Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Delay and backlog control of aggregation systems for wireless communications

Jean-Philippe Georges, Thierry Divoux, Damien Breck

Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France
(e-mail: firstname.name@univ-lorraine.fr).

Abstract: The aggregation system implemented in wireless communication networks aims to optimize the network efficiency regarding the encapsulation information and the medium access mechanisms. This paper proposes to evaluate the performances of such (A-MSDU like) system from a dedicated frame point of view, which is important in order to predict the Quality of Service offered to a client or to an application. Regarding the incoming flows, enabling or disabling the aggregation system, and tuning in real time the aggregate size, makes possible to satisfy both the users' requirement and the network provider efficiency. In the case of voice or video class wireless communication where some loss of data is acceptable, numerous simulations using Riverbed Modeler led to express several recommendations embedded in an algorithm which sets dynamically the aggregation parameters in order to adjust it to the incoming traffic.

Keywords: Telematics, Control of Network, Wireless Communications, Frame Aggregation.

1. INTRODUCTION

Better performances and reduced costs are expected for telecommunications networks and have led to the development of various aggregation systems. These systems gather several incoming flows in order to carry them simultaneously through the network. They obviously impact the quality of service offered to each of the aggregated flows. In this sense, it appears necessary to understand their nature, their behaviours and assess their performances. The frame/packet aggregation boosts the efficiency of the network. The encapsulation weight is optimized and especially for wireless networks, the aggregation limits the competition for accessing the medium, and so enhances the bandwidth offered to the users. These improvements are in addition to the physical advances (Wi-Fi standards IEEE Computer Society (2009, 2013) and IEEE 802.11ax).

This type of system is particularly useful at the border between two networks having different capacities. The aggregation is realized in the gateways, such as in 802.11 access points. The performance gap between Ethernet and Wi-Fi makes the aggregation interesting for the bandwidth optimization. Indeed, it enables a station to transmit more users' data when it gains the medium. So it is expected a better service for the concerned application. But what is the impact for the concurrent applications? Do all the applications benefit from this system when they are aggregated together? Do they compete for the available space in an aggregate? What about the stations which compete for the medium access with a station which aggregates? Today, these parameters are set beforehand in each device, and are never modified online. Nevertheless, it is difficult for the provider to set them a priori, without any knowledge on the incoming flow. Based on recommendations resulting from extensive simulations (Section 3), we propose in Section 4 an algorithm enabling to choose dynamically who aggregates, and to set the aggregation parameters, from Quality of Service requirements given by both the client and the provider. This algorithm is evaluated and discussed in Section 5.

2. THE PACKETS AGGREGATION IN THE 802.11 STANDARDS

Despite the CSMA-CA mechanism, collisions may occur and the distance between the station and the access point influences the bitrate. From the station point of view, when gaining the medium, it is hence important to transmit as much data as possible. Aggregating will reduce the number of competitions to gain the medium. This is illustrated in the Fig. 1 where only 6 packets are classically transmitted when eleven by activating the aggregation.

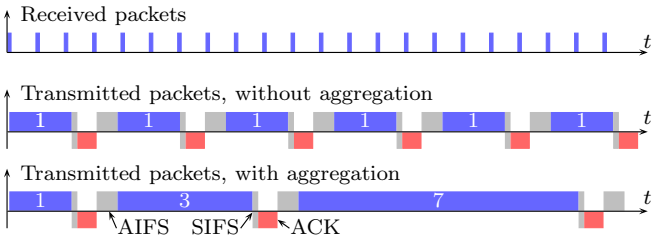

Fig. 1. The yield, with and without aggregation

Improving the yield at the MAC level relies on two aggregation systems Karmakar et al. (2017). The first one is located at the MAC layer input and aggregates the MSDU (*Mac Service Data Units*) in order to build an A-MSDU (*Aggregated MSDU*). The second one is located at the MAC layer output and aggregates the MPDU (*Mac Protocol Data Units*) in order to build an A-MPDU (*Aggregated MPDU*) Saldana et al. (2017). In this paper, we decided to only study the MSDU aggregation, since it proposes a greater reduction of the headers' ratio (our

works will then apply both for 802.11n and 802.11ac). However, we claim that the works presented in this paper can be easily transposed to MPDU aggregation. A packets' aggregation system will transform several applications flows into a unique aggregated "super-flow". As defined in IEEE Computer Society (2009), the system accumulates the packets (MSDU), and releases them if:

- A size threshold $\mathcal{S}$ is reached.
- The associated class gains the medium.

The size threshold cannot be exceeded. It is *a priori* configured at one of the two values defined in the standard.

3. 802.11 AGGREGATION SYSTEM ANALYSIS

The purpose here is to formulate observations (regarding delays and backlogs evolution) that will be used next in order to control aggregation parameters. We used extensive simulations (through *Riverbed Modeler*) of the system that aggregates MSDU received by a 802.11n MAC layer.

3.1 Simulations

In order to obtain significant delays, it is necessary to stress our system. This is achieved considering that bursts increase the delay since they increase the backlog which is part of the delay. That is why we have used a traffic profile which is based on bursts, while allowing the system enough time to proceed at least a part of this burst. It is the same as Skordoulis et al. (2008) who used also *Riverbed Modeler*, but with the Intel model. The burst is repeated and arrival and duration times are fixed as constant (ON/OFF traffic). Inter-arrival has been obtained by trial and error in order to stress the system (just before having packets drops due to memory overflow). Keeping this average bitrate, the size and the inter-arrival time vary according to an exponential distribution as described in Table 1. The competing flow is then arbitrarily chosen.

Table 1. Incoming flows parameters

Distribution law	exponential
Mean packets' size (bytes)	{250, 500, 1 500}
Mean inter-arrival time (μs)	{25, 50, 150}

The argument of the exponential distribution is the mean of the interval between successive events (frames' size and/or departure time) during the ON state. In the following, the notation $\xi\,(125\,|\,25)$ corresponds to a flow that is going to send 125 bytes packets every 25 μs on average during the ON period.

Reflecting the numerous parameters offered by the MAC and physical layers, we have restricted the study by fixing some physical ones (frequency, bandwidth...), and some MAC others, like TXOP (*Transmission Opportunity*) or the memory size. The considered topology is based on an Access Point relaying the packets transmitted by one or two stations to a wired network. Other scenarios might boil down to this topology: for example, a station competing with the Access Point, an AP with several flows belonging to different service classes... A station only differs from an AP by a potential lower AIFS for *VOice* and *Video* classes. This topology is also representative of the Wireless Sensor Networks in which all sensors are transmitting to the sink. Even if they highly use other protocols, these networks could benefit from an aggregation mechanism. The wireless device configuration is detailed in the Table 2. It is representative of the minimum needed to be 802.11 certified. From the (IEEE Computer Society, 2009, Table 20-30), these parameters enable to deduce a 65 $Mbps$ bitrate.

Table 2. Configuration of the wireless stations in the *Riverbed Modeler* simulator

Medium Access Policy	HCF EDCA
A-MPDU	deactivated
Block acknowledgment	deactivated
TXOP limit	1 MSDU
Memory size	1 024 000 bits
SigExt	6 μs
ISM band	2.4 GHz
Base bitrate	65 $Mbps$
Maximum bitrate	600 $Mbps$
Space flows	1
Guard interval	800 ns
Bandwidth	20 MHz
Time slot	9 μs
SIFS	10 μs
PPDU format	HT-mixed
Sensitivity	95 dBm
Propagation	free space
Memory	shared
Seeds	30

The remaining configuration parameters are those we consider as control variable whose influence will be studied now. The tested values are gathered in Table 3.

Table 3. MAC parameters

A-MSDU	{activated, deactivated}	
Size threshold $\mathcal{S}$	$\{\mathcal{S}_l : 3\,839, \mathcal{S}_H : 7\,935\}$ bytes	
AIFSN (Service class)	BE(3)	VO(2)
$CW_{\min}$	$aCW_{\min}$	$(aCW_{\min}+1)/4-1$
$CW_{\max}$	$aCW_{\max}$	$(aCW_{\min}+1)/2-1$

The system aggregates MSDU received by the 802.11 MAC layer. These are stored in a queue which is associated to a service class. This system closes an aggregate if:

- The amount of data stored in this queue exceeds a size threshold $\mathcal{S}$.
- The service class associated to the queue gains the medium.

3.2 Observations

After more than 15 000 simulations, Breck et al. (2019), we know that aggregating always improves the application bitrate for the aggregated flow but also that:

- Aggregating reduces the average delay of the packets.
- Aggregating also reduces the maximum delay and avoids packets drops.
- Without drops, aggregating reduces the maximum backlog, and so, the memory occupancy.
- Choosing what service class to aggregate significantly modifies the packets' maximum and average delays.
- Setting the threshold such packets are aggregated significantly, modifies the packets' maximum delay.
- A flow which competes with a priority flow which aggregates has less chance to gain the medium.

- A flow which competes with a lower priority aggregating flow has some packets whose delay increases.
- Sometimes, the less priority station gains the medium, even if the priority queue is not empty. This is due to the conservation mechanism of the backoff.
- The behaviours observed on the delay and the backlog remain identical for different distributions (constant or exponential for 30 seeds).
- Large packets (1 500 bytes) limit the benefits.
- Too spaced packets inhibit the aggregation.

Nevertheless, the competitor flow is often penalized: if a low priority flow aggregate, the other flow has to wait for the whole aggregate to be transmitted, and its delay and backlog increase. On the other side, high priority flows are always aggregated at the expense of their competitors, due to the long transmission time of the aggregates which increases the waiting duration to try to gain the medium again, and so decreases the remaining backoff decrementing speed. This reduction is added to the 802.11e mechanisms which already limited the medium access probabilities for the low priority classes. Some applications do not use the Classification of Service, and that could lead to *VOice* packets to be proceeded as *Best Effort* packets. In this case, aggregating affects the performances of these critical applications.

Excepted for the two specific cases described above, aggregating improves the performances. If the high priority flow is aggregated, its competitor will also benefit because the medium will be free earlier. If the low priority flow is aggregated, it has the possibility to transmit a greater amount of data when it gains the medium. In conclusion, we can claim that it is globally interesting to activate the aggregation on all queues. Indeed, the high priority flow processes its backlog more quickly, and that offers more space for the low priority flow. From all these observations, we propose now several configuration recommendations:

Remark 1. When a lower-class flow reaches the famine, upper-class flows have to be aggregated in order to reduce the average delay and the maximum backlog.

Remark 2. If the delay and the backlog remain unsatisfactory, the considered flow has to be aggregated in order to reduce the average delay and the maximum backlog.

Remark 3. When the backlog of an upper class increases up to the memory limit, the aggregation has to be disabled on the lower classes flows in order to reduce the backlog and the average delay of the upper-class flow.

Remark 4. For small size packets (less than 500 bytes), there is no need to modify the aggregate size threshold to reduce the maximum backlog and the delays.

Remark 5. For large size packets (1 500 bytes and more), flows have to be aggregated at the maximum threshold in order to reduce the maximum backlog and the delays.

4. DYNAMIC CONFIGURATION OF THE SYSTEM BASED ON THE QOS REQUIREMENTS

Fig. 2 shows the "black box" representation of the algorithm which will control the aggregation parameters (ON/OFF, per class, threshold), regarding the dynamic behaviour of the system (backlog and delays), and the clients and access providers' requirements.

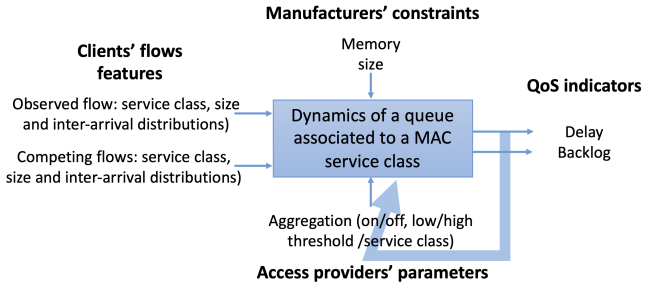

Fig. 2. The feedback on the aggregation system

4.1 Scientific positioning

Authors created algorithms either for the packets scheduling for optimizing the aggregation performance, or for the aggregation mechanism itself. For example, Bhanage et al. (2011) propose an aggregation algorithm based on the backlog. The key principle is: *aggregating while there are frames which may be transmitted in a same TXOP duration*. The algorithm behaviour can be summarized as follows: when a station gains the medium, it stores the packets while the size threshold is not reached, and while the TXOP duration has not expired. As soon as one of these two conditions is false, the available aggregate is transmitted. Respecting the TXOP duration ensures fairness between competing stations. The algorithm is evaluated thanks to real audio and video traffic traces on the *ORBIT* experimental platform, with equipment which embeds a modified *MadWifi* driver. Experiments show significant improvements on the average delay, application bitrate and jitter. The relation between TXOP and aggregation mechanisms has been also studied in Kosek-Szott and Rapacz (2020).

On another hand, Selvam and Srikanth (2010) propose a different approach based on the packets arrival deadline, a size threshold, and an optimal aggregate size. This latter is calculated from the Bit Error Rate. The algorithm firstly tests if the packet deadline is reached, then if the size threshold is. If one of the conditions is true, the backlog is compared to the optimal size. If false, an A-MSDU is normally building. If true, packets are sorted by increasing size and A-MSDU are built and encapsulated in an A-MPDU. Evaluated thanks to a simulator they have developed in C language, authors claim that this algorithm improves the performances if the medium is not saturated. A different method is used by Saif et al. (2012) who suggest to reduce the MSDU headers from 10 bytes to 4. Benefits are significant especially for small packets: a NS2 simulation shows it is 30% better than a classical aggregation for 128 bytes' packets. Kowsar and Biswas (2017) study a combination of A-MSDU and A-MPDU techniques different compared to the standard in order to improve parameters like the throughput.

Dely et al. (2010) use a fuzzy control in order to configure the aggregation system. Authors control the delay added by the aggregation regarding information on the MAC delay and the bitrate. Several indicators are used to offer a high application bitrate for a small delay. Azhari et al. (2016) develop a novel PID controller for (A-MPDU) aggregation focusing on the global link queue size.

Karmakar (2019) use a fuzzy control in order to minimize the energy consumption issues.

All these works focus on the optimization of the aggregation system regarding global performances such as the aggregates delay, the channel percentage of use, the total throughput Saldana et al. (2017) or even the efficiency regarding the contention window Machrouh and Najid (2018). Some also propose modifications of the standard (header modification for example). Our objective is to evaluate the performance of the aggregation system, without any modification, and for a specific flow (in opposition to global analysis). Furthermore, no temporal threshold is considered since it may involve a retention of the aggregates, and so a delay increase. We want to preserve the opportunistic philosophy of the aggregation system.

4.2 Aggregation location

Analysing the code related to the finite state machine in the *Riverbed Modeler* simulator (Fig. 3), we have identified that the aggregation is implemented at two steps. Firstly, at each packet reception, the specific headers are added to the aggregates. Secondly, the aggregation itself is achieved just before the frame transmission (input of the TRANSMIT state). It means that a potential aggregation parameters adaptation will have to be run as soon as the station wins the medium and will be called relatively to the service class that will have won the competition.

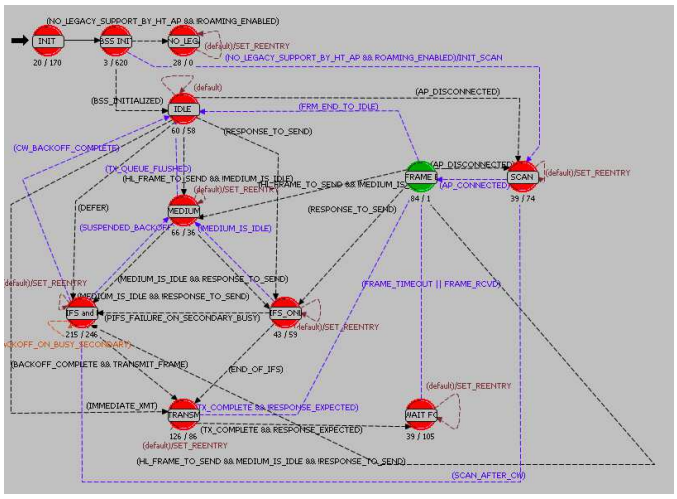

Fig. 3. Hybrid Coordination Function in *Riverbed Modeler*

In order to follow the evolution of each class state, we have developed two new probes: the first one dynamically measures the maximum aggregates' size for each class, enabling to follow the actions realized by the algorithm. The second probe enables to measure the backlog in bytes when *Riverbed Modeler* previously gave it in packets.

Other modifications have been done in the 802.11 program: firstly, the aggregation may be now dynamically activated or not during a simulation, when it was previously static and set before the simulation. Then, the threshold may dynamically vary and different values can be associated to each class. In the original model, it was a common value for all classes and set before running the simulation.

4.3 Proposal of an algorithm to configure the aggregation

To design this algorithm, the five recommendations previously stated will be exploited. The first one tells us what to do when the backlog or the delay becomes critical. That is why our algorithm will be based on an observer of these two indicators. Fig. 4.3 describes such an observer.

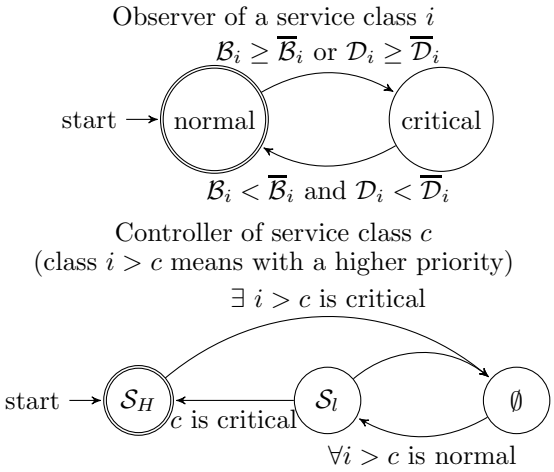

Fig. 4. Principles of the aggregation configuration

Let $\mathcal{B}_c$ and $\mathcal{D}_c$ be respectively the measured backlog and the measured delay for a service class c (*Access Category* AC). Remember that we consider here an aggregation system implemented on a station or an access point which deals with four service classes. Regarding these indicators, the quality offered to a service class might be sufficient (normal case) or not (critical case) compared to maximum set points $\overline{\mathcal{B}_c}$ and $\overline{\mathcal{D}_c}$. This last state critical will be used next to activate and to set the aggregation parameters in accordance with the recommendations claimed in the previous section. Each class may be, not aggregated (state $\emptyset$), aggregated at the low threshold $\mathcal{S}_l$ or aggregated at the high threshold $\mathcal{S}_H$ (for each station, 81 combinations may occur).

Fig. 4 explains the principle of the algorithm. It directly comes from the recommendations, especially claim 3. Initially, a class c aggregates at the high threshold, but if a class with a higher priority starts to be saturated (critical), the aggregation is deactivated for the class c. Since it is expected that such control will be distributed for all classes, it means that the aggregation will also be deactivated for all class with a lower priority. Since the initial state for a service class corresponds to the aggregation, claim 1 is already taken into consideration. The link between aggregation states $\mathcal{S}_l$ and $\mathcal{S}_H$ directly comes from claim 2. When the backlog and the delay return to the normal values of high-priority classes, low-priority classes can aggregate again. In order to avoid the system to be unstable, we assign the low threshold to the low-priority classes while they respect their own maximum backlog. Claims 1 and 2 may hence only be considered when claim 3 statement is not activated. Then, claims 4 and 5 lead to use the high threshold for large packets, which is not necessary for small ones. But since it is not negative, the initial state consists on aggregating all classes at the high threshold. The full description of the algorithm is hence given in Algorithm 1.

Data: the anticipation factor percentage $p \in \mathbb{R}$, the list of service class $\mathcal{C} := \{BK : 1, BE : 2, VI : 3, VO : 4\}$ and for each class, the current backlog level $\mathcal{B}$ and the backlog limit permitted $\overline{\mathcal{B}}$, and finally, the last delay $\mathcal{D}$ and the maximum delay permitted $\overline{\mathcal{D}}$
Result: the state of the aggregation function for each class $\mathcal{A}$ ($\emptyset$ means no aggregation for the given class)

Class c **ready to transmit**
  /* *detect if a higher class (queue) is saturated* */
  overflow $\longleftarrow$ $false$ **foreach** $i \in \mathcal{C} \mid i > c$ **do**
    if $\left(\mathcal{B}_i \geq p \times \overline{\mathcal{B}}_i\right)$ *or* $\left(\mathcal{D}_i \geq p \times \overline{\mathcal{D}}_i\right)$ **then**
      overflow $\longleftarrow$ $true$ break
  /* *update the aggregation set points* */
  if overflow **then** $\mathcal{A}_c \longleftarrow \emptyset$;
  else if $\mathcal{A}_c = \emptyset$ **then** $\mathcal{A}_c \longleftarrow \mathcal{S}_l$;
  else if $\mathcal{A}_c = \mathcal{S}_l$ **then**
    if $\left(\left(\mathcal{B}_c \geq \overline{\mathcal{B}}_c\right)$ *or* $\left(\mathcal{D}_c \geq \overline{\mathcal{D}}_c\right)\right)$ **then** $\mathcal{A}_c \longleftarrow \mathcal{S}_H$;
  ...
    /* *finalize the A-MSDU* */
  /* *send the aggregate (/frame)* */

Algorithm 1. The algorithm to configure the aggregation

Algorithm 1 takes up the principles defined in Fig. 4. An instance is executed for each service class of each station. The distribution enables to save computation resources. A class adapts hence its behaviour (aggregation or not, size threshold) each time it wins the medium, just before the transmission, regarding the states of all the higher priority classes' states (normal or critical). It introduces, however, a factor p in order to anticipate the saturation of the higher priority classes. It is better to target a percentage of the desired value in order to avoid to exceed it, since the system does not react instantaneously.

5. ALGORITHM EVALUATION

The algorithm has been included in a new wireless station model into the *Riverbed Modeler* simulator. Other parts of the model have also been modified in order to measure backlogs and delays. Such probes are computed each time a frame enters into the station and is transmitted. To increase the stability, variance might be taken into consideration like in *Active Queue Management* (AQM) mechanisms in addition to the last measurement.

5.1 Simple case

We considered the topology described above, with the service classes Best Effort and VOice configured with $\xi\,(500 \mid 50)$. The following results were computed for a backoff value fixed at $CWmin$ and AIFS and $CW_{\min}$ values as given in Table 4. In the *Riverbed Modeler* simulator, the backoff is, however, randomly drawn. 30 seeds have been done in order to minimize the influence of this backoff variation and we set the values of $CW_{\min}$ and $CW_{\max}$ in order to obtain average values around the ones defined in Table 4 (the backoff follows a uniform distribution).

The physical features of the Access Point remain the same, with a 65 *Mbps* bitrate. The internal memory of a station could be an only shared memory or composed of several memories associated to each class. We assume here that the memory is shared, but that it is possible

Table 4. AIFS and backoff values for a first evaluation

	Best Effort	*VOice*
Slot Time (ST)	9 μs	9 μs
AIFS (#ST)	3	1
$CW_{\min}$ (#ST)	31	7

to measure the backlog of each class. Each class has the same amount of the shared memory. From the maximum size proposed in the *Riverbed Modeler* simulator (128 000 bytes), we deduce the maximum backlog dedicated to each class (32 000). Finally, the anticipation factor is $p = 50\%$. Table 5 highlights the efficiency of the algorithm.

Table 5. Gain (%) with the algorithm for two classes

	average (%)	maximum (%)
BE backlog		-16
VO backlog		23
total backlog		-4
BE delay	-7	-5
VO delay	18	20

These results are encouraging, since the algorithm enables a 23% reduction of the *VOice* class backlog, when the *Best Effort* one only increases of 16%. Nevertheless, the total backlog is 4% higher: it is logical, because the *Best Effort* class temporarily stops the aggregation. For the *VOice* class, the average delay is reduced of 18% and the maximum one of 20%. But for the *Best Effort* class, the average delay increases of 7% and the maximum one of 5%. Proportionally, the improvement offered to the *VOice* class is greater than the degradation incurred by the *Best Effort* class.

5.2 Multiple classes

Let us use a second scenario with different flows:

- a *Best Effort* flow $\xi\,(300 \mid 90)$
- a *VIdeo* flow $\xi\,(500 \mid 50)$
- a *VOice* flow $\xi\,(100 \mid 125)$

The backoff is set to 15 slot times and AIFS to 2 slot times for the *VIdeo* flow. For the other flows, these parameters keep the values given for the previous example (Table 4). Table 6 gives the results for an anticipation factor respectively set to 50%, then to 10%.

Table 6. Gain (%) with the algorithm for three classes

	$p = 50\%$		$p = 10\%$	
	avg. (%)	max. (%)	avg.	max.
BE backlog		-72		-72
VI backlog		14		14
VO backlog		21		37
total backlog		-9		-9
BE delay	-56	-43	-65	-49
VI delay	26	26	45	31
VO delay	17	25	27	41

When it is 50%, the performances offered to the *VIdeo* and *VOice* flows are improved by 14% and 26%, both for the delay and the backlog. But the *Best Effort* class pay it dearly, since its backlog increases by 72%, and its average and maximum delays by about 50%. Finally, the

total backlog increases too, what is not interesting for the provider which prefers to minimize the size of the shared memory.

When the anticipation factor decreases to 10%, the results are nearly twice better (see Table 6). By reducing this factor, the algorithm is more frequently activated, and its benefits are cumulated. Furthermore, the total backlog and the *Best Effort* one remain identical. We only notice a small increase of the delays (about 7%).

5.3 Discussion

When the competition between the classes is managed with a simple strict priority scheduling, the algorithm activates the aggregation at the high threshold, on all the classes. No modification of the aggregation for the low priority classes is then efficient: they cannot access the medium while high priority classes' queues are not empty. From the provider point of view, the best (minimizing the backlog) is to keep this situation with all classes which aggregate independently of the clients' requirements (in case of shared memory). A possible enhancement is to find a compromise between both the clients and the provider requirements: since those of the clients are already taken into account, it is possible to add a condition to the algorithm in order to force the aggregation on all classes if the global backlog rises the limit. In case of dedicated classes' memories, it is possible to manage them differently: for example, one may be saturated without having drops for another.

Our algorithm is particularly advantageous when requirements are strong on upper classes and weak on lower ones: for example, if delays and backlogs are constrained for a *VOice* class when the *Best Effort* one accepts great delays and even drops.

6. CONCLUSION

When the access to the medium is competitive, aggregating enables to take advantage of gaining the medium and reduces the competition. The consequence is an increase of the backlogs and of the delays for the other stations, but we have demonstrated that it is largely compensated by a better yield. It leads to an algorithm which dynamically adapts the aggregation parameters for each service class, regarding its Quality of Service requirements and the ones of its competitors.

Further works may be launched in order to evaluate our proposal on real devices which nowadays provide access to these aggregation parameters. Finally, our works are in line with the spirit of the Software Defined Network concept whose objective is to control the network devices of the access providers.

REFERENCES

Azhari, S.V., Gurbuz, O., and Ercetin, O. (2016). Qos based aggregation in high speed IEEE802.11 wireless networks. In *2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 1–7.

Bhanage, G., Raychaudhuri, D., and Seskar, I. (2011). Backlogged queue based MAC frame aggregation. *Pervasive and Mobile Computing*, 7(4), 449–466.

Breck, D., Georges, J.P., and Divoux, T. (2019). Étude de performances par simulation d'aggrégation A-MSDU en 802.11n. Technical report, Université de Lorraine, CRAN. URL `http://www.cran.univ-lorraine.fr/jean-philippe.georges/wi20/simuls.pdf`.

Dely, P., Kassler, A., Bayer, N., Einsiedler, H.J., and Sivchenko, D. (2010). FUZPAG: A fuzzy-controlled packet aggregation scheme for wireless mesh networks. In *Fuzzy Systems and Knowledge Discovery (FSKD), 7th International Conference on*, volume 2, 778–782.

IEEE Computer Society (2009). IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications - amendment 5: Enhancements for higher troughput. IEEE Std 802.11n-2009.

IEEE Computer Society (2013). IEEE standard for information technology - telecommunications and information exchange between systems local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications - amendment 4: Enhancements for very high throughput for operation in bands below 6 ghz. IEEE Std 802.11ac-2013.

Karmakar, R. (2019). Online learning-based energy-efficient frame aggregation in high throughput WLANs. *IEEE Communications Letters*, 23(4), 712–715.

Karmakar, R., Chattopadhyay, S., and Chakraborty, S. (2017). Impact of IEEE 802.11n/ac PHY/MAC high throughput enhancements on transport and application protocols—a survey. *IEEE Communications Surveys Tutorials*, 19(4), 2050–2091.

Kosek-Szott, K. and Rapacz, N. (2020). Tuning wi-fi traffic differentiation by combining frame aggregation with txop limits. *IEEE Communications Letters*, 24(3), 700–703.

Kowsar, M.M.S. and Biswas, S. (2017). Performance improvement of IEEE 802.11n WLANs via frame aggregation in ns-3. In *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 1–6.

Machrouh, Z. and Najid, A. (2018). High efficiency IEEE 802.11ax MU-MIMO and frame aggregation analysis. In *2018 International Conference on Advanced Communication Technologies and Networking (CommNet)*, 1–5.

Saif, A., Othman, M., Subramaniam, S., and Hamid, N.A.W.A. (2012). An enhanced A-MSDU frame aggregation scheme for 802.11n wireless networks. *Wireless Personal Communications*, 66(4), 683–706.

Saldana, J., Ruiz-Mas, J., and Almodóvar, J. (2017). Frame aggregation in central controlled 802.11 WLANs: The latency versus throughput tradeoff. *IEEE Communications Letters*, 21(11), 2500–2503.

Selvam, T. and Srikanth, S. (2010). A frame aggregation scheduler for IEEE 802.11n. In *Communications (NCC), 2010 National Conference on*, 1–5.

Skordoulis, D., Ni, Q., h. Chen, H., Stephens, A.P., Liu, C., and Jamalipour, A. (2008). IEEE 802.11n MAC frame aggregation mechanisms for next-generation high-throughput WLANs. *IEEE Wireless Communications*, 15(1), 40–47.