Virtual 802.11 Wireless Networks with Guaranteed Throughout Sharing

Kostas Katsalis*, Kostas Choumas*, Thanasis Korakis*, Leandros Tassiulas†
*University of Thessaly, Greece
†Yale University, USA
kkatsalis, kohoumas, korakis@uth.gr, leandros.tassiulas@yale.edu

***Abstract*—In this work, we present how programmable data-plane technology (software routers) offers an easy-to-apply mechanism to create virtual wireless networks and support buffering and scheduling decisions. Furthermore, we present a feedback-based buffering mechanism that is able to provide throughput ratio guarantees per virtual network, without requiring any modifications in the 802.11 driver and without relying on statistical knowledge of the workload per virtual network or knowledge regarding the channel conditions. We implement the proposed mechanism in a software router in a 802.11 Access Point and we evaluate its performance in a wireless testbed environment. The methodology and the mechanics developed are generic and with some modifications can be applied to differentiating services for other types of guarantees like delay.**

***Keywords*—Virtual Wireless Networks, Stochastic Control, Queueing theory, Mobile Cloud Networking.**

I. Introduction

In order to face physical infrastructure limitations, resource sharing and virtualization technology can be used to facilitate the creation of virtual wireless infrastructures. Depending on the virtualization approach, technology already exists to create 802.11 virtual wireless networks. Nevertheless, due to the varying channel conditions, the unpredictable behavior of the wireless medium and limitations in the access technology, it is very difficult to provide real time guarantees by means of delay, throughput etc., while at the same time differentiate services effectively between competing virtual networks.

In order to build 802.11 virtual wireless networks, spatial or temporal sharing of the wireless channel, using beamforming techniques in $802.11n$ MIMO systems [1] or tuning techniques of contention window size parameters and transmission opportunity limits in $802.11e$ [2], have been proposed. In contrast, in this work we examine how software routers, like the Click Modular Router [3], can be utilized in native 802.11 Access Points (APs), in order to offer group-based scheduling, where groups of users belong to different virtual networks (VNets). The operations are performed in a layer between the MAC and IP, while the APs operate in infrastructure mode. The software router will be used to implement a stochastic buffering mechanism that we propose and that is able to provide guaranteed service differentiation between the aggregated virtual network (VNet) flows. The differentiation objective is related to specific ratios of transmitted bytes per VNet flow. The problem we are investigating is the following: "guarantee specific percentage of transmitted bytes to every VNet flow, using a policy that is work conserving (never go idle in case of available load) and should not rely on assumptions or predictions regarding the arrival statistics, the packet sizes and the channel conditions".

Our contributions are the following. We present how software routers can be used in native 802.11 APs, without requiring any modifications on the driver and the client side in order to handle VNet flows. In addition we present a buffering policy that guarantees transmitted bytes ratios without relying on complex scheduling policies or admission control or/and differential dropping [4]. The proposed mechanism can be extended to a large number of VNets, without requiring one queue per VNet like in classic scheduling schemes. We evaluate our approach in the NITOS [5] wireless testbed, under real interfering conditions by neighbor transceivers. We note that the proposed solution applies to the downlink, which we target due to its importance in cloud services, like video broadcasting. The proposed mechanism can be applied in conjunction with other QoS mechanisms, e.g. $802.11e$, to provide QoS guarantees besides service differentiation, while a combination of additional control mechanisms that take into consideration the channel conditions can be used to provide throughput optimality [6]. Nevertheless, the primary aspect we focus on this work, is throughput-bytes ratio guarantees between the wireless VNets. The total throughput can be increased if techniques like rate adaptation are used.

The rest of the paper is organized as follows: in Section II we present the motivation for this work, the system model and the problem under consideration; in Section III we describe the differentiation scheme; in Section IV we evaluate the proposed solution in a wireless test-bed environment, while we conclude the paper and present our future plans, in Section V.

II. Motivation & Problem statement

Converged wireless heterogeneous networks with macro, pico base stations, femtocells and Wi-Fi APs, aim to cover the communication needs of billions of subscribers, towards the 5G vision. Projects like CONTENT [7] focus on such converged virtual wireless networks, that explore multi-domain virtualization. In end-to-end architectures, the goal of multi-domain virtualization is to build end-to-end paths from the access network up to the virtualized data center and allow for seamless orchestrated on-demand service provisioning. The motivation for this work comes from such environments, where the end-to-end virtual path terminates not in the backhaul

ethernet network, but to the virtualized wireless 802.11 APs. By enhancing virtualized APs with service differentiation capabilities, an 802.11 network provider can share virtualized resources to the various stakeholders, like Mobile Optical Virtual Network Operators (MOVNOs) (Fig.1).

In order to support such an enhancement, we need a virtualization approach in order to virtualize the APs and policies that are able to differentiate services, while at the same time limit the disturbances that derive from the stochastics of the wireless channel. In this study we adopt the software router solution in order to satisfy the first requirement and closed-loop (feedback-based) control to satisfy the second, where the differentiation objective is related to specific transmitted bytes ratios per VNet. We analyze both of them in the following subsections, making the necessary references to the related work. We begin by stating the problem studied in this work.

A. System Model & Problem statement

A single 802.11 Access Point, is equipped with a single 802.11 wireless interface and serves a number of virtual network flows $\mathcal{V} = \{1, 2, \cdots, V\}$. Let $\mathcal{U}_i = \{1, 2, \cdots, U_i\}$ denote the set of users that are associated with each VNet i. The aggregated traffic from all the users in some VNet defines the corresponding VNet flow. With respect to practical considerations, we assume that the total number of users does not violate the limits that are set by the AP administrator. The packet arrivals for every virtual flow have a general distribution with unknown mean. Furthermore, we assume operation in a completely stochastic environment where interference, collisions and congestions are difficult to model and so the corresponding parameters that could affect our model are considered unknown.

Let $B_i^j(t)$ denote the number of bytes transmitted to user j associated with VNet i. Then, the aggregated downlink traffic $B_i(t)$ of VNet i up to time t equals $B_i(t) = \sum_{j=1}^{|\mathcal{U}_i|} B_i^j(t)$. We also define $B(t) = \sum_{i=1}^{|\mathcal{V}|} B_i(t)$ as the total bytes transmitted by all users and by all VNets until time t and weight w_i as the proportion of $B(t)$ we wish VNet i to receive in the long run, such that $\sum_{i}^{|\mathcal{V}|} w_i = 1$. The problem under examination is to find a policy that in the long run (sufficiently large t) guarantees that every VNet i will have $w_i \cdot B(t)$ bytes transmitted. For example, in the case of two VNets and a single AP, if 10 GB were transmitted until time t, $w_1 = 0.2$ and $w_2 = 0.8$, the policy guarantees that 2 GB have been transmitted for VNet 1 clients and 8 GB for clients of VNet 2, without being affected by the channel conditions or arrival rate variations.

B. Wireless Virtualization approach

In principle, resource virtualization in 802.11 wireless networks can be made in both the physical and MAC layers. In the physical layer, radio resources can be shared and thus virtualized in different ways such as time, space and frequency [8]. Spatial or temporal sharing of the wireless

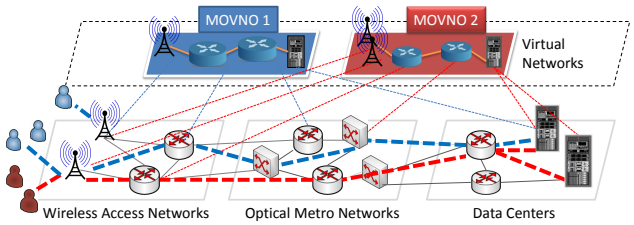


Fig. 1: Virtualized end-to-end Network Architecture

channel using beamforming techniques, has been proposed in 802.11n MIMO systems [1] and tuning techniques of 802.11e in [2]. Custom enqueueing actions that take into consideration the station identification or the traffic type using 802.11e are presented in [9]. Nevertheless, these schemes require enhanced driver capabilities and cannot be applied in native wireless drivers. In the MAC layer, Multi-SSID virtualization is investigated in works like [10] and [11] to group users by assigning them to different Virtual APs (VAPs), where each VAP uses different SSID. By mapping each VNet to a different VAP, different handling per VNet is feasible (eg. authentication, encryption and transmission rate etc). Although this mechanism is widely used, it is prone to increased channel utilization, due to overhead caused by beacon frames that advertise the multiple SSIDs. In addition, regarding the differentiation objective we study, the previous approaches do not take into the account the dynamics of the wireless channel. This means that the actual throughput achieved per VNet, can significantly vary from the actual goal. To face these limitations, we consider a feedback based approach that adapts the system according to runtime performance. We explain this in more detail, in the following. Similarly to [11], where the SplitAP architecture is proposed, we utilize software routers [3] to program forwarding operations. However, we focus on operations between MAC and IP layers, our focus stays on the downlink and our objective is not to allocate fairly the airtime usage to every VNet, but to provide specific throughput ratios by means of transmitted bytes.

In this work, we distinguish the traffic between different VNets, based on the VLAN identification. In a native 802.11 AP all the VNet flows must enter, prior to transmission, the single FIFO queue implemented in the driver and so be served in a "First come First served" fashion. A software router, like the Click router [3], can be used to implement user-defined queueing structures and sophisticated buffering and scheduling policies, as shown in Fig.2. The Click router is extremely extensible and can be used to perform actions like packet scheduling, traffic shaping, filtering, packet dropping and header rewriting. In the following, we answer why a dynamic control mechanism is required in order to give accurate percentages of transmitted bytes to every VNet flow and we describe the proposed differentiation mechanism.

C. The need for dynamic control

A static approach to handle the VNet flows, would suggest to schedule each flow according to the desired weight in a weighted Round Robin fashion or even probabilistically. In addition, in the case we have knowledge of the arrival process

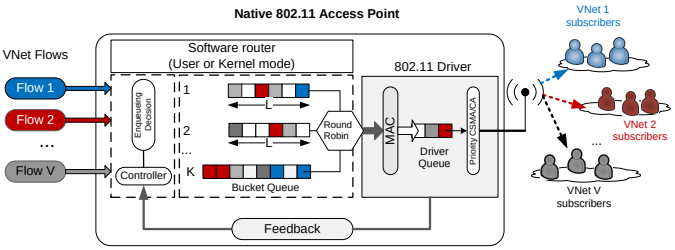


Fig. 2: Virtualized end-to-end Network Architecture

of every VNet, the channel conditions and the transmissions rate of the AP, a model could be constructed to find the appropriate scheduling weights. Token principles and Weighted Fair Queueing (WFQ) [12] have been used extensively, to provide throughput guarantees through scheduling; in [13] rate allocations across wireless and wired devices in a weighted fair manner are presented; and to achieve time fairness, the packet release rate from the network layer to MAC layer is adjusted based on the queue length in [14]. Nevertheless, this is not the case if we have no statistical knowledge of the workload per VNet, the channel and interfering conditions and statistical knowledge about packet drops. The reason is that without any modification of the Distributed Control Function (DCF) mode of the 802.11, the AP and the users that are associated with it, are competing for the medium between them and between neighbor APs that operate in the same channel. If the scheduler assigns weight statically to some VNet, but as opposed to the users of other VNets, its customers face increased collisions and packet drops or/and the distribution of its workload significantly varies, then the requested percentage cannot be achieved.

Our approach in this work is to use closed-loop stochastic control in order to achieve the differentiation objective. The reason is that we want our system to respond in real time in the varying channel conditions and interference. In addition, we don't want to reside on knowledge regarding the arrival or the service process. The proposed control mechanism is aiming to minimize at each control instant, the runtime input error of the difference between the transmitted bytes measurements and the goal percentages. The control decision must be carefully taken, since giving increased priority to some VNet, will clearly affect (decrease) the performance of the other VNets.

III. Differentiation Scheme

The solution proposed in this work, is similar in rationale to the approach developed in [15], where a stochastic buffering mechanism was used to provide service differentiation between a number of competing customer classes in a data center. In this work, we show how this policy can be extended and be used in a completely different environment, with all the restrictions the wireless medium imposes. Also, we stress it under a different objective, namely guaranteed service differentiation, by means of transmitted bytes shares. In contrast to works like [4], the proposed buffering policy, requires no admission control to perform the necessary traffic shaping. The proposed scheme is similar to Stochastic Fair Queueing [16], in the sense that different number of queues are required per active flow, multiple sessions might end up in the same queue and queues are serviced in a Round Robin fashion.

- **The proposed queueing structure**
  A software router is used to implement the following queueing structure in the AP, without affecting the driver. There is a number of queues accessible by all the VNets, where an artificial bound L *is set to their size. In addition, we add one more queue, that all the VNets can also access, which we refer as the* bucket queue *and that we allow to grow to infinity (practically the excess load will be dropped if it violates the system limitations). Is up to the policy to decide in which queue to store an incoming packet; in a* limited size *queue or the bucket queue. All queues are served in Round Robin fashion and the first packet of every queue is forwarded to the single FIFO queue implementation of the driver.* The queueing structure is depicted in Fig.2.
- **Bucket queue with Limited Size Queues (*BLSQ*)**
  For every incoming packet of VNet flow i *that must be enqueued: If the number of transmitted bytes for VNet* i *are less than* $w_i \cdot B(t)$ *(it is below its goal) then the policy uses Join the Shortest Queue (JSQ) [17], between the limited size queues that have not reached the queue limit; in all other cases the packet is forwarded to the bucket queue*. The algorithm is outlined in Algorithm 1.

The intuition behind this algorithm is the following. What will occur after the packet enters the driver's FIFO queue, is based on factors like interference, collisions and congestions that practically are difficult to model. In contrast to complex scheduler implementations (that require changing the weights of a weighted round robin scheduler or changing the probabilities of a probabilistic scheduler, or apply admission control and drop such packets (e.g [18], [4]), by using the proposed scheme with a simple Round Robin scheduler, the required ratio is preserved per flow. Although, this comes with the effect of packet re-ordering, that is not an issue in practical deployments, since (up to some point) it can taken care by higher layer protocols.

The ratio is preserved by indirectly speeding up the VNet clients in the *limited size* queues and by holding back in the bucket queue the VNet flows, which packets contribute in a ratio higher than the one defined in the SLA. This control decision is agnostic to the number of users per VNet, it only requires knowledge of the aggregated traffic served per VNet. We also point that although the decision is made per packet, this decision is agnostic to the packet size distribution, since it only requires knowledge of the bytes transmitted. If a packet arrives from a "suffering" VNet, we enqueue it in the limited size queues, independently of its size.

From a theoretical perspective, we refer to [15] for a queueing analysis of the traffic splitting between the bucket queue and the limited size queues and how stability conditions are related to the feasibility region. The most important features of the algorithm are that no statistical knowledge is required regarding workload dynamics, channel conditions or interference; the operation of the MAC layer and the driver are left unattached and no modifications are required on the client

Algorithm 1 BLSQ - Algorithm Description

w_i : transmitted bytes percentage goal for VNet i
t : enqueueing instant
L : a queue limit
$X_j(t)$: the queue size of queue j at time t (j not the bucket)
$Update\ B_i(t)$ (the measured throughput percentage)
if $B_i(t) < w_i \cdot B(t), \exists j : X_j(t) \leq L$ **then**
    JSQ between the limited size queues
else
    enqueue in the bucket queue
end if

side. Furthermore, the number of queues can be sufficiently smaller than the number of VNets, while its operation is made with a plain Round Robin scheduler.

IV. Benchmarking in a Wireless Testbed Environment

A large set of experiments were conducted in order to present the algorithm's efficiency and demonstrate how system and statistical parameters affect the algorithm performance. Due to page size limitations we present an indicative set of experiments using a single AP; extensions for distributed/centralized operation in a clustered environment are left for future work.

A prototype solution was implemented in a Commell node in the NITOS outdoor wireless testbed [5]. The outdoor NITOS wireless testbed (50 wireless nodes) was selected to demonstrate the algorithms efficiency in realistic conditions, since the testbed operates in an urban area and additionally, multiple experiments run concurrently from other experimenters. Thus, all our experiments faced uncontrollable collisions and interfering conditions from neighbor APs and users. As we show in the following, interference was also created on purpose to further stress the algorithm.

The Commell node selected is equipped with Core 2 Duo 2.26 GHz CPU, 2G DDR3 RAM, two Gigabit network interfaces, Atheros $802.11a/b/g/n$ wireless interfaces, multiband 5dbi and operates both on $2.4Ghz$ and $5Ghz$ antennas. The setup we present in the following, uses a single interface ($802.11a$ in the $5Ghz$ band), while the queueing structure as long as the buffering control mechanism were implemented using the Click Modular Router [3]. The *BLSQ* enabled AP was used to send traffic to users that were logically associated with different VNets, using VLANs. The experimentation model is depicted in Fig.3.

Experiments Parameterization: In every experiment, specific percentage goals were set for every VNet. The arrival process was created with *iperf* (UDP traffic) and the measurements were collected using the *OML library*. The payload was set 1470 bytes for every packet and the AP was tuned to transmit in a physical rate of 12 Mbps (similar results were obtained when auto-rate adaptation was used). In all the experiments we wanted to stress the *BLSQ* algorithm in heavy load conditions, so the arrival rate for every user was set equal to 7 Mbps.

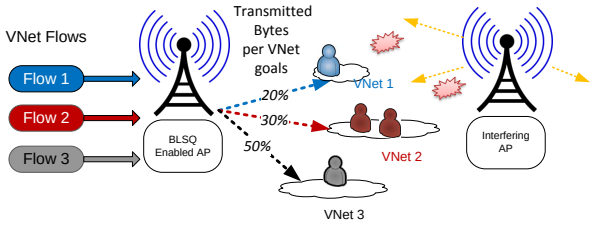


Fig. 3: Experimentation model

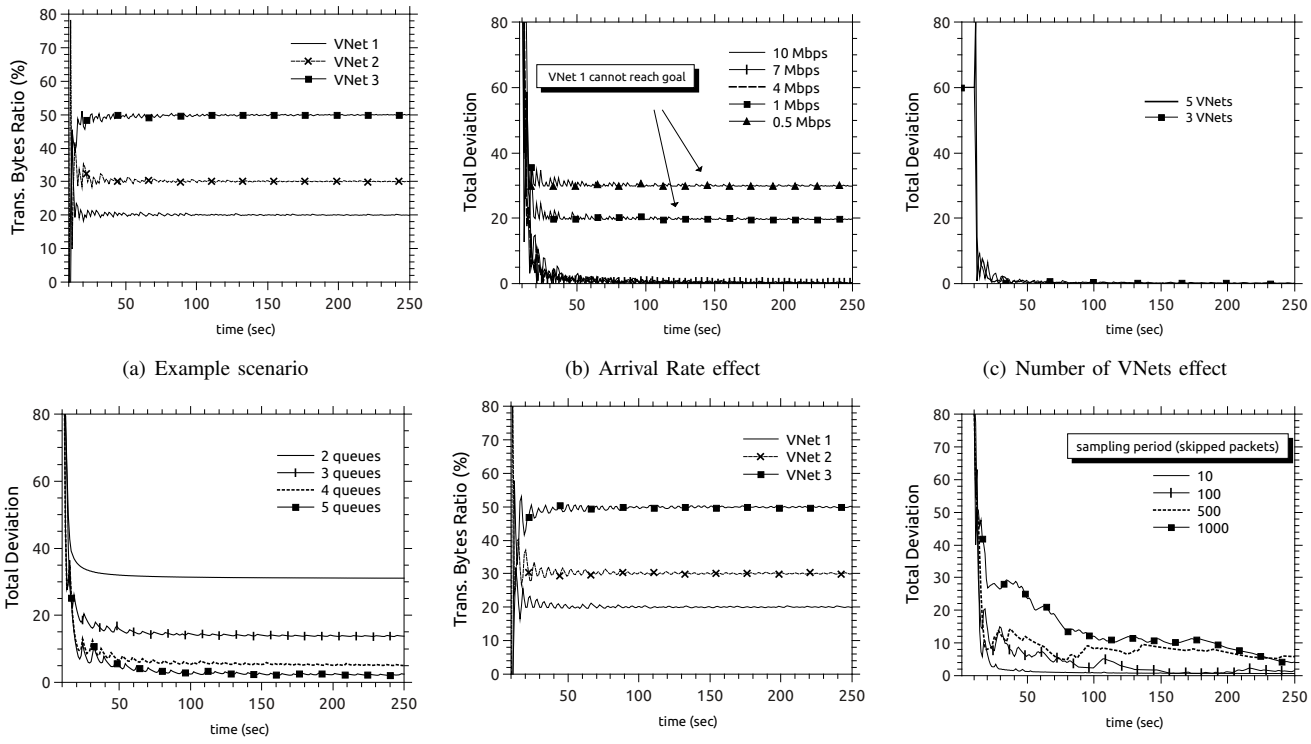
In order to investigate how the system and statistical parameters affect the algorithm performance, we used a basic scenario and each time we varied a parameter to investigate its effects on performance. The basic scenario is the following.

Basic scenario parameters: We used a single AP and we defined 3 VNets; VNet 1 (1 user), VNet 2 (2 users) and VNet 3 (1 user). The VNet goal vector was set $(20\%, 30\%, 50\%)$, while we made statistics updates whenever a packet was send successfully (ACK received). In the software router we used 3 queues in total (the bucket queue plus 2 limited size queues). In all the experiments presented, the bucket queue size was set to the maximum available (1,000,000 packets) and the limit for the limited size queues L was set equal to 100. This number was selected after experiments that presented good performance, but actually the rule of thumb is to select a number that will keep the limited size queues short, in order to be able to *speed up* the VNet we want.

A. Experimental results

1) Basic scenario - Fig.4(a): This is the basic scenario experiment, where the VNet goal vector was set $(20\%, 30\%, 50\%)$. Fig.4(a) is used to demonstrate that the algorithm provides the requested ratio of transmitted bytes per virtual flow. It is interesting to note that VNet 2 serves two users. Since, the mechanism only checks the total number of transmitted bytes per VNet, in order to adapt its control decision, each individual user will receive a percentage proportional of the load transmitted to him, to the load transmitted to all the other users in the same VNet. The number of users per VNet and their transmitting rates, plays no role in convergence as long as their aggregated rate can satisfy a feasible goal. As we present in the following, a goal vector like 98%,1%,1% may not be feasible.

2) Varying the Arrival Rate - Fig.4(b): In Fig.4(b) the total deviation from the goal is depicted ($\sum_{i=1}^{|\mathcal{V}|} |\frac{B_i(t)}{B(t)} - w_i|$). We remind $B(t)$ is the total transmitted bytes and $B_i(t)$ is the ratio VNet i achieved until time t. In this experiment we used the basic configuration, where we varied the arrival rate of the traffic destined to the user of VNet 1, from 0.5 to 10. As we can see in Fig.4(b) the algorithm operation is insensitive to arrival rates variations, as long as the goal is feasible. When the sending rate to user/VNet 1 is very low (eg. 0.5 or 1 Mbps), the goal of 20% is infeasible. Nevertheless, in all other cases the goals are achieved. Similarly, if packet sizes significantly vary, because of the feedback control, the algorithm operation guarantees the requested transmitted bytes ratio.



(a) Example scenario

(b) Arrival Rate effect

(c) Number of VNets effect

(d) Number of queues effect

(e) Interference effect

(f) Sampling effect

Fig. 4: Algorithm performance in different scenarios

3) Varying the number of VNets - Fig.4(c): In this setup we varied the number of VNets from 3 to 5 (one user per VNet), where the goal was to load balance the traffic between the VNets. The number of queues in both cases was set equal to 2, where again we present the total absolute deviation. One of the main features of the algorithm is that is able to differentiate, using a number of queues significantly less than the number of VNets. As we can see in this example, we can perform load balancing between the VNets using only 2 queues (plus the bucket queue). We also stressed the algorithm in different setups with multiple VNets with different goals defined and less queues. Again we report that the feedback mechanics of the algorithm took into the account the dynamics of the system and adapted transmitting rates accordingly.

4) Varying the number of Queues - Fig.4(d): As already discussed, the only requirement for the policy to work is to have a feasible goal defined. For e.g. a goal vector $(98\% - 1\% - 1\%)$ may not be feasible for the policy; this depends both on the queue structure and the system dynamics. One simple way to increase the feasibility space (succeed more *disperse* goals), is to increase the number of limited size queues. The bucket queue policy, gives more opportunities to adjust the "suffering" VNet percentage to a higher value, when increasing the number of the limited size queues. In Fig.4(d) the SLA throughput goal vector was set equal to $\{15\%, 15\%, 70\%\}$ and we varied the number of the limited queues. As we can see, for a small number of queues, the policy fails to meet the objective, nevertheless as the number of limited size queues increases the total deviation decreases. By increasing the number of the limited size queues, we give more opportunities to the "suffering" VNets to increase their percentages in each round.

5) Increasing interference - Fig.4(e): Increasing the number of users, results in increased congestions, increased interference and increased number of collisions. In such environment, in the same time window more retransmissions (MAC originating) take place. We observed these phenomena, by using the same configuration as the basic setup with the three VNets, while adding an additional interfere AP on the same frequency channel, transmitting to a single interfering node in the maximum rate (using Rate adaptation and downlink sending rate 50 Mbps). As we can see the feedback mechanics of the algorithm takes this into the account and adapts transmitting rates accordingly. In Fig.4(e) we see that increased interference results in "slightly" slower convergence, but does not affect convergence itself. This is very important since interference may not have a similar, uniform effect in all users/VNets transmissions. The dynamic feedback-based operation of the algorithm is the one that stabilizes the percentages achieved by each VNet around the desired value.

6) Increasing the statistics update period - Fig.4(f): Implementability enhancements on the proposed algorithm are made based on work presented in [18]. In order to avoid updating the statistics based on all the transmitted packets information, using a sampling technique, we updated the statistics according to a sampled ACKed packet we select periodically, Fig.4(f) presents the effects of increasing the sampling period (in number of skipped packets). This directly correlates with how updated is the information that is used by the mechanism, the

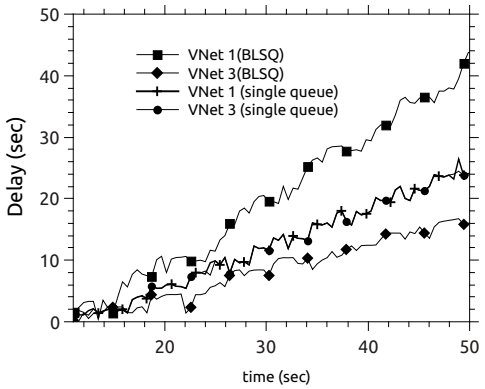


Fig. 5: Delay

time instants a buffering decision is made. As we can see, and is also expected intuitively, there exists a trade-off between the signaling overhead required and the convergence speed.

7) Delay performance - Fig.5: In Fig.5 we use the basic configuration to present the delay performance of every VNet, when the *BLSQ* is used and we compare it with the case where all the requests from all VNets, are entering directly the single FIFO queue of the driver. As expected different VNets experience different delay. Note that the delay in Fig.5 is increasing staggeringly, since total arrival rate (28Mbps) over-exceeds the sending rate ($\sim$ 12Mbps). Nevertheless, these values are not representative for a system that operates in stability region. Using *BLSQ* the delay for VNet 1 (requesting 20% throughput) is increased, in contrast with the native single queue implementation, while the delay for VNet 3 (requesting 50% throughput) is reduced. The reason is that most of the packets from VNet 1 are send from the bucket queue to the driver queue, while traffic to VNet 3 is mostly using the limited size queues, thus on average the delay will be better.

V. Conclusions & Future work

In this work, we presented how software routers offer an easy-to-apply mechanism to build virtual wireless networks. In addition, we presented a buffering mechanism with queue size limitations, that can be used to provide exact throughput percentage guarantees in 802.11 virtual wireless networks. In the proposed scheme less queues are required than the number of virtual networks, while the system operates with a Round Robin scheduler. Future plans include study of the algorithm behavior in an environment with multiple APs and its performance under distributed and centralized control. The proposed scheme is not able to guarantee throughput optimality or QoS, so extensions are planned on these directions also. Delay bounds investigation and better performance regarding jitter will also be part of our future research. A factor that affects jitter and the order of delivered packets is the queue limit we set to the limited size queues. In this direction, we plan to enhance the *BLSQ* algorithm with a packet dropping scheme and congestion marking schemes, while also schemes that can safely drop over-delayed packets.

VI. Acknowledgements

The work of Kostas Katsalis and Thanasis Korakis has been funded by the EU Project 318514 "Convergence of wireless Optical Network and IT rEsourcesiN support of cloud services" (CONTENT). The work of Kostas Choumas has been funded from the EU FP7 Project, under grant agreement 285969 (CODELANCE).

References

[1] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: Practical many-antenna base stations," in *Proceedings of ACM MobiCom*, 2012.

[2] K. Guo, S. Sanadhya, and T. Woo, "ViFi: virtualizing WLAN using commodity hardware," in *Proceedings of ACM MobiArch*, 2014.

[3] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click Modular Router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[4] F. Lu, G. Voelker, and A. Snoeren, "Weighted fair queuing with differential dropping," in *Proceedings of IEEE INFOCOM*, 2012.

[5] K. Pechlivanidou, K. Katsalis, I. Igoumenos, D. Katsaros, T. Korakis, and L. Tassiulas, "NITOS testbed: A cloud based wireless experimentation facility," in *Proceedings of ITC*, 2014.

[6] K. Choumas, T. Korakis, I. Koutsopoulos, and L. Tassiulas, "Implementation and End-to-end Throughput Evaluation of an IEEE 802.11 Compliant Version of the Enhanced-Backpressure Algorithm," in *Proceedings of TRIDENTCOM*, 2012.

[7] K. Katsalis *et al.*, "CONTENT Project: Considerations towards a Cloud-based Internetworking Paradigm," in *Proceedings of IEEE SDN4FNS*, 2013.

[8] G. Smith, A. Chaturvedi, A. Mishra, and S. Banerjee, "Wireless virtualization on commodity 802.11 hardware," in *Proceedings of ACM WiNTECH*, 2007.

[9] K. Choumas, T. Korakis, and L. Tassiulas, "New prioritization schemes for QoS provisioning in 802.11 wireless networks," in *Proceedings of IEEE LANMAN*, 2008.

[10] G. Aljabari and E. Eren, "Virtualization of wireless LAN infrastructures," in *Proceedings of IEEE IDAACS*, 2011.

[11] G. Bhanage, D. Vete, I. Seskar, and D. Raychaudhuri, "SplitAP: Leveraging Wireless Network Virtualization for Flexible Sharing of WLANs," in *Proceedings of IEEE GLOBECOM*, 2010.

[12] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," in *Proceedings of ACM SIGCOMM*, 1989.

[13] C. Gkantsidis, T. Karagiannis, P. Key, B. Radunovic, E. Raftopoulos, and D. Manjunath, "Traffic Management and Resource Allocation in Small Wired/Wireless Networks," in *Proceedings of ACM CoNEXT*, 2009.

[14] M. Zhang, S. Chen, and Y. Jian, "MAC-layer Time Fairness across Multiple Wireless LANs," in *Proceedings of IEEE INFOCOM*, 2010.

[15] K. Katsalis, G. S. Paschos, L. Tassiulas, and Y. Viniotis, "Service differentiation in multitier data centers," in *Proceedings of IEEE ICC*, 2013.

[16] P. McKenney, "Stochastic fairness queueing," in *Proceedings of IEEE INFOCOM*, 1990.

[17] G. Foschini and J. Salz, "A Basic Dynamic Routing Problem and Diffusion," *Communications, IEEE Transactions on*, vol. 26, no. 3, pp. 320–327, 1978.

[18] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker, "Approximate Fairness Through Differential Dropping," in *Proceedings of ACM SIGCOMM*, 2003.