{i=1}^{I} w_{i,j} \leq 1 \qquad \forall j \in N \tag{2}$$

On the other hand, to cope with the asymmetry and dynamicity of the traffic of the different tenants in every AP, the weight assignment in every AP provided by equations (1) and (2) must be time variant according to the different traffic requirements perceived by the tenants in each AP. Therefore, the first thing we need to achieve is to ensure that resources are shared dynamically according to the weights assigned by the network Controller. In order to clarify the above presented concept, in the following we present a simple example that illustrates the idea.

Let us consider the example of Figure 1, where two phases of the resource utilization are presented, assuming 2 APs and 2 tenants. On the top of the figure, the initial resource allocation that is based on the SLA, is depicted. As it can be seen, in both APs we consider that both tenants have an SLA agreed to 50%. In the middle of the figure, we can see a given point in time wherein the traffic demand presents fluctuations. For AP 1, tenant 1 has 10% more traffic than its SLA agreement, while tenant 2 has 20% less. In AP 2, tenant 1 requires 10% less of its resources and tenant 2 10% more. With a static assignment of weights in the hypervisor's scheduler, the excess/default of resources for every tenant in each AP due to the traffic fluctuations cannot be exploited. However, if we consider a joint weight assignment, by considering all the APs in the network, the resource allocation can be balanced as presented in the bottom of the figure. As it can be seen, tenant 1 gets the

10% that needs in AP 1 from the resources initially assigned to tenant 2, but as a counterpart, it also allows tenant 2 to use 10% of its resources in AP2. So on average both SLAs are guaranteed and the traffic of both tenants in both APs is being served.

Let us notice that the proposed solution targets the adjustment of the resource allocation when there are unused resources in all/some APs of the network. In case where both tenants in the same AP require more resources than the assigned ones (according to their SLA), the allocation will not be changed and traffic will be served as in a regular AP based on the agreed SLAs.

A. *Test-Bed Overview*

In order to check the above presented ideas in a real life scenario, a test-bed has been adopted. The proposed architecture is build upon the 5G-EmPOWER test-bed. The main reason for this choice is because it is implemented following the SDN paradigm and, in addition, provides virtualization functionalities allowing the control and virtualization of the network APs. For this reason the 5G-EmPOWER test-bed constitutes an ideal candidate for the implementation of our slicing approach. It is worth of noting that the code is released under an opensource license on the EmPOWER Website [11], where also the implementation details are included. The system overview of our considered scenario is shown in Figure 2. Let us point out that traffic and consequently the radio resource sharing, is considered per tenant basis.

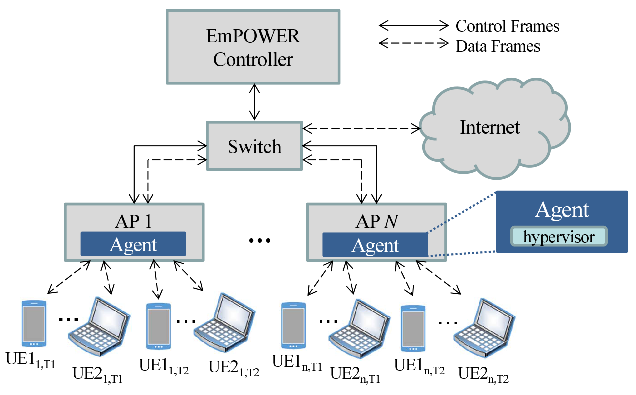

Fig. 2: System Overview

The core of the test-bed is composed by the 5G-EmPOWER Runtime (Controller) and multiple Agents running on each WiFi-AP (see Figure 3). The Controller allows multiple virtual networks (i.e. slices) to be instantiated on top of the same physical infrastructure. Network Apps of a given virtual network are implemented on top of the Controller in their own slice of resources. Each AP is based on the platform PCEngines ALIX and is equipped with two Mikrotik R52Hn wireless cards that use the Linux kernel driver ath9k. Moreover, each AP hosts a radio Agent as presented in Figure 3, which implements an interface for controlling WiFi-specific transmission settings and for gathering wireless related measurement data, such as link and channel statistics, e.g. number of packets transmitted, modulation and coding scheme used through the air interface, RSSI, frame loss ratio, etc. Furthermore, the radio agent also hosts the so-called Light Virtual AP (LVAP) abstraction [12] which simplifies and decouples association/authentication from the physical connection between clients and AP. The key idea behind the LVAP abstraction is to create a virtual AP for every client connected to the AP by assigning a unique BSSID, i.e. every client is given the illusion of having a dedicated AP. In this way, an efficient handover procedure can be achieved by migrating an LVAP between two physical APs, since it is not required to re-associate/re-authenticate. The Agent running within each AP is implemented using the Click Modular Router [13], which is a software implementation of the routing capabilities of the AP and that allows to be reconfigured by modifying already existing packet processing modules (known as elements) or by introducing new ones. The proposed hypervisor is a Click element that belongs to the Agent configuration of each AP.

Finally, it has to be pointed out that the Controller is responsible for sending to the Agent the selected weights, while the hypervisor assigns dynamically the resources in terms of allocated time, based on the received weights. For the time being the weights are set by the Controller according to the setup average traffic values of the tenant traffic sources.

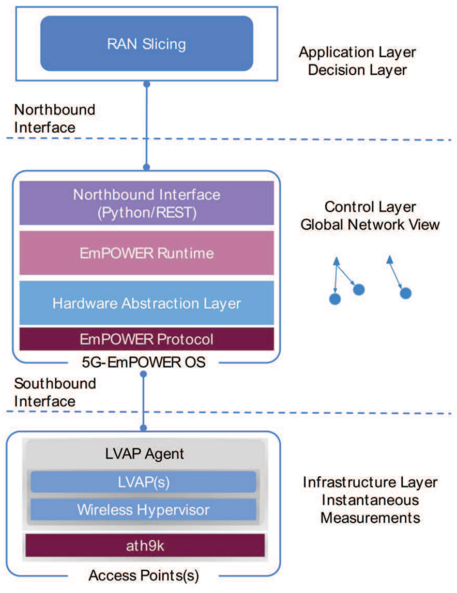

Fig. 3: 5G-EmPOWER Architecture

B. *Simulator Overview*

Since software development in real life systems can be complicated and slow due to the compilation and debugging stages, it can be more convenient to test previously the desired solutions in a simulation environment. Then, the validation and the different adjustments (if necessary) can be performed in a faster way. To this end, a simulator of the 5G-EmPOWER test-bed has been implemented using Python as programming language. In addition, the simulator allows us to study the performance of the proposed strategy in order to be proactive in addressing the issues related to the employed algorithm. Since the functionality of the simulator resembles that of the test-bed, it provides us with an estimation of what can be expected when the hypervisor is running in the test-bed. Let us notice that since our study is focused on the performance of the hypervisor, not all the processing modules of the test-bed have been simulated, however the simulator can be easily extended to cover the needs of each experiment.

The simulator consists of 5 modules: the Controller, the AP, the Tenant, the Tenant-AP and the Channel Model, as presented in Figure 4. In addition, a script known as Scenario is included where the desired parameters of each experiment are set. The functionality of each module is related to the corresponding component of the test-bed. Furthermore, the tenant entity is split into two modules, the Tenant and Tenant-AP, in order to have more flexibility when it comes to the control of them. The Tenant module includes the general definitions of a tenant, e.g. the SLA, while the tenant parameters with respect to a specific AP, such as the weight $w_{i,j}$, are set in the Tenant-AP module. Moreover, let us note that the simulator allows us to generate traffic for each tenant in different modes, i.e. in a deterministic or a random way. Finally, the effective rate in which packets are transmitted through the air interface is selected randomly based on the Minstrel algorithm [14], which takes into account the behavior of the interface including retransmissions.

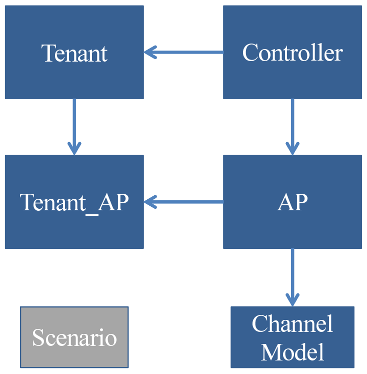

Fig. 4: Simulator Architecture

III. Weighted Air-Time DRR (WADRR)

The ADRR [8] has been proposed with purpose to address the *802.11 performance anomaly* of having a node with poor link quality monopolizing the air-interface. In that respect, ADRR enhances the DRR resource sharing mechanism by considering the Estimated Transmission Time (ETT) of a packet as part of the algorithm. In this work, we further modify the ADRR in order to consider different weights for each tenant's queue that can be dynamically modified depending on the traffic fluctuations. It is worth pointing out that in this context each tenant queue corresponds to a tenant slice. More details about the implementation of the WADRR are given in the following:

Let us consider one AP, since the hypervisor is located at each Agent of each AP. As such, index j denoting the j^{th} AP, is omitted throughout the algorithm description for simplicity purposes. We start with a system quantum (Q_s) that is common for all the I available tenants and corresponds to the time needed to transmit the packet of maximum size at the lowest bit rate. Then, a tenant quantum (Q_i) resulting from the multiplication of Q_s with the agreed weight w_i (that reflects the desired sharing of the resources), is assigned to each tenant (i). Moreover, each tenant is assigned with a Deficit Counter (DC_i), while we consider that packets belonging to each tenant are stored in its corresponding queue.

Algorithm 1 WADRR

```
Initialisation:
 1: for $i = 0$ to $I$ do
 2:     $Q_i = Q_s \times w_i$
 3:     $DC_i = 0$
 4: end for

Enqueuing:
 5: for each arrived packet $p$ do
 6:     $i \leftarrow Queue(index)$
 7:     if ($ExistsInActiveList(i) \leftarrow$ false) then
 8:         $ActiveList.AddToTail(i)$
 9:         $DC_i = 0$
10:     end if
11:     $Queue(i).enqueue(p)$
12: end for

Dequeuing:
13: if ($ActiveList.empty \leftarrow$ false) then
14:     $i \leftarrow ActiveList[0]$
15:     $DC_i = DC_i + Q_i$
16:     while ($DC_i > 0$ and Queue(x).empty $\leftarrow$ false) do
17:         $calculate(t_{i,p})$
18:         if ($t_{i,p} \leq DC_i$) then
19:             $Queue(i).dequeue(p)$
20:             $DC_i = DC_i - t_{i,p}$
21:         else
22:             $break;$
23:         end if
24:     end while
25:     if ($Queue(i).empty \leftarrow$ true) then
26:         $DC_i = 0$
27:         $ActiveList.remove(i)$
28:     else
29:         $ActiveList.remove(i)$
30:         $ActiveList.AddToTail(i)$
31:     end if
32: end if
```

Starting from time $t = 0$, the DC_i of each tenant is initialized to 0. Packets that arrive are stored in the corresponding tenant's $queue(i)$. Then, the algorithm checks for the first tenant if there is a packet at its queue to be transmitted. In case that there is a packet, it sets DC_i equal to the value of Q_i, calculates the time needed for the transmission of it ($t_{i,p}$) and checks if:

$$t_{i,p} \leq DC_i \tag{3}$$

In the case that the above inequality is true, the packet is transmitted and DC_i is set equal to DC_i - $t_{i,p}$. If it is false, the packet is not transmitted. In both cases, in the next iteration DC_i is incremented by Q_i. Let us note that in the case that the packet could not be transmitted, the increment of DC_i by Q_i guarantees that the packet will be transmitted in the next iteration(s). If the tested queue has no packets to transmit, then DC_i is set to zero and the next tenant's queue in line is checked.

Moreover, in order to avoid examining queues that have no packets, we consider a list denoted as *ActiveList* that keeps track of the queues that are not empty. When a packet arrives for a particular queue (corresponding to a particular tenant), the index of the queue is stored to the tail of the *ActiveList*. The algorithm starts its execution (dequeuing process) by serving (according to (3)) the tenant's queue that is located at the head of the list. If at some point in time there are packets to be transmitted, but the DC_i is not enough (i.e. $t_{i,p} > DC_i$), the index of the queue is moved to the tail of the list, otherwise in case that there are no more packets left in the queue, then the index of the queue is removed from the *ActiveList*. The pseudocode of the algorithm is presented in *Algorithm1*. Let us notice that $t_{i,p}$ is computed based on an estimation of the bit rate that is going to be selected by the AP for the transmission of the next packet. This information can be drawn by the rate selection algorithm (Minstrel) that is implemented as a Click element in the Agent.

IV. Simulation Results and Experimental Evaluation

The evaluation of the proposed WADRR algorithm is carried out through comparisons with the WDRR [3]. The later is a variation of the DRR that considers different (weighted) quantum values for each queue.

As a first step, we evaluate the performance of the two scheduling algorithms by means of simulations. Then, in the second part, the algorithms are compared in a real environment using the 5G-EmPOWER test-bed with a system configuration as presented in Figure 2. It has to be noted that the reference scheme (WDRR) was not included in the test-bed's default algorithms, thus it had to be additionally implemented. Moreover, we present a study with respect to the algorithm convergence in both simulation and test-bed environments. The results presented here are the average of 10 experiments assuming the parameters of the 802.11g (see table 1). Finally, all the experiments have been carried out considering 1 AP and 2 tenants with one client each one. Although one can argue that the considered scenario with only one user per tenant does not correspond to a real one, the resource sharing performed by the proposed solution is per tenant basis, thus it is almost equivalent of having all the traffic aggregated to a single user. In particular we consider the (aggregated) traffic that meets the SLA agreement plus (or minus) a certain percentage in order to represent the necessary fluctuations. How packet scheduling is performed for the users of each tenant (e.g. using RR, FIFO, etc.) is out of the scope of this paper.

TABLE I: 802.11g Parameters

DIFS	SIFS	ACK	CWmin	CWmax
28 us	10 us	30 us	31	1023

A. Simulation Results

As already mentioned in section II-B, a software simulator has been implemented in order to allow the faster deployment and validation of algorithms before applying them in the 5G-EmPOWER test-bed. Thus, the use of the simulator allows us to gain knowledge of the expected behavior of the algorithms and also adjust the related parameters in a faster way. In addition, the process of debugging can also be accelerated.

The comparison is carried out for two general cases. In the first case, called "Same Packet Size", we consider that both tenants transmit packets of the same size (1500 bytes), while in the second case known as "Dif. Packet Size", the first tenant transmits packets of 1500 bytes and the second one packets of 500 bytes. The generation of the traffic in the simulator is carried out through a Python class, with an average rate of 10Mbps. Results for both study cases are presented with respect to the percentage of the final resource sharing (that is equivalent to the time allocated in the case of a WLAN) to each tenant for various weight settings assumed (provided by the Controller). Let us note that the system quantum for the WADRR has been set to 3ms, while for the WDRR to the maximum packet size (i.e. 1500 bytes).

Figure 5 depicts the comparison of the percentage of resources allocated to each of the tenants (T1, T2) for the WADRR and the WDRR when transmitting the same packet size, while Figure 6 for different packet sizes. As it can be seen from the figures, in the case of "Same Packet Size" both algorithms assign the transmission time as instructed by the selected weights. This result is as expected, since in this case both algorithms work in a similar way. Certainly, for every tenant, the WDRR algorithm takes into account the amount of the transmitted bytes of each packet, while the WADRR algorithm considers the packet transmission time. Then, when all the packets have the same size both metrics are equivalent and the behavior of both algorithms is the same. However, when studying the Dif. Packet Size case, it is obvious that the different packet size impacts on the transmission time. Therefore, since the reference scheme (WDRR) is based on the packet size, the resource sharing it provides in terms of time does not meet up with the selected weights. Then, the proposed WADRR algorithm significantly outperforms the WDRR in this case. Overall, it can be stated that the proposed hypervisor can adapt the allocation of the available resources according to the selected weights with target to maximize their utilization whenever it is possible. Moreover, it is shown that the simulator can help us study the behavior of the tested algorithms and also equip us with valuable knowledge with respect to the expected results from the test-bed. Therefore, the software simulator can be understood as a powerful tool in order to set-up and test new algorithms before they are deployed in the test-bed.

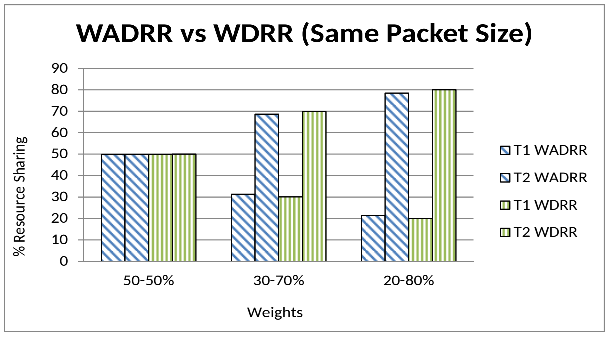

Fig. 5: Simulation Evaluation: WADRR vs WDRR - Same Packet Size

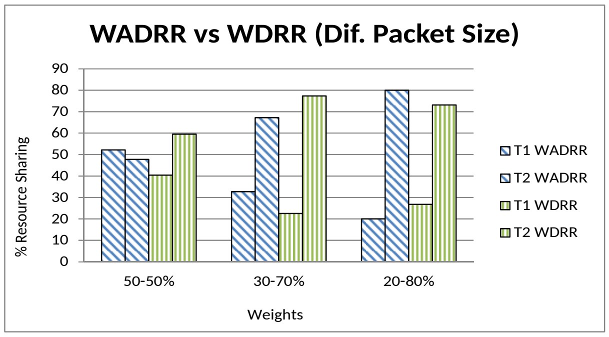

Fig. 6: Simulation Evaluation: WADRR vs WDRR - Dif. Packet Size

B. Test-bed Results

As a next step, the validation and evaluation of the proposed solution has been carried using the 5G-EmPOWER test-bed. Similarly as in the simulator study, WDRR is used as a reference scheme for the comparison with the proposed WADRR algorithm. Let us notice that the same setting of parameters is considered. The traffic for each tenant in this case has been generated with an average rate of 10 Mbps and considering UDP streams using the online traffic generator *iperf* [15]. Figures 7 and 8 depict the comparison of the percentage of resources (time) allocated to each of the tenants for the two algorithms, for the ”Same Packet Size” and ”Dif. Packet Size”, respectively. For the ”Same Packet Size” the proposed WADRR guarantees a resource allocation exactly as the weights instructed by the Controller, while the WDRR presents an acceptable percentage of error (with respect to the selected weights) of approximately 4.5%. For the ”Dif. Packet Size”, this percentage raises up to 17.5%, proving that the way WDRR performs the resource allocation is not always allowing it to meet up with the targeted resource sharing. Considering that the complexity of the algorithms is the same, the WADRR performs better than the WDRR. Overall, it can be stated that the proposed hypervisor can dynamically adapt the resource allocation according to the traffic fluctuations on each tenant, with target to maximize the utilization of the resources.

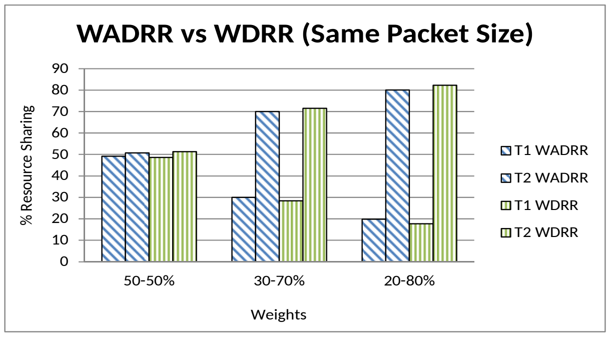

Fig. 7: Test-bed Evaluation: WADRR vs WDRR - Same Packet Size

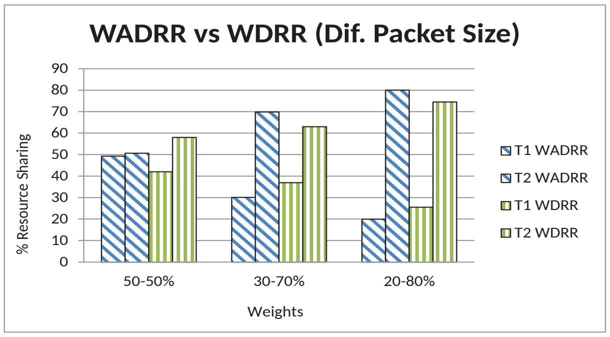

Fig. 8: Test-bed Evaluation: WADRR vs WDRR - Dif. Packet Size

C. Algorithm Convergence

The convergence of the proposed algorithm towards the targeted resource sharing (weights) has been studied for both the simulator and the test-bed. This study is carried out in order to determine the required execution time for each of the experiments, as well as to estimate the frequency with which the algorithm can adapt the resource allocation according to the change of the weights instructed by the Controller. Let us notice, that we consider that the algorithm has reached convergence when the absolute error value of the final weights (resource allocation) assigned to each tenant with respect to the weight values provided by the Controller is below the value of 2.5%.

Figure 9, presents the convergence as resulted of the execution of 10 experiments (averaged). As it can be observed, for the test-bed case the algorithm converges after approximately

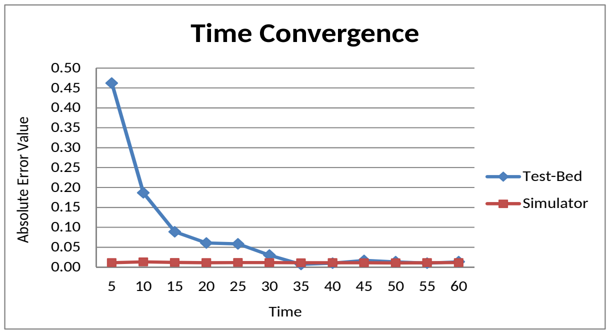

Fig. 9: Convergence Time

30 to 35 seconds, while the simulator converges faster. This behavior results from the fact that the simulator is based on a simplified approach, meaning that it does not include all the processing capabilities as the AP (e.g. complete signaling with the Controller, encapsulation, etc.), therefore it processes the packets faster. On the other hand, having in mind that traffic changes in a network occur in most of the cases in a large scale, it is demonstrated that the proposed algorithm is able to adjust the resource allocation quite quick in order to cope with the tenants' traffic fluctuation perceived for the AP.

V. Conclusions and Future Work

In this paper we have presented a RAN slicing approach based on a virtualized WiFi network hypervisor. The hypervisor employs a novel resource sharing algorithm known as Weighted Air-Time DRR (WADRR) that allocates the available resources in a given AP of the network based on the current traffic demand of each tenant. For the evaluation of the proposed solution, two methods have been used. As a first step, the hypervisor has been tested and validated using a simulation environment. Then, the solution has been implemented in a real test-bed with SDN-NFV capabilities, known as 5G-EmPOWER. Experimental results demonstrate that the proposed solution outperforms classical schemes and is able to allocate dynamically the available resources based on weights that reflect the specific traffic requirements. Moreover, it is shown that the presented simulator can provide developers with a tool for fast implementations and validations, as well as with previous knowledge regarding the behavior of the tested algorithms. Our future work will include more complex scenarios, consisting of multiple APs and tenants. Moreover, we envisage to study scenarios where the traffic will be changed frequently in order to test the adaptability of the proposed algorithm. Finally, it is envisaged to consider the Light Virtual Access Point feature of the 5G-EmPOWER test-bed in order to perform fast handovers with purpose to further improve the network resource utilization.

Acknowledgment

This work has been supported by the Spanish Research Council and FEDER funds under SONAR 5G grant (ref. TEC2017-82651-R) and by the European Union's H2020 Research and Innovation Action under Grant Agreement H2020-ICT-671639 (COHERENT).

References

[1] H. Hawilo, et al, "NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC)," in IEEE Network, vol. 28, no. 6, pp. 18-26, Nov.-Dec. 2014.

[2] Y. Li and M. Chen, "Software-Defined Network Function Virtualization: A Survey," in IEEE Access, vol. 3, no. , pp. 2542-2553, 2015.

[3] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," in IEEE/ACM Transactions on Networking, vol. 4, no. 3, pp. 375-385, Jun 1996.

[4] M. Richart, J. Baliosian, J. Serrat and J. L. Gorricho, "Resource Slicing in Virtual Wireless Networks: A Survey," in IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 462-476, Sept. 2016.

[5] A. Banchs, P. Serrano, P. Patras, M. Natkaniec, "Providing Throughput and fairness guarantees in virtualized wlans through control theory" Mobile Netw. Appl., vol. 17, pp. 435-446, Aug. 2012.

[6] K. Nakauchi, Y. Shoji and N. Nishinaga, "Airtime-based resource control in wireless LANs for wireless network virtualization," Fourth International Conference on Ubiquitous and Future Networks (ICUFN), Phuket, pp. 166-169, 2012.

[7] K. Guo, S. Sanadhya, and T. Woo, "ViFi: virtualizing WLAN using commodity hardware", In Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture (MobiArch '14). ACM, New York, NY, USA, pp. 25-30, 2015

[8] R. Riggio, D. Miorandi and I. Chlamtac, "Airtime Deficit Round Robin (ADRR) packet scheduling algorithm," 2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, Atlanta, GA, pp. 647-652, 2008.

[9] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski and T. Rasheed, "Programming Abstractions for Software-Defined Wireless Networks," in IEEE Transactions on Network and Service Management, vol. 12, no. 2, pp. 146-162, June 2015.

[10] I. Vila Munoz, "Contribution to the Development of a Hypervisor in a Virtualized Mobile Communication Network," Master Thesis, Univesitat politecnica de Catalunya (UPC), July 2017, Available [Online]: https://upcommons.upc.edu/bitstream/handle/2117/109282/tfm_irenevila_final_version.pdf?sequence=1&isAllowed=y

[11] 5G-EmPOWER, Available [Online]: http://empower.create-net.org/

[12] L. Suresh et al., Towards programmable enterprise wlans with odin, in Proc. of ACM HotSDN, 2012.

[13] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, The click modular router, ACM Transactions on Computer Systems (TOCS), vol. 18, no. 3, pp. 263297, 2000.

[14] Minstrel Specification, Available [Online]: https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel

[15] Iperf, Available [Online]: https://iperf.fr/