SDN-based Slice Orchestration and MAC Management for QoS delivery in IEEE 802.11 Networks

Pedro Heleno Isolani*, Nelson Cardona*, Carlos Donato*, Johann Marquez-Barja*,
Lisandro Zambenedetti Granville†, Steven Latré*

* University of Antwerp - imec,
IDLab - Department of Mathematics and Computer Science
Sint-Pietersvilet 7, 2000 Antwerp, Belgium
E-mail: {pedro.isolani, nelson.cardona-cardenas, carlos.donato, johann.marquez-barja, steven.latre}@uantwerpen.be

† Federal University of Rio Grande do Sul - UFRGS,
Institute of Informatics - Computer Networks Group
Av. Bento Gonalves, 9500 - Bloco IV - Agronomia 91501-970 - Porto Alegre, RS - Brazil
E-mail: granville@inf.ufrgs.br

***Abstract*—The ever-increasing deployment of new wireless technologies and the demand for mobile services delivery has hampered the efficient and reliable wireless communication. While traditional IEEE 802.11 Wireless Local Area Networks (WLANs) suffer interference from other wireless technologies, their manageability is currently limited. In this context, the overall performance in terms of bitrate, latency, and reliability depends on a number and often dynamically changing aspects where the Medium Access Control (MAC) layer plays a crucial role. Current IEEE 802.11 MAC protocols cannot be programmed fine-grained enough and they cannot manage multiple networks at runtime. In this paper, we propose an approach and an algorithm for on-the-fly End-to-End (E2E) Quality of Service (QoS) slice orchestration and IEEE 802.11 MAC management based on Software-Defined Networking (SDN) principles. We argue that, by performing slice orchestration and IEEE 802.11 MAC management at runtime, it is possible to deliver improved and reliable E2E QoS. To demonstrate the feasibility of our approach, we developed a prototype where our hands-on experiments show that we can reduce the average latency in half while compromising only less than 8% of the average throughput.**

I. Introduction

In addition to the high number of wireless devices and networks, the demand for mobile services delivery has hampered the efficient and reliable wireless communication. In the face of the nowadays evolving and highly heterogeneous wireless environment, different Medium Access Control (MAC) protocols have been proposed to improve wireless connectivity and, hence, to provide Quality of Service (QoS) delivery [1]. Nevertheless, there is no one-size-fits-all solution [2]. Moreover, wireless technologies have to compete for wireless spectrum, multiple services have to coexist within the same wireless infrastructure (*e.g.*, voice over IP, video on demand, sensor monitoring). Users and applications usually have different and dynamic requirements in terms of performance (*e.g.*, bitrate, latency, and reliability) and therefore service-oriented approaches for network resource provisioning are needed.

In traditional IEEE 802.11 networks, Stations (STAs) establish associations with Access Points (APs) mainly based on the measured strongest Received Signal Strength Indicator (RSSI). These associations may lead to uneven and inadequate distribution of STAs across the network, especially when STAs QoS requirements are not considered in the process [3]. The IEEE 802.11e amendment [4] has established the foundations for traffic prioritization through the Enhanced Distributed Channel Access (EDCA) function. EDCA defines traffic categories in which determine how STA access to the channel for a period called Transmission Opportunity (TXOP). A traffic category with more priority has more TXOP than the others and once the STA reaches the contention period, more time is reserved for its transmission. However, EDCA does not ensure radio resource isolation between traffic classes. In addition, it does not provide standard interfaces for End-to-End (E2E) service management at this moment. Therefore, Software-Defined Networking (SDN)-based approaches cannot control it in a centralized manner.

Recently, a few proposals [5] [6] [7] [8] have addressed E2E network slicing for Wireless Local Area Networks (WLANs). On the other hand, only a single proposal [8] has focused on programmable and dynamic E2E network slicing considering resource isolation in a real-world testbed. This isolation has enabled multiple network slices to share the same network resources without performance degradation. In this paper, we propose an SDN-based approach and an algorithm for on-the-fly E2E QoS slice orchestration and IEEE 802.11 MAC management according to application's demand. To the best of our knowledge, there is no approach nor algorithm to provide on-the-fly E2E QoS delivery with reliability for the future IEEE 802.11 architecture.

Our goal is to deliver E2E QoS guarantees at runtime by performing slice orchestration and IEEE 802.11 MAC layer management. Taking this into consideration, we developed a prototype able to: (*i*) perform monitoring with configurable polling intervals specifically focused in metrics related to data channel load and latency, (*ii*) periodic statistics aggregation and processing, and (*iii*) support for slice orchestration and IEEE 802.11 MAC layer reconfiguration according to application's QoS requirements. We conducted our evaluation based on a case study over a real-world testbed. Our results show that our approach and algorithm provide improved E2E QoS reliability both in terms of throughput, latency, and reliability.

II. Related Work

Given that the IEEE 802.11e amendment [4] has established the foundations for traffic prioritization, many investigations have focused on queuing management [9] [10] [11]. The authors have proposed scheduling schemes according to the length of the traffic queues, the lowest time to serve a packet, or the amount of time waiting on the scheduler. After the improvements in radio resource utilization provided by the IEEE 802.11n amendment [12], researchers have been focusing mainly on channel optimization and fairness by modifying or predicting the Aggregated MAC Service Data Unit (A-MSDU) feature behavior [13] [14] [15] [16] [17]. Nevertheless, most of these modifications imply on non-standard compliant solutions.

Other proposals focused on airtime-based resource allocation mechanisms for network virtualization [5] [6] [7]. The focus of airtime scheduling in IEEE 802.11 networks has been extensively studied as a means to overcome the performance anomaly [18]. Nakauchi *et. al.* [5] have proposed an airtime resource allocation method through the management of the Contention Window (CW) size. Bhanage *et. al.* [19] and Katsalis *et. al.* [20] presented similar efforts. The resource allocation problem is tackled by performing traffic shaping to limit the resources usage of each slice and by the use of a queuing model with feedback control to guarantee throughput per slice. Besides those aforementioned, there exist many others in the literature [21]. However, or they are addressed via simulation or cannot ensure slice isolation properly.

Despite the QoS improvements achieved, the aforementioned proposals share the same limitations. First, they are usually designed to very specific use cases and problems. Second, their contributions make their approaches non-standard compliant. Last, but not least, those solutions cannot ensure traffic isolation or the experimentation was conducted through simulations. The most recent and expressive work on providing E2E network slicing while ensuring resource isolation is proposed by Coronado *et al.* [8]. The authors have provided a framework that enables programmable and dynamic E2E network slicing over real-world WLANs. We have enhanced their slicing approach by providing an algorithm besides experimenting it over a real testbed. In this paper, our goal is to demonstrate that our solution is capable to improve QoS according to its requirements, applications demand, and network conditions.

III. E2E QoS Provisioning Approach

In this section, we present how our proposed approach makes use of the benefits of the SDN paradigm to provide IEEE 802.11 network programmability. Then, we explain how our algorithm performs slice orchestration and IEEE 802.11 MAC management based on QoS requirements.

We have added our approach in both SDN *control* and *management* planes. In the *control plane*, we have introduced a network polling mechanism with configurable polling interval in which is responsible for gathering statistics about network slices and maintaining a set of calculated metrics based on a configurable series of measurements (*e.g.*, moving average, moving median, moving standard deviation). In addition, we have added runtime MAC adaptation and slice orchestration support by the definition of transmission policies [22] and the installation of OpenFlow rules [23], respectively. In the *management plane*, on the other hand, we perform the application's QoS requirements analysis and comparison against the statistics gathered from each of the slices monitored. In this manner, they can be orchestrated and the MAC layer can be managed according to application's demand and QoS required (*e.g.*, increase slice's resource allocated on a specific AP, the use A-MSDUs or not for AP's transmissions to certain STAs).

A. Slice Configuration Algorithm

The main idea of our algorithm consists of creating and periodically adjusting the resource allocated for each network slice instantiated on the APs. Each slice is configured according to a specific QoS requirement established from a certain service, application, or STA. We consider two different slices in our approach: QoS and Best Effort (BE) slices. The QoS slice is the one where there are one or more metrics that need to be optimized while the BE slice is the slice that can be modified in case QoS slices have their performance compromised. For the sake of clarification, we have divided the *resource allocation function* and the *slice configuration loop* in different pseudo algorithms.

Algorithm 1 comprises the *resource allocation function* used to adapt the airtime reserved for a given slice on the AP. In other words, it computes the *new quantum* value for a given a slice based on the *old quantum* (Lines 2 to 9), representing the fraction of airtime that can be assigned to that slice in each round. We refer to Coronado et al. for the whole dequeuing process [8]. Our *quantum* adaptation is given by an exponential function where two rates are used as factors. Those rates represent the allocation and releasing factors, given by a_rate and r_rate, respectively. First of all, these rates are initialized and therefore dictate the increase and the decrease rates of the *new quantum* values. The *status* variable (Line 3) it is controlled by the configuration loop (Algorithm 2) and therefore dictates if the *new quantum* has to increase, decrease, or not in the next configuration round.

Algorithm 1 Exponential Quantum Adaptation

Input: $old\_quantum$ ▷ old quantum for given slice
Output: $new\_quantum$ ▷ new quantum for given slice

1: $initializeQuantumAdaptationRates()$
2: **function** QUANTUMADAPTATION($old\_quantum$)
3: **if** $status = allocation$ **then**
4: $new\_quantum \leftarrow 1 + (old\_quantum \times a\_rate)$
5: **else if** $status = release$ **then**
6: $new\_quantum \leftarrow 1 - (old\_quantum \times r\_rate)$
7: **else**
8: **return** $old\_quantum$
9: **return** $new\_quantum$

Algorithm 2 performs the *slice configuration loop* by periodically checking whether the application's QoS requirements are met and triggering slice configurations accordingly.

Algorithm 2 E2E QoS Provisioner

Input: $every$ ▷ loop interval

1: **loop** $every$
2: $apps\_qos\_reqs \leftarrow GetAPPsQoSRequirements()$
3: $slc\_stats \leftarrow GetAllSlicesStats()$
4: **for all** qos_req in $apps\_qos\_reqs$ **do**
5: **if** qos_req is not in $GetQoSSlices()$ **then**
6: $CreateNewSlice(qos\_req)$
7: $qos\_slice \leftarrow GetQoSSlice(qos\_req)$
8: **if** qos_slice in slc_stats **then**
9: $UpdateStatus(qos\_slice, slc\_stats, qos\_req)$
10: **for all** be_slc in $GetBESlices()$ **do**
11: $crr\_q \leftarrow GetCrrQuantum(be\_slc)$
12: $new\_q \leftarrow QuantumAdaptation(crr\_q)$
13: **if** $crr\_q \neq new\_q$ **then**
14: $ChangeQuantum(be\_slc, new\_q)$

First of all, the configuration period is given as input in variable $every$ (Line 1). Next, on every loop iteration, the application's QoS requirements are requested/updated in addition to the statistics about throughput and latency of each network slice (Lines 2 and 3). Then, for all application QoS requirements (Line 4), slices are created and the status of a slice is updated by comparing the QoS requirements with the slice statistics (Line 9). After the $status$ decision variable from Algorithm 1 is updated, another iteration is started but this time over the slices that serves the BE services. Finally, if there is the need for adapting their *quantum* value, a message is sent to the AP in order to reallocate the slice's airtime according to the new configuration provided (Line 14).

IV. CASE-STUDY: LOW-LATENCY QOS DELIVERY

In this section, we depict the use case scenario: a low-latency service coexisting with a BE service within the same infrastructure. Since we do not focus on evaluating our solution in the presence of interference, the chosen case study is in a controlled factory environment. In this context, we demonstrate the coexistence of two different network services with different QoS requirements sharing the same infrastructure. One service is characterized by its low-latency QoS restrictions and the other serves in BE mode. The low-latency service can be associated with remote control applications, emergency systems, and remote control. On the other hand, BE services can be associated with file sharing and e-mail. For those services, we have defined a scenario with three different slice configurations where they have to share and compete with one another. Figure 1 illustrates the scenario with the three slices configurations, A, B, and C.

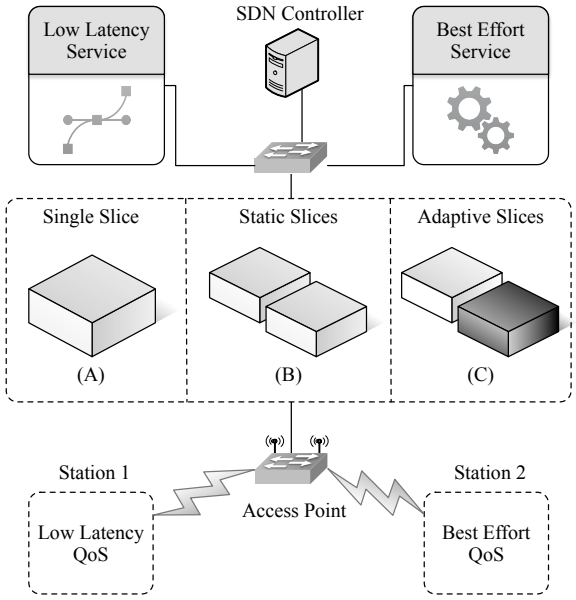


Fig. 1. Evaluation scenarios for the two services with different QoS requirements and their prior slice configurations. Scenario (A), (B), and (C) represent the different slice configurations.

To keep slices configuration accordingly, our algorithm specifies fractions of airtime that are dedicated for each slice to transmit. In all scenarios, we consider having a single AP, however, with different slice configurations. In the first scenario (Figure 1-A) we depict whenever the two services have to compete within the same slice, as in traditional networks without QoS service differentiation. In the second scenario (Figure 1-B), we depict the state-of-the-art network slicing and isolation in WLANs [8]. However, with no feedback about service performance or initial slice bootstrapping and configuration according to the type of service. Therefore, we use static and equal slice configurations. In the third scenario (Figure 1-C), however, we analyze the performance of the two services using our approach and algorithm. In this case, slices are adapted according to the application's QoS requirements and network conditions.

V. Evaluation

In this section, we describe the evaluation of our approach using our developed prototype. We present the methodology, workload, and setup used during the experimentation followed by a discussion of the results obtained.

A. Methodology and Workload

Our prototype has been developed and tested in a realworld testbed composed of one centralized SDN-based controller, one AP, and two STAs as clients. The SDN-based Wi-Fi controller consists of a computer, connected to the wired segment of the network, running both the Ryu[1] and the 5GEmPOWER[2] controllers. The AP is based on the PC Engines APU2D4 (x64) processing board and is equipped with one WiFi card based on the Qualcomm Atheros AR958x 802.11 a/b/g/n wireless network adapter. Linux LEDE 4.4.138 is used as an operating system. We conducted the experiments on the 5 GHz band with the card operating in 802.11n mode and on channel 36 and we ensured that there was no interference in that channel during the experimentation.

Our goal is to analyze services average latency in Round-Trip Time (RTT) and throughput in Mbps according to a given radio resources configuration over the experiment timespan. To understand the impact of changing slice's resource allocation at runtime, we have compared the three slice configurations depicted in our aforementioned scenario in the same channel conditions (see Figure 1). We have also configured different transmission policies according to the application's QoS. We have set that for the low-latency slice, the AP should no wait for Acknowledgments (ACKs) to transmit subsequent packets. Also, whenever the case where there are two slices in our experiment, running the two different services, we have configured not to use A-MSDU for the STA using the QoS enabled slice (Slice 1), since it introduces more throughput, but also delay and packet error rate. We have also initialized the rates for *quantum* for allocation and releasing 5% every 5 loops and 20% at every loop, respectively. Table I summarizes the workload parameters used during experimentation.

During all experiments, we have used standard laptops as our STAs, positioned around 2 to 5 meters away from the AP. In all the scenarios, TCP streams are generated between the SDN controller and STA in order to test our approach in a saturated channel condition. Each experiment runs over 3 minutes long. The traffic generated was done using Iperf3. The results reported in the next subsection are the related average of 10 runs. The controller periodically polls the AP to gather STA/slice utilization statistics and polling period is set to 1 second. It is important to emphasize that the QoS-enabled service is based on the maximum expected latency and latency may suffer from the masking effect in the presence of outliers. Therefore, we have decided to use the median from the last measured latencies as a decision variable to determine if the current latency is achieving the expected QoS or not.

[1]https://osrg.github.io/ryu/

[2]https://github.com/5g-empower/empower-runtime

TABLE I
Workload parameters used during the experimentation

Parameter	Value
Iperf3 Protocol	TCP
Iperf3 bandwidth	40 Mbps (downlink) to STA 2
QoS requirements	10ms max latency for Slice 1 BE behavior for Slice 2
Transmission Policies	do not wait for ACKs for STA 1 wait for ACKs for STA 2
Slice A-MSDU configurations	no A-MSDU for Slice 1 A-MSDU for Slice 2
Slice's initial *quantum*	12 000 µs per round
Slice's configuration interval	every 5 seconds
Quantum adaptation rates	5% for allocation every 5 loops 20% for releasing at every loop
Monitoring polling interval	every 1 second
Measurements subset size	10
Experiment duration	3 minutes

B. Results

In this subsection, we present and discuss the results obtained throughout our experimentation. For the sake of understanding, we present the variations of slices latency, throughput, and slices configurations during a single experiment. For each scenario configuration (Figure 1), we present the moving median and the moving average of the 10 last measurements for both latency and throughput, respectively. Standard deviation bars are placed to depict the variation of the throughput on the BE slice. Because the moving average is skewed by the presence of outliers, we decided to analyze the moving median for the latency measurements. In such cases, the moving median is much more robust because it better represents the central tendency of such distribution. For the third scenario configuration, we have added the updated *quantum* values for both BE slice and QoS enabled slice. Figure 2 presents the quantum values adapted for both BE and QoS slices during the 3 minutes of experiment timespan.

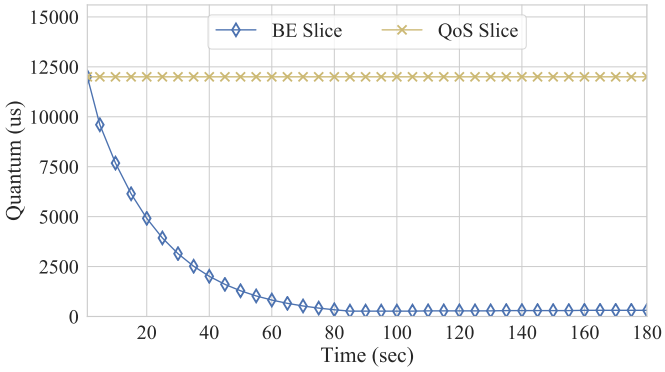
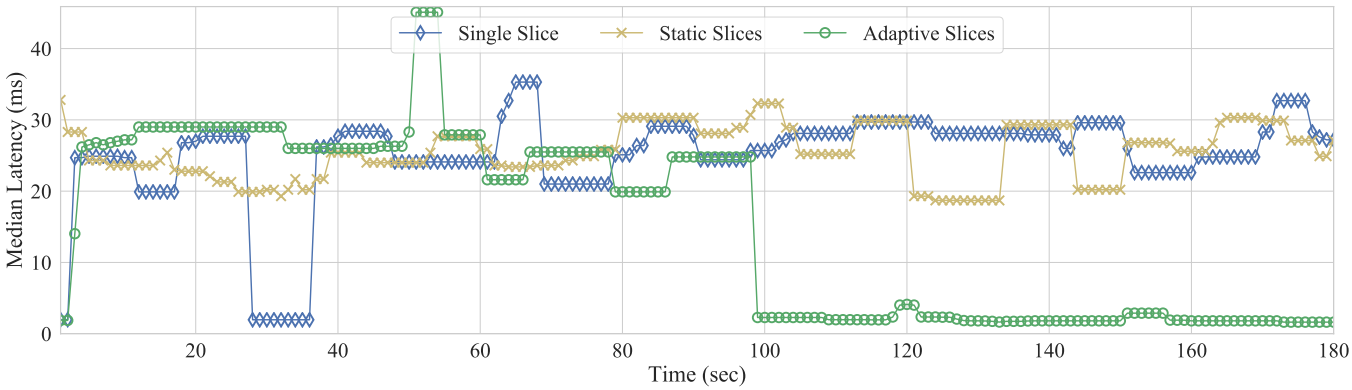
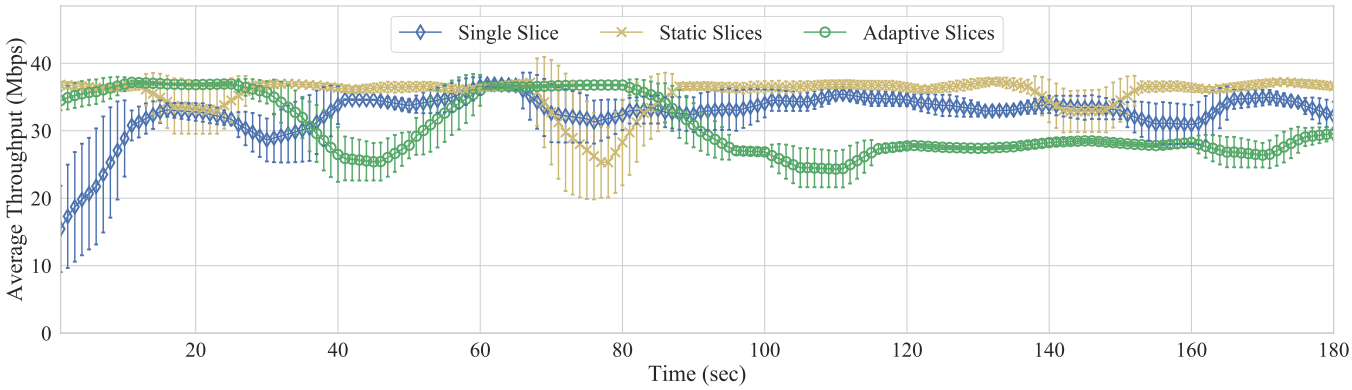


Fig. 2. Quantum values of each slice configuration during the experimentation timespan.

As we can see, our slice configuration algorithm adapts the quantum value of the BE slice in an exponential pace. The curve is dictated by the allocation/releasing rates that are configured to be 20% and 5%, respectively. In our experiment,



(a) Median latency on each scenario configuration for the QoS restricted slice during a single experiment.



(b) Average throughput on each scenario configuration for the BE slice during a single experiment.

Fig. 3. Latency and throughput per slice during the experimentation timespan.

both slices *quantum* values start with 12 000 us each and, after a few seconds of the experimentation, there is a dramatic decrease in the *quantum* value for the BE slice. This occurs because the initial slice configurations do not match with the QoS requirements of the QoS-enabled slice. After 80 seconds of the experiment, the *quantum* value of the BE slice stabilizes at roughly 400 us, implying that the BE slice has to wait for more rounds at the dequeuing process. Consequently, leaving more radio resources to the other coexisting slices on the AP.

Figures 3a and 3b present the median latency and the average throughput along the 3 minutes experiment timespan. At the beginning of the experiment, the moving median remains between 20 ms and 30 ms for all scenario configurations (see Figure 3a). Latency was calculated based on a less than 2 kbps traffic and its peaks may occur due to the dynamic nature of the wireless environment. Although, it is clear that the latency for the QoS slice has been improved in the third scenario configuration, especially after half of the experiment. The behavior is directly related to the *quantum* configurations. After the second 100 of the experiment, around 20 seconds after the *quantum* configuration has stabilized, the median latency of the QoS-enabled slice drops to less than 5 ms. This is followed by a gradual increase on the BE *quantum* and also on the QoS-enabled median latency as attempts to release as many resources to the BE slice as possible. Similar behavior happened with the throughput, but negatively and impacting the BE slice. In the second 100, the throughput from the BE slice decreased by 20% followed by a gradual increase in the throughput as well (5% every 5 loops). Figure 4 illustrates the average throughput and latency of all 10 experiments.

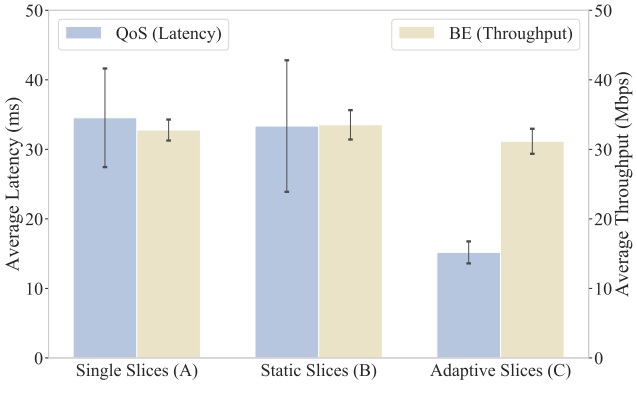


Fig. 4. Average latency and throughput over the 10 experiments for each scenario configuration.

As can be seen, the first and the second scenario configurations have similar performance both in terms of latency and throughput. This happens mainly because the BE slice uses the TCP protocol to transmit its data and therefore the transmission rate is adjusted according to data loss. Besides preventing even more data loss, TCP prevents channel saturation. In the scenario C, on the other hand, our algorithm achieves 47,4% and 54% lower latency while reducing the throughput of the BE slice by only 4,94% and 7,07% in comparison with the same scenarios.

VI. Conclusion

Although network monitoring and configuration are common management activities, these can be considerably different in the context of SDN-based WLANs. We have presented an approach and an algorithm for on-the-fly E2E QoS slice orchestration and IEEE 802.11 MAC management based on SDN principles. Our results show that we can reduce the average latency in half while compromising only less than 8% of the average throughput. More specifically, we achieve 47,4% and 54% lower latency using our algorithm (Scenario 1-C) in comparison with Scenario 1-A and Scenario 1-B. To do such a thing, our algorithm reduces the throughput of the BE slice by only 4,94% and 7,07% in comparison with the same scenarios. As future work, we plan to scale up our experimentation testbed with more slices, QoS metrics, APs and STAs. In addition, we plan to improve our quantum adaptation algorithm to consider the use of handovers and performance estimations, such as Signal-to-Interference-plus-Noise Ratio (SINR) and resource availability. In this manner, we envision an enhanced SDN-based QoS delivery for the future IEEE 802.11 networks.

Acknowledgment

The research leading to these results has received funding from the European Horizon 2020 Programme under grant agreement n^{o}732174 (ORCA project) and the VLAIO SBO project SAMURAI.

References

[1] L. D. Braun and M. Porrmann, "The Comprehensive MAC Taxonomy Database: comatose," Center of Excellence - Cognitive Interaction Technology CITEC, Technische Fakultät AG Kognitronik und Sensorik, Tech. Rep., 2018.

[2] P. H. Isolani, M. Claeys, C. Donato, L. Z. Granville, and S. Latr, "A survey on the programmability of wireless mac protocols," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[3] E. Coronado, J. Villaln, and A. Garrido, "Dynamic aifsn tuning for improving the qos over ieee 802.11 wlans," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Aug 2015, pp. 73–78.

[4] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification. Amendment 7: Medium Access Control (MAC) Quality of Service (QoS), ANSI/IEEE Std 802.11e, LAN/MAN Standards Commit- tee of the IEEE Computer Society Std., 2005.

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

[6] 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*, ser. MobiArch '14. New York, NY, USA: ACM, 2014, pp. 25–30. [Online]. Available: http://doi.acm.org/10.1145/2645892.2645893

[7] M. Richart, J. Baliosian, J. Serrati, J. Gorricho, R. Agero, and N. Agoulmine, "Resource allocation for network slicing in wifi access points," in *2017 13th International Conference on Network and Service Management (CNSM)*, Nov 2017, pp. 1–4.

[8] E. Coronado, R. Riggio, J. Villa1ón, and A. Garrido, "Lasagna: Programming abstractions for end-to-end slicing in software-defined wlans," in *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2018, pp. 14–15.

[9] H. Luo and M. Shyu, "An optimized scheduling scheme to provide quality of service in 802.11e wireless lan," in *2009 11th IEEE International Symposium on Multimedia*, Dec 2009, pp. 651–656.

[10] W. L. Pang, D. Chieng, and N. N. Ahmad, "Adaptive priority sliding admission control and scheduling scheme for dcf and edca wlans," *Wireless Personal Communications*, vol. 70, no. 1, pp. 295–321, May 2013. [Online]. Available: https://doi.org/10.1007/s11277-012-0695-2

[11] E. Charfi, C. Gueguen, L. Chaari, B. Cousin, and L. Kamoun, "Dynamic frame aggregation scheduler for multimedia applications in ieee 802.11n networks," *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 2, p. e2942, 2017.

[12] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 5: Enhancements for Higher Throughput, ANSI/IEEE Std 802.11n, LAN/MAN Standards Committee of the IEEE Computer Society Std., 2009.

[13] M. G. Sarret, J. S. Ashta, P. Mogensen, D. Catania, and A. F. Cattoni, "A multi-qos aggregation mechanism for improved fairness in wlan," in *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*, Sep. 2013, pp. 1–5.

[14] D. Kim and S. An, "Throughput enhancement by dynamic frame aggregation in multi-rate wlans," in *2012 19th IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*, Nov 2012, pp. 1–5.

[15] S. V. Azhari, O. Gurbuz, and O. Ercetin, "Qos based aggregation in high speed ieee802.11 wireless networks," in *2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, June 2016, pp. 1–7.

[16] B. Maqhat, M. Dani Baba, R. A. Rahman, and A. Saif, "Performance analysis of fair scheduler for a-msdu aggregation in ieee802.11n wireless networks," in *2014 2nd International Conference on Electrical, Electronics and System Engineering (ICEESE)*, Dec 2014, pp. 60–65.

[17] S. Seytnazarov and Y. Kim, "Qos-aware adaptive a-mpdu aggregation scheduler for voice traffic in aggregation-enabled high throughput wlans," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2862–2875, Oct 2017.

[18] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 2, March 2003, pp. 836–843 vol.2.

[19] G. Bhanage, D. Vete, I. Seskar, and D. Raychaudhuri, "Splitap: Leveraging wireless network virtualization for flexible sharing of wlans," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Dec 2010, pp. 1–6.

[20] K. Katsalis, K. Choumas, T. Korakis, and L. Tassiulas, "Virtual 802.11 wireless networks with guaranteed throughout sharing," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, July 2015, pp. 845–850.

[21] M. Richart, J. Baliosian, J. Serrat, and J. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, Sep. 2016.

[22] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed, "Programming abstractions for software-defined wireless networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 146–162, June 2015.

[23] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Computer Communincation*, vol. v.38, no. 2, pp. 69–74, mar. 2008.