Finding the Sweet Spot for Frame Aggregation in 802.11 WLANs

Jose Saldana, *Senior Member, IEEE,* Omer Topal, José Ruiz-Mas, and Julián Fernández-Navajas

Abstract—This letter proposes an algorithm for the dynamic tuning of the maximum size of aggregated frames in 802.11 WLANs. Traffic flows with opposed requirements may coexist in these networks: traditional services as web browsing or file download that need high throughput, and services with real-time requirements that need low latency. The proposed algorithm allows the network manager to find an optimal balance (*i.e.* the "sweet spot" between throughput and latency: a "delay budget" can be assigned to real-time flows, with the objective of keeping the latency as close as possible to that budget, while penalizing the throughput of traditional services as little as possible.

Index Terms—802.11, Wireless LAN, frame aggregation, real-time services

I. INTRODUCTION

FRAME rame aggregation can provide a significant efficiency improvement in 802.11 networks [1]. It was included for the first time in 802.11n, and it has been present in all the subsequent versions of the standard. There are two different aggregation schemes: A-MSDU (Aggregated MAC Service Data Unit) sends a number of MSDUs sharing a PHY header; in A-MPDU (Aggregation MAC Protocol Data Unit) scheme, a number of MPDUs are sent back-to-back, once access to the channel has been obtained. The latter scheme is widely used, and 802.11ac made it compulsory for all the frames.

The traffic mix that can be found nowadays in these WLANs combines flows of different nature, with diverse requirements. Traditional services (web browsing, file download) that just need throughput, share the network with other flows for which latency is critical, and typically consist of small packets [2] with different delay constraints [3].

As an example, in [4], a novel medium access control (MAC) scheme for hospital e-health systems was proposed, to reduce the latency of delay-sensitive healthcare traffic. The delay constraint was extremely low: 200 μs. Other studies consider that ultra-low latency means that the delay is looser, in the order of some milliseconds (*i.e.* $\leq$ 5 ms) [5]. Another study looking for ultra-low latency in WLANs [6] puts the target in 1 ms, considering emerging applications as vehicular or ultra-reliable communications. Finally, in [7], a QoS-aware

J. Saldana, J. Ruiz-Mas and J. Fernández-Navajas are with CeNIT Group - Aragon Institute of Engineering Research (I3A), EINA, University of Zaragoza, Spain. Email: {jsaldana, jruiz, navajas}@unizar.es. O. Topal is with Research and Development Department, AirTies, Turkey. Email: omer.topal@airties.com

adaptive scheduler was proposed, which was able to keep VoIP delay under a limit, based on cross-layer optimization, using certain RTP (Real-Time Protocol) features.

When a STA sends or receives a big aggregate frame, the rest of the STAs sharing the same AP have to wait, and this results in an increased delay and jitter for them. Therefore, frame aggregation has a double effect: its efficiency increase is highly beneficial for bandwidth-hungry flows, but it comes at the cost of undesired delays for real-time ones.

In order to mitigate these bad effects for real-time flows, the authors of [8] studied a scenario in which a central controller could enable or disable the aggregation features of a set of Wi-Fi APs belonging to the same local deployment. The mechanism gathered information from the network, and disabled aggregation in an AP whenever a flow with real-time requirements became associated to it. As soon as it left the AP (due to user movement), aggregation could be enabled again, as long as there were no more STAs generating that kind of traffic. As an alternative, the study also considered the possibility of permanently configuring a reduced value of the maximum size of the A-MPDU, which also provided some advantages. In [9] the use of Machine Learning (ML) techniques was proposed for adaptively adjusting the maximum A-MSDU length on a per-user basis. The paper was mainly focused on reducing packet loss rate and increasing channel occupation.

In this research context, the main contribution of the present paper is the proposal of an algorithm that dynamically tunes the value of the maximum A-MPDU in order to meet a "delay threshold," defined as the maximum latency tolerated by the real-time flows present in the WLAN. It can be considered as the take of the total "delay budget" that can be "spent" in the Wi-Fi network. As we will next explain, different methods can be employed by the algorithm.

The difference with respect to [8] is that we do not follow an on/off approach for the aggregation, but a real-time adaptation that will allow us to find the "sweet spot" of the trade-off between throughput and latency explored in that paper.

There are two differences with respect to [9] first, we are not considering A-MSDU (length up to 3,839 bytes), but A-MPDU (length up to 65,535 bytes), which is more widely used and may result in a much higher delay for real-time flows; second, our objective is not to achieve the maximum throughput, but to achieve the optimal balance between the users of real-time applications and "traditional" ones that share the same network.

The objective of the proposed algorithm (we will refer to it as the *tuning* algorithm from now on) is to maximize the

throughput of the TCP flows, while granting that the latency of real-time STAs is below the desired threshold.

It behaves as follows: at the beginning, the APs start with the maximum A-MPDU value, *i.e.*, they aggregate as many frames as possible. The flows with real-time requirements are periodically monitored, and their maximum delay is obtained at each AP. If this delay is above a threshold, the A-MPDU size is reduced. If the delay is below the threshold, the A-MPDU limit is increased until it reaches the maximum.

Different methods for increasing and decreasing the limit can be employed:

- a linear increase / decrease with a fixed-size step;
- a geometrical increase / decrease, using a coefficient for obtaining the new value;
- a *drastic* increase / decrease, *i.e.* as soon as the value exceeds the threshold, the aggregation is enabled / disabled.

Obviously, a different method can be employed for increasing and decreasing the value. This results into a total of 9 possible combinations, among which the following ones have been chosen as significant (the concrete values of the increasing / decreasing steps have been found empirically as the ones that provide the best results, as we will see in Section III):

- Method #1. Linear increase and linear decrease: if the latency is above the objective, the maximum A-MPDU value is decreased by a fixed value (3,000 bytes). If it is below, the value is increased by the same value.
- Method #2. Geometric decrease and increase: if the latency is above the limit, the A-MPDU is reduced by a factor of 0.618; if it is below, it is multiplied by its inverse (1.618).
- Method #3. Instantaneous drastic reduction to the minimum, linear increase: if the latency is above the latency budget, the A-MPDU is reduced to the minimum. If it is below, the A-MPDU is increased by 6,000 bytes.
- Method #4. Linear decrease, and instantaneous *drastic* rise to the maximum: if the latency is below the latency budget, the A-MPDU is increased to the maximum. If it is above, it is decreased by 6,000 bytes.

As a baseline, we will use the method in which aggregation is just activated or de-activated [8] (we will call it the *disable* algorithm, and it is in fact a particular case including a *drastic* increase and a *drastic* decrease).

In the next section we present the test setup, and in Section III the results are presented and compared. The paper ends with the Conclusions.

II. TEST SETUP

The tests have been performed using ns3 simulations of an 802.11 scenario with a number of APs and a central controller. As a starting point we used the code released by the authors of [8], in which we have included new features as *e.g.* 802.11ac. On behalf of accuracy, we have also used the next models instead of the original ones:

- *Idealwifi*, an SNR-based rate manager, in which a transmitter learns the SNR information at the receiver by an

TABLE I
SIMULATION PARAMETERS

ns3 version	ns-3.30.1	**UDP pkt size**	60 bytes
Mobility model	Random Waypoint	**UDP rate**	50 pps
Walking speed	1.5 m/s with pause time 2 s	**TCP pkt size**	1,500 bytes
WiFi model	*SpectrumWifiPhy* with *MultiModel SpectrumChannel*	**TCP variant**	New Reno
Channels	36 to 128 (20 MHz channels)	**Simulation time**	60 s
WiFi phy std	802.11ac	**RTS/CTS**	Enabled
WiFi rate control model	*Idealwifi* Manager	**Inter-AP distance**	50 m (25 m to the border)
Propagation loss model	*FriisSpectrum*	**Short guard**	Not enabled
Error rate model	*NistErrorRateModel*	**EDCA priorities**	Enabled
Delay monitoring period	*250 ms*	**A-MPDU range**	1,600 to 65,535 bytes

out-band channel. It directly manages the rates and also considers the bit error rates (BER) during transmissions.
- *FriisSpectrum* propagation loss model is used to take the frequency into account in the tests [1]. It is important, since it directly impacts the RSSI and hence system performance.
- *Nist* error rate model, which has a performance closer to real behavior than other ones (*e.g. YansWifi*) [10].

Table I summarizes the parameters employed. The code used in the presented tests has been released (see https://github.com/wifi-sweet-spot/ns3).

In all the simulations, two kinds of STAs share the scenario: *a)* "TCP STAs" are performing a file download, and *b)* "UDP STAs" are running an application that sends small packets with real-time requirements (see Table I). In all cases, $2N$ STAs will be present: N TCP STAs and N UDP STAs. These applications have opposite target Key Performance Indicators (KPIs): TCP STAs want high throughput whereas UDP STAs' main objective is to keep the latency low.

An interval of 250 ms has been used as the time between adjustments of the maximum A-MPDU size. This value was found empirically, as it presented a good trade-off between a fast adjustment (it correctly keeps track of the evolution of the latency) and a low computation overhead.

III. TESTS AND RESULTS

We will first present a scenario with a single AP and two STAs (one TCP and one UDP), which will allow us to qualitatively examine the behavior of the proposed algorithm, and the ones used for comparison purposes. Then, we will use a bigger scenario with 4x4 APs and a variable number of STAs, in order to quantitatively measure the overall parameters (latency and throughput) with more accuracy.

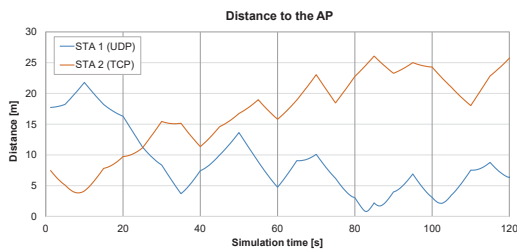[1]See https://www.nsnam.org/doxygen/classns3_1_1_friis_spectrum_propagation_loss_model.html

Fig. 1. Distance of the STAs to the AP.

A. Scenario with a single AP and two STAs

In this subsection we use a 50x50 meter scenario that includes a single AP shared by two Wi-Fi STAs (*N*=1). We present three experiments: *a) no aggregation; b) aggregation always active; c) tuning algorithm, method # 1*. They will let us observe the behavior of the A-MPDU adaptation in detail.

Note that the *disable* algorithm [8] would produce the same results as the *no aggregation* case: since a TCP STA is always connected to the AP, aggregation in the AP would in fact be always disabled.

The same random seed has been used in the three experiments, in order to have the same mobility patterns. In Fig. 1 (valid for *a*, *b* and *c*), the distance from the STAs to the AP is shown. It can be seen that the TCP STA gets very close to the AP in $t = 8$ s. From that moment, it moves away from it, while the UDP STA gets closer.

Fig. 2 shows the results of the *no aggregation* experiment. The highest TCP throughput is obtained around $t = 8$ s, and the delay is always below 20 ms (the average delay is 4.7 ms, and the average throughput is 16.6 Mbps).

In the *aggregation always active* case, (Fig. 2(b)), the throughput obtained by the TCP STA is much higher (34.5 Mbps average), but this comes at the cost of a higher latency for the UDP STA (10 ms average) . Some latency spikes of the UDP flow appear when the distance of the TCP STA grows, since its rate gets reduced and A-MPDUs can be up to 65 KB.

Finally, if the *tuning* algorithm (*method* #1) is employed (Fig. 2(c)), we can still maintain a high throughput for the TCP STA (31.3 Mbps , *i.e.* only 10 percent reduction with respect to the *aggregation always active*), whereas the UDP latency presents a behavior somewhat similar to the one obtained without aggregation. Its average delay is now 6.2 ms (25 percent increase with respect to *no aggregation*). Fig. 3 shows the evolution of the A-MPDU maximum size as the algorithm adapts it as a function of the UDP latency.

The trade-off appears clearly: the system sacrifices some throughput of the TCP flow in order to grant a low delay for the real-time one. The *tuning* algorithm allows us to find and set this "sweet spot," giving us the possibility to tune the trade-off between maximum throughput for TCP and minimum latency for UDP.

In the next subsection we will measure this in detail, with a higher number of realizations, more APs and different numbers of STAs sharing the scenario.

B. Scenario with a number of APs

In order to capture the effect of the proposed *tuning* algorithm, this section presents averaged results obtained in a scenario with a higher number of APs and STAs: it consists of a set of 16 APs, deployed in a 4x4 array with 50 meters between them, and 25 m to the border (total 200 x 200 m). A different number of UDP and TCP STAs (from 4 to 100, *i.e.* $N = 2$ to 50) move through the scenario while running a TCP or a UDP application.

The four methods are compared between them, and also versus *no aggregation*, *aggregation always active*, and the *disable* algorithm. Fig. 4 shows the total throughput obtained by the TCP STAs, and Fig. 5 presents the average latency of the UDP STAs. The results are the average of 15 realizations of 60 seconds each (50 realizations were required for the *aggregation always active* scenario). 95% confidence intervals are provided.

It can be observed that the *no aggregation* policy adds very low latency, but it is always the one with the lowest throughput: this could be expected, since the transmission of every single frame requires the use of the CSMA/CA mechanism.

In contrast, the *aggregation always active* experiment is the one that achieves the highest throughput (which always doubles the one achieved by means of *no aggregation*). However, this comes at the cost of a significantly higher latency (in some cases even doubled).

The test of the *disable* algorithm reports the same latency as the one in the *no aggregation* one: this could be expected: as soon as a UDP STA gets associated, the AP disables the aggregation, so this STA always *sees* a non-aggregating AP. however, the throughput of the TCP STAs gets reduced significantly (between 52 and 81 percent).

Finally, if the proposed *tuning* algorithm is used, it can be seen that the latency *vs* throughput trade-off can be adjusted in order to find the most convenient point: the throughput (Fig. 4) is now much closer to the *aggregation always active* case, while the latency (Fig. 5) is very similar to the one achieved by *no aggregation* or the *disable* algorithm. Fig. 6 shows the jitter of the UDP traffic, in which the same tendency can be observed: it is very similar in all cases, except for the *aggregation always active* one, in which the jitter is increased by roughly 70 %.

As far as loss rate is concerned (Fig. 7), no special impact is observed if the different methods are employed: the maximum difference between them is 7% . In addition, the loss rate is always below the values obtained by the *no-aggregation* (14% below on average) and *aggregation always active* (18% below on average) cases.

Regarding the differences between the four methods, #4 is the one that grants a highest throughput for the TCP STAs. The cause is the fact of having a more aggressive rise when the latency is below the limit. However, this comes with a penalty in terms of latency for the UDP flows. To a lesser extent, something similar happens with *method* #2 (geometric increase and decrease), and #3, which also has a fast rise in steps of 6,000 bytes and a drastic reduction.

At the same time, *method* #1 is the one with the best results in terms of latency for the UDP flows. The results obtained with *method* #3 are very similar to those obtained with #1, with a slight throughput reduction caused by the aggressive policy
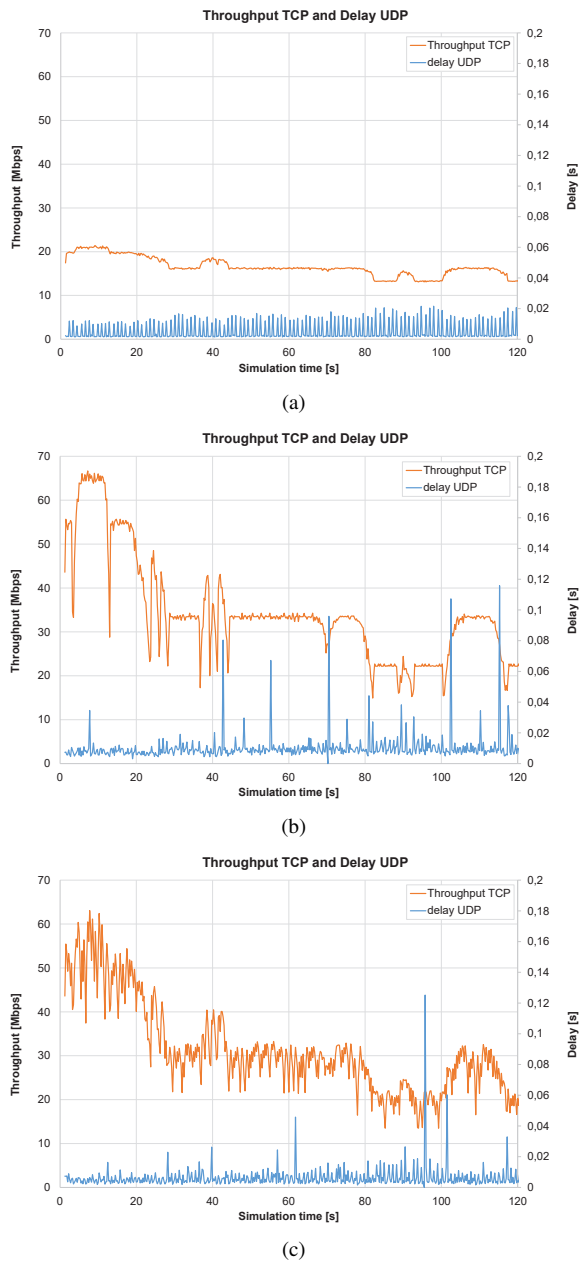
(a)

(b)

(c)

Fig. 2. Single AP scenario: TCP throughput and UDP real-time delay: *a)* no aggregation; *b)* aggregation enabled with a constant value of A-MPDU size of 65 KB; *c)* the *tuning* algorithm (*method* #1) limits the maximum A-MPDU value.

that reduces the A-MPDU size to the minimum whenever the latency limit is reached.

All in all, it seems clear that the use of a soft increase and decrease draws better results in terms of latency, but it slightly reduces the throughput. This is quite noticeable for *method* #1 (linear) and also for #3 (drastic reduction). In contrast, the aggressive rise of *method* #4 is better for bandwidth-hungry flows and worse for real-time ones.

Fig. 8 presents some of the tests that have been run in order to find the best value of the increasing / decreasing steps. In this case, *method* #1 with five different values for the step (ranging from 1,000 to 15,000) is compared. It can be seen that high values of the step are not convenient: with 10,000 and 15,000 we obtain a higher latency, but the throughput is
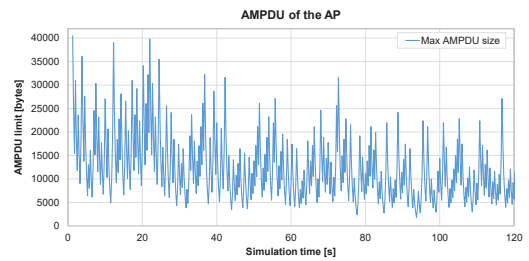

Fig. 3. Single AP scenario with *method* #1: evolution of the maximum A-MPDU value.
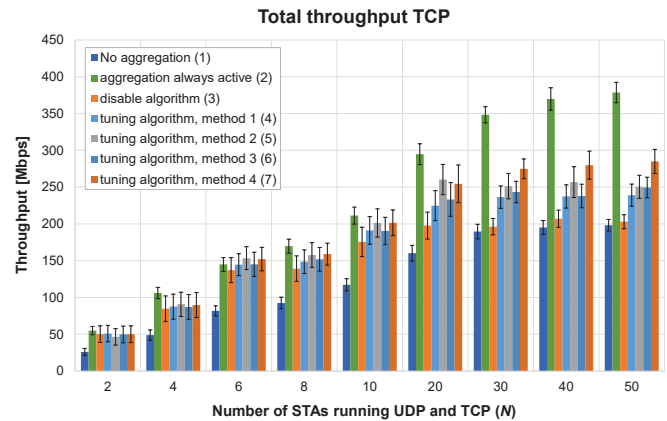

Fig. 4. Scenario with 16 APs: total throughput of the TCP STAs.
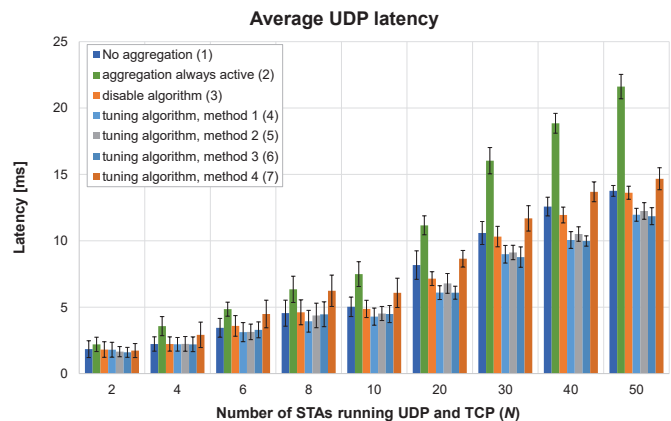

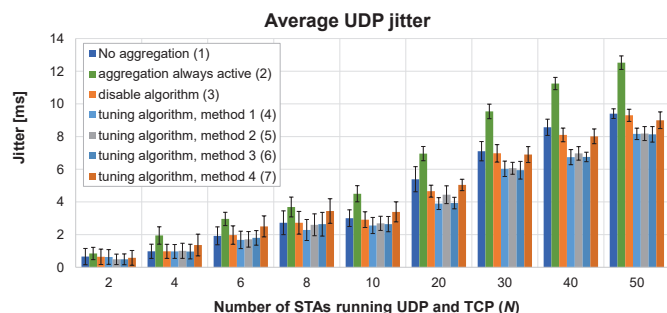Fig. 5. Scenario with 16 APs: average latency of the UDP STAs.

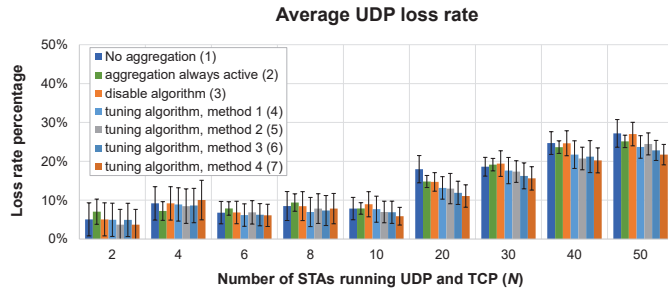Fig. 6. Scenario with 16 APs: average jitter of the UDP STAs.

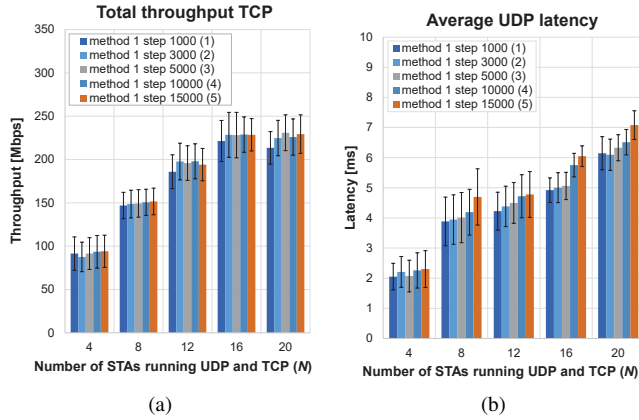Fig. 7. Scenario with 16 APs: loss rate of the UDP STAs.

Fig. 8. Scenario with 16 APs: comparison for *method* #1 with different values of the step: *a)* throughput; *b)* latency.
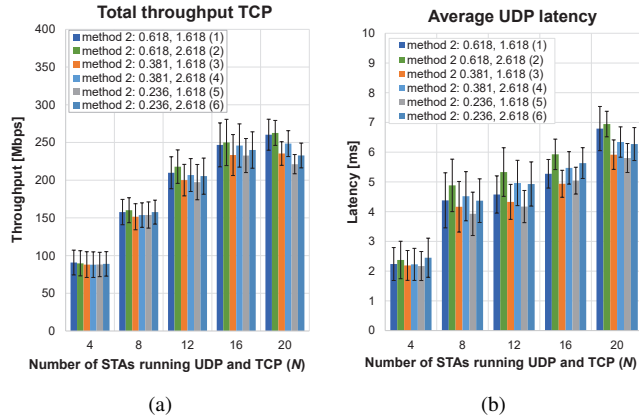

Fig. 9. Scenario with 16 APs: comparison for *method* #2 with different values of the coefficients: *a)* throughput; *b)* latency .

not increased. In contrast, a value of 1,000 results in a lower throughput with the same latency than 3,000 or 5,000, which are the best values for the increasing / decreasing step.

Different coefficients have been employed for *method* #2, as shown in Fig. 9: 0.618, 0.618^2 (0.381) and 0.618^3 (0.236) for decreasing the size, and 0.618^{-2} (2.618) and 0.618^{-1} (1.618) for increasing it. It can be observed that an increase factor of 2.618 results in a slightly higher latency in most cases. The decreasing factor of 0.236 reduces the throughput. The pair 0.618 and 1.618 (used as default in previous tests) presents a good balance.

IV. CONCLUSIONS

This paper has proposed the dynamic tuning of the maximum A-MPDU value in an 802.11ac WLAN. It allows to find

a good balance when applications with different objectives in terms of network KPIs share the network. If aggregation is used, the throughput rises, but at the cost of a higher latency. If it is not used, or disabled, the latency gets reduced, but this comes with a penalty for the services with high throughput requisites.

The proposed algorithm allows the network manager to find an optimal balance (the "sweet spot") between these KPIs. In our case, a "delay budget" has been assigned to the real-time flows, so the algorithm has the objective of keeping the latency as close as possible to that budget, while penalizing the throughput as little as possible.

Different methods for tuning the maximum A-MPDU value have been compared. The results indicate that the use of a soft increase and decrease can reduce the latency, but it slightly penalizes the throughput. In contrast, methods with a more aggressive policy result in a higher throughput, at the cost of an increased delay.

As future work, the effect of the use of different values for the monitoring period will be studied. In addition, the dynamic tuning of the value of the "delay budget" depending on the end-to-end latency could be considered. If RTP traffic is used, periodic RTCP Receiver Report (RR) packets could be used for obtaining the end-to-end latency, as done in [7]. Other policies for the tuning of the A-MPDU could also be explored, as *e.g.* considering as the main objective to maximize user's satisfaction, defined as a weighted combination of different KPIs, depending on the kind of services present in the network. As this would increase the complexity, the use of ML could be considered in that case.

REFERENCES

[1] B. Ginzburg and A. Kesselman, "Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11n," in Proc. 2007 IEEE Sarnoff Symposium (pp. 1-5). IEEE.
[2] R. Xie, H. Yin, X. Chen and Z. Wang, "Many access for small packets based on precoding and sparsity-aware recovery," *IEEE Transactions on Communications*, vol. 64, no. 11, pp. 4680-4694, Nov. 2016.
[3] M. Suznjevic and J. Saldana, "Delay limits for real-time services," Dec. 2015. [Online]. Available: https://tools.ietf.org/html/draft-suznjevic-tsvwg-delay-limits-00.
[4] Y. Lv, M. P. I. Dias, L. Ruan, E. Wong, Y. Feng, N. Jiang and K. Qiu, "Request-Based Polling Access: Investigation of Novel Wireless LAN MAC Scheme for Low-Latency E-Health Applications," *IEEE Communications Letters*, vol. 23, no. 5, pp. 896-899, Mar. 2019.
[5] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi and C. Assi, "Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668-682, Mar. 2019.
[6] C. A. Grazia, "Towards 1 ms WLAN Latency," In Proc International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob) pp. 278-284. IEEE, 2018.
[7] S. Seytnazarov and Y. T. 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.
[8] J. Saldana, J. Ruiz-Mas and J. Almodóvar, "Frame Aggregation in Central Controlled 802.11 WLANs: The Latency Versus Throughput Tradeoff," *IEEE Communications Letters*, vol. 21, no. 11, pp. 2500-2503, Nov. 2017.
[9] E. Coronado, A. Thomas and R. Riggio, "Adaptive ML-Based Frame Length Optimisation in Enterprise SD-WLANs," *Journal of Network and Systems Management*, vol. 28, pp 850–881, Mar. 2020.
[10] G. Pei and T. R. Henderson, "Validation of OFDM Error Rate Model in ns-3," Boeing Research Technology. 2010: 1-5.