International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Providing Delay Guarantees to Time-Sensitive Traffic by Scheduling using OFDMA with Deadline Estimation While Avoiding Best-Effort Traffic Starvation

Muhammad Inamullah[1*]

Submitted: 25/11/2020 **Revised:** 10/12/2020 **Accepted:** 15/12/2020

***Abstract*:** OFDMA is introduced to WLANs in the ax amendment of the IEEE 802.11 standard. It makes the WLAN a closed-loop system, where the AP assigns OFDMA resource units to the STAs for uplink transmissions. The scheduling algorithms generally optimize throughput by minimizing upload time. For time-sensitive traffic, the time-critical packets that miss their deadlines become useless, even if overall upload time is minimized. To provide schedules that prioritize the STAs for transmission whose traffic has earlier deadlines, the AP needs to know the packet deadlines in the queue of each STA. Since STAs do not provide any information on the expiration time of the packets they send to the AP, we give an algorithm to estimate on the AP side the deadlines of HOL packets waiting at STAs and make a scheduling decision that minimizes the number of packets dropped due to expired deadlines. In doing so, the best-effort traffic is also scheduled to prevent its starvation. Compared to the number of packet drops with the traditional approach of throughput optimization, our approach shows up to a six-fold reduction in packet drops due to missed deadlines at the STAs. Our approach also avoids starvation of the best-effort traffic.

***Keywords*:** deadline; 802.11ax; OFDMA; scheduling; WLANs; time-sensitive networking; best-effort traffic; resource allocation; starvation

1 INTRODUCTION

In time-sensitive networking, packets have upper bounds on the delay between their arrival into the queue and their reception at the destination. That is, packets, if received after a specified delay, have to be discarded. 802.11 wireless LANs (WLANs or Wi-Fi) employ contention-based channel access and hence are best-effort in that they try their best but cannot guarantee any deadlines. Orthogonal Frequency Division Multiple Access (OFDMA) is an important MAC mechanism of WLANs, both for uplink (UL) and downlink (DL) traffic since the IEEE 802.11ax amendment [1]. One of the most remarkable changes the OFDMA brings in is that during channel access, the access point (AP) remains in charge of deciding the access of each station (STA) to the channel, as opposed to the previous contention-based access, where each STA contends for the channel and does a random back-off if it is unsuccessful in finding the channel free. Now that OFDMA has removed the uncertainty in accessing the channel, we investigate whether Wi-Fi can handle delay-bounded traffic.

In OFDMA, Orthogonal Frequency Division Multiplexing subcarriers are grouped into a resource unit (RU). To provide access to STAs, the AP assigns one RU per STA, and the STAs transmit in their RUs parallelly. This is called scheduled UL OFDMA. The other is UORA (UL OFDMA random access), where the AP assigns a group of STAs an RU, and the STAs of the group contend for that particular sub-band. We explore the scheduled UL OFDMA because it provides channel access in a deterministic manner, making it a naturally better choice to handle time-bounded traffic.

With time-sensitive networking, receiving a packet on time is more important: the destination discards a packet on receiving it after its deadline. When AP schedules UL RUs, it optimizes some metrics. Traditionally, this metric is the throughput that the WLANs maximize by minimizing the overall upload time. Minimizing the upload time requires the STAs with better channel conditions (and thus higher MCS indices) to get scheduled before the

[1] *Department of Computer Engineering, Aligarh Muslim University, Aligarh 202002, INDIA*
** Corresponding Author Email: inamullah.m@gmail.com*

Nomenclature

ACK	Acknowledgement frame
AC	Access category
AP	Access point
BSR	Buffer status report
CTS	Clear to send frame
DCF	Distributed coordination function
DIFS	Distributed interframe spacing
EDCA	Enhanced distributed channel access
EDF	Earliest deadline first
HOL	Head-of-line
OFDMA	Orthogonal frequency division multiple access
RTS	Ready to send frame
RU	Resource unit
SIFS	Separate interframe spacing
STA	The stations or nodes sending or receiving data to and from the AP
TSN	Time-sensitive networking
QoS	Quality of service

STAs with poorer channel conditions (lower MCS values), regardless of whose packet has an earlier deadline. Therefore, in such cases, more packets of STAs with lower MCS indices will surpass their deadlines and get dropped than those of higher MCS-STAs. In the following, assuming that each STA has packets of only one traffic type, we use the phrases "the deadlines of the head-of-line (HOL) packets at each STA" and "the STA deadlines" interchangeably. We can easily extend our approach to such scenarios if there are packets of more than one traffic type.

Thus, in time-sensitive networking, instead of maximizing the upload time, saving the packets from being dropped due to expired deadlines is essential. The scheduling algorithm needs to decide how to assign RUs to STAs to minimize packets dropped due to missed deadlines. The well-known algorithm of such deadline-based scheduling guarantees minimum lateness [2], which is defined as the time elapsed for a packet between its reception and its deadline, summed across all dropped packets, and is directly proportional to the total number of packets all the STAs drop due to missed deadlines. If we give a STA whose HOL packet has the earliest deadline the first chance to transmit, we can minimize such packet drops. The algorithm seems trivial, but the fact is that there is no way to inform the AP about the deadlines of packets waiting for UL transmission in STA queues. AP may know the delay tolerance of the UL traffic, which is an attribute of the traffic defined by 802.11. If two packets of the same traffic type wait in the queues of two STAs, the older of the two will have an earlier deadline. As explained below, we use this knowledge of queue sizes, which can be supplied to the AP, to estimate the packet deadlines.

Buffer status report (BSR), which informs AP about the status of STA queues, is introduced in 802.11ax [1]. We present a scheduling algorithm that runs at

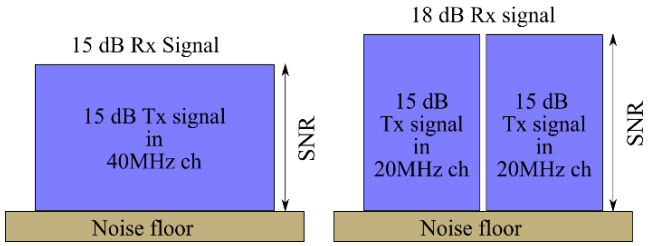


Figure 1: Two STAs transmitting in splitted channel get better SNR and MCS.

MCS Index	20MHz		40MHz	
	0.8µs GI Data rate (Mbps)	SNR (dB)	0.8µs GI Data rate (Mbps)	SNR (dB)
0	6.5	2	13.5	5
1	13	5	27	8
2	19.5	9	40.5	12
3	26	11	54	14
4	39	15	81	18
5	52	18	108	21
6	58.5	20	121.5	23
7	65	25	135	28
8	78	29	162	32
9	N/A	31	180	34

Figure 2: SNR Vs MCS for 20 and 40 MHZ channels.

the AP. The AP uses the value of the queue sizes that the BSRs from all STAs have reported to estimate the time a packet has spent in the STA's queue. The AP then uses this time to calculate the remaining time to the deadline for the HOL packet of each STA and then uses the estimated deadlines to select an RU assignment that results in the minimized total number of packets dropped by all STAs.

Our simulation results show that when the AP schedules as per our algorithm, there is up to six times the reduction in packet drops due to missed deadlines compared to when scheduling with the conventional approach, which minimizes total upload time.

The rest of the paper is organized as follows: Section 2 builds up the problem and provides the details on related work, and Section 3 explains the design of our algorithm based on our deadline estimation model. Section 4 presents our simulations and the results with proposed modifications. Section 5 concludes the work.

2 BACKGROUND OF THE STUDY AND PROBLEM STATEMENT

The benefits brought to WLANs through the introduction of OFDMA are numerous, like transmissions that are more resilient to frequency selective interference and fading and DL and UL multiuser (MU) OFDMA that allows simultaneous transmissions and receptions to and from multiple STAs, thus saving much of the wasted airtime of the distributed coordination function (DCF) of the 802.11 in the form of back-offs, DIFS, SIFS, ACKs, and (possibly) RTS/CTS. More importantly, in the context of this paper, when a channel is split into smaller RUs, the power spectral density of the received signal increases in the smaller RU if a STA keeps transmitting with the same signal power, which allows the STA to use higher MCS in that RU. This is illustrated in Figure 1. This allows the AP to receive more data in UL transmissions from multiple STAs sending in smaller RUs than from a single STA sending in the whole channel. The same effect is elaborated in Figure 2. For example, a STA transmitting in a 40 MHz channel at 18 dB SNR gets a data rate of 81 Mbps. Now, if the channel is split into two 20 MHZ channels and the two STAs transmit at 18 dB SNR each in each of the two channels, the transmission will jump to MCS index 5, allowing the STAs to achieve a data rate of 52 Mbps each so that the combined data rate reaches 104 Mbps. The arrows show the effect in the figure. Earlier works have used this effect to design scheduling algorithms that pick the best RU configuration from a list that optimizes the overall upload time [3]. Below, we explain the meaning of RU configuration, which we will also use in our scheduling algorithm.

An RU configuration is the set of RUs obtained after splitting a channel according to the 802.11ax standard. For example, we show the permissible ways of splitting a 20 MHz channel in Figure 3. One RU configuration on splitting the channel is {RU106, RU106, RU2}, where RU *i* means an RU consisting of *i* tones. Recursively splitting this RU configuration one more time results in {RU106, RU52, RU52, RU26} – another RU configuration. Here, we assume that there is no frequency selective fading, and hence, relative positions of RUs within an RU configuration can be interchanged. That is, for example, from the RU configuration {RU106,

RU106, RU26}, splitting either the right RU106 or the left RU106 into two RU52s results in the same sorted RU configuration: {RU106, RU52, RU52, RU26}. We split the channel in all possible ways, sort the resulting RUs with the widest RU first, and keep the distinct RU configurations. Splitting in this way, for example, gives nine possible configurations for the 20 MHz channel, excluding the {RU242} that corresponds to the entire channel. These are shown below (numbered 0-9).

0. {242}

1. {106, 106, 26}

2. {106, 52, 52, 26}

3. {106, 52, 26, 26, 26}

4. {106, 26, 26, 26, 26, 26}

5. {52, 52, 52, 52, 26}

6. {52, 52, 52, 26, 26, 26}

7. {52, 52, 26, 26, 26, 26, 26}

8. {52, 26, 26, 26, 26, 26, 26, 26}

9. {26, 26, 26, 26, 26, 26, 26, 26, 26}

Using the channel-splitting rules of 11ax, we can obtain similar configurations for 40, 80, and 160 MHz channels.

Optimizing the overall upload time is appropriate for the best effort traffic, which was earlier the sole traffic on Wi-Fi, where packet delay bounds are not considered. Honoring the packet deadlines is essential in time-sensitive networking. Since the medium is accessed in Wi-Fi through the contention-based protocol, providing delay guarantees for packets was impossible. With OFDMA, the closed-loop scheduled access has arrived, and therefore, it is now possible to consider deadlines. We will consider this problem in detail next. We discussed in detail in Section 1 that the problem of providing delay guarantees translates into minimizing the number of packets dropped due to missed deadlines and that if we minimize the total upload time resulting from an RU assignment, it does not necessarily result in minimizing the total number of drops due to missed deadlines: Giving STA A the chance to transmit before B merely because A has a higher MCS would result in more packet drops at B if B has an earlier deadline than A. Such packet drops due to missed deadlines at B can be reduced if B transmits before A in this case.

To schedule according to deadlines, the AP needs to know the deadlines for UL packets, as discussed in Section 1. In EDCA, as described in 802.11 specifications, the UL traffic does not include delay information. Though the Delay Bound parameter is available in the TSPEC Information Element of 802.11 ADDTS action frame, it is recommended to be ignored when EDCA is in use [4]. Therefore, it becomes necessary for the AP to have a procedure to estimate the deadlines of HOL packets waiting in the queues of STAs.

The delay information is not available to the AP, but the AP gets the queue size of each STA through the BSR that a STA sends, which is introduced in 802.11ax. The HT Control field in the MAC header of various Wi-Fi frames like QoS Data, QoS Null, Management, and Control Wrapper frames carries the BSR. Among other information about a STA's queue, a BSR also reports the queue size of the highest access category (AC) traffic waiting at the STA. A STA can handle multiple ACs and their delay bounds by reporting the queue sizes of various ACs in different BSRs, and has the option of deciding the priority of an AC over other ACs.

In the next section, we propose our algorithm to estimate the deadline of an HOL packet based on queue sizes. Our scheduling algorithm, then using the estimated deadline, decides the assignment of RUs to STAs such that the number of packets dropped from the queue due to expired deadlines is minimized. Once the AP has assigned RUs to STAs, it sends the OFDMA schedule to the STAs either in a Trigger Frame, a new control frame in 11ax, or in the header of a DL data frame.

Next, we discuss some related work in OFDMA scheduling. OFDMA in 802.11ax networks is the next logical step of OFDMA in LTE. Scheduling in LTE networks is reported in many research papers.

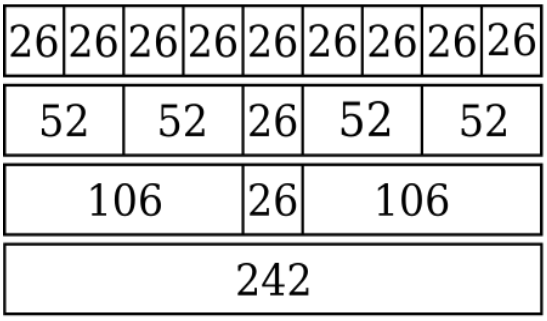


Figure 3: Proper way of splitting 20 MHz channel according to

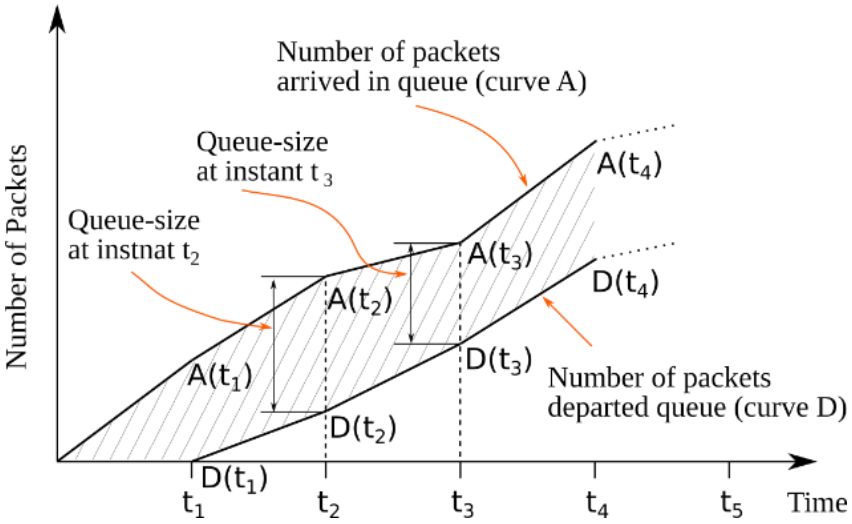


Figure 4: Packet arrivals and departures in a queue. Shaded area shows the time-averaged queue-size.

Resource blocks are assigned to the UE with the best CQI such that the highest throughput is achieved by BestCQI [5]. As many groups of RBs are made as there are UEs, and are assigned to the UEs by RR [6]. Throughput or delay guarantees are not provided as there is no optimization. UE with the maximum value of the metrics (generally the throughput) among the unassigned RBs is searched, and multiple contiguous RBs are allocated to the UE in FME and RME [7]. An RB allocation is searched that takes care of the long-term service rate of the UE, in addition to the channel condition, so that throughput, as well as fairness, are maintained in PF [8] and ODM [9] is somewhat in between PF and BestCQI in that a choice is made between throughput and achievable rate for a UE and RBs are allocated to UEs with good channel conditions while starvation of the poor channel condition UEs is averted. Scheduling based on the packet reception deadline is not considered in any of the discussed works.

A metric called Degree of Urgency is devised for urgency-based fair scheduling (UFS) [10], which is based on HOL packet delay. RBs are allocated to UEs in the order of highest HOL delay first, assuming that the eNodeB knows the delay of HOL packets. The number of RBs previously allocated to a UE is also considered, and the packet loss and fairness are improved.

In the theoretical scheduling problem of [11], the flow whose HOL bit has spent the most time as an HOL bit, referred to as the HOL access delay, is scheduled first, resulting in throughput-optimal scheduling. This needs the knowledge of the time a bit becomes HOL and will primarily work with downlink flows unless there is some reporting mechanism to know this delay value from UE (or STA) to the eNodeB (or the AP).

Though [10] and [11] consider HOL delay for scheduling, these algorithms cannot work in 802.11ax because the scheduling algorithm at the AP does not know the time the packets have spent in the STA queues. To the best of our knowledge, our work is the first to estimate the deadlines of packets waiting in STA queues using the knowledge of temporal changes in STAs' queue sizes.

A greedy and recursive algorithm of splitting the channel into RUs for 802.11ax to maximize the total rate is given as a solution to the generalized scheduling and resource allocation problem by [12]. Again, the delay bounds for the traffic are not considered.

With OFDM, the channels can be assumed non-frequency selective, which makes the search space of RUs with such channels considerably smaller than with frequency-selective channels. Exploiting this fact, [3] use a Hungarian algorithm to match STAs with RUs to find an RU allocation that maximizes the data the STAs can upload parallelly in the current OFDMA slot, thus coming up with a schedule that minimizes the total upload time. This approach cannot work when the traffic is delay-bounded because a STA with an earlier deadline but poorer channel condition than another STA will not get an RU in the current schedule and will experience a high packet drop rate of the delay-

bound traffic. Our approach is compared to this work in Section 4.

3 ESTIMATING THE DEADLINE AND SCHEDULING WITH OFDMA

In time-sensitive scheduling, a packet that has surpassed its deadline is useless and, hence, dropped. Thus, deadline-based OFDMA scheduling aims to minimize the number of such packet drops. This can be achieved by assigning an RU configuration to the list of STAs sorted according to deadlines so that the STA with the earliest deadline gets the largest RU in the RU configuration. The sorting needs to know the deadlines, but the AP does not know the exact per-packet deadlines nor the time they have spent in their queues. So, the AP needs to estimate the deadlines based on its knowledge of the delay tolerance of the AC of each STA's traffic. A lax strategy would be to set the deadline to the sum of the delay tolerance of the HOL packet's AC and the current time, updating whenever the current time exceeds the maintained value of the deadline. We will refer to this method as LAX.

For the AP to be able to estimate the deadlines better, it needs to adjust the deadlines according to the time a packet has spent in the queue (that is, the queueing time of the HOL packet), which might be different even for two STAs sending traffic of the same AC. The AP neither knows this time nor the instant when the packet has entered the queue. The STAs only report their queue sizes to the AP.

Thus, to estimate the deadline, let us find the time a packet has spent in the queue. Little's theorem [13] relates the average delay T of a packet in the system (the STA queue), the arrival rate λ of packets, and the average number of packets N in the system as:

$$N = \lambda T \tag{1}$$

Little's theorem assumes the system to be *ergodic*, which means that the system can go from any state to any state irrespective of the choice of initial state (i.e., in our case, whether the queue was initially empty or not), it can go to state 0 (i.e., the state with empty queue) infinitely often, and the state the system is in does not depend on the number of time steps. The theorem, however, does not make any assumptions on network topology, arrival process (i.e., how the packets arrive), service order (i.e., in what order the packets leave the queue), or service time distributions (i.e., how much time the packets spend in the queue). This means that Equation 1 can be used to estimate the average delay if the average number of packets and the arrival rate in the system at any particular time can be known. The time-averaged number of packets in the queue in Equation 1 up to time t is given as:

$$N = \frac{1}{t}\int_0^t N(t)dt \tag{2}$$

To derive a discrete expression for N equivalent to Equation 2, consider Figure 4; curve A shows the cumulative arrivals of packets in a STA's queue till time t, which, for simplicity, can be obtained by joining the points $A(t_i)$ (cumulative arrivals till time t_i) and $A(t_{i-1})$ (cumulative arrivals till time t_{i-1}) assuming that packets arrive with a constant rate within a slot. That is, we ignore the intermediate arrival pattern within the slot. Similarly, curve D shows the total number of packets that departed the STA queue till time t where we join consecutive points considering only the values at slot boundaries. The area of the shaded region then gives N.

The time-averaged number of packets in the queue given by Equation 2 is the sum of the time-weighted number of packets in each slot divided by time. Weights here are the variable lengths of the OFDMA slot. The area of the trapezoid in the slot $[i, i-1]$, bounded on left and right by the queue sizes reported in BSR by the STA at t_i and t_{i-1} and by the segments of curves A and D on top and bottom gives the time-weighted number of packets in any slot. Hence, the time-weighted average number of packets in the queue up to OFDMA slot i is:

$$N_{avg}(i) = \frac{1}{t_i}\sum_{s=1}^{i}\frac{1}{2}(Q(t_s) + Q(t_{s-1}))(t_s - t_{s-1}) \tag{3}$$

where $Q(t_i)$ is the queue size reported by the STA at t_i. The average arrival rate at t_i is the cumulative arrivals till time t_i divided by t_i, and is given as:

$$\lambda = \frac{A(t_i)}{t_i} \tag{4}$$

Using Equations 3 and 4 in Equation 1, the average packet delay in the system at slot i is given as:

$$T_{avg}(i) = \left(\frac{1}{2A(t_i)}\right)\sum_{s=1}^{i}\big(Q(t_s) + Q(t_{s-1})\big)(t_s - t_{s-1}) \tag{5}$$

The AP deduces the instantaneous value of cumulative arrivals at a STA ($A(t_i)$) from the

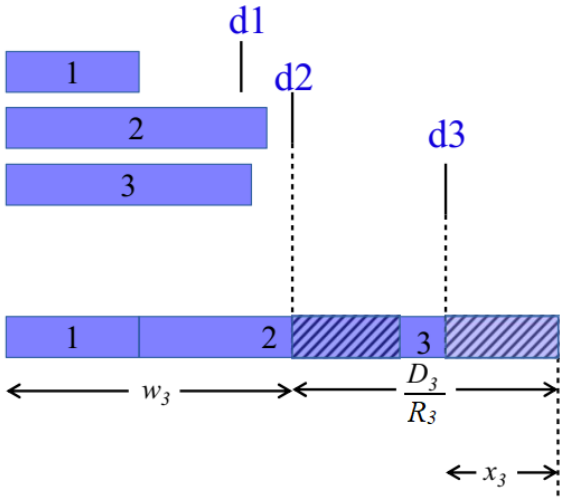


Figure 5: Drop-time without using OFDMA. Blocks 1, 2, 3 are the times required for transmission of a packet each of STAs 1, 2, and 3, respectively. Shaded regions are drop-times of STAs 2 and 3 respectively.

current queue size reported by the STA to the AP in BSR ($Q(t_i)$) and the number of packets received by the AP so far ($D(t_i)$). It then calculates the average delay experienced by the packets in the STA's queue using Equation 5. The AP subtracts the delay from the deadline value it maintains for the STA and gets an estimate of the remaining time until the deadline. The AP similarly maintains the deadlines for each associated STA.

The AP does not know the number of packets dropped at a STA; it bases all its calculations on the queue sizes the STAs have reported and the number of packets it has received. Thus, the arrival rate value in Equation 5 is slightly underestimated. Though this approach is explained for one traffic category, our approach can be extended to more than one type of traffic by making the AP maintain per-TID deadlines for each STA.

The AP sorts all the STAs associated with it in the earliest deadline first order using the estimated deadlines of the HOL packets of the STAs and finds which RU configuration will result in the minimum number of packets dropped by all the STAs that are assigned RUs from the RU configuration.

To select a particular RU configuration for assignment to the STAs, we first need to find, for each RU configuration, how many packets would drop when assigned. Thus, given the deadlines and the allowable MCS indexes for each STA, we find the total number of possible packet drops due to missed deadlines that we want to minimize for each RU configuration. First, let us derive the case without OFDMA. This is mostly what we discussed in [14], with detailed elaboration with the help of figures and more rationale behind the quantities used to derive the drop-time.

Channel is assigned to STAs in the earliest deadline first (EDF) order. A STA i needs a time $t_i = D_i/R_i$ (the STA upload time) to transmit its full queue with D_i amount of data in its queue at a link rate R_i but will have to drop the remaining queue at deadline d_i. (with the assumption that all packets in the queue have the same deadline and get dropped at once when the deadline expires). If allowed to continue the transmission after its deadline, the STA would take an extra $D_i/R_i - d_i$ time to transmit its otherwise dropped data. The number of packets dropped is then proportional to $D_i/R_i - d_i$, which we term as the drop time and define as the time to complete a transmission without considering the deadline minus time-to-deadline. Thus, instead of minimizing the number of packet drops, we can equivalently minimize the drop time.

Considering other STAs in the list, we need to add to the transmission time the waiting time for each STA i as follows, depicted in Figure 5.

$$x_i = \max\left(0, w_i + \frac{D_i}{R_i} - d_i\right) \qquad (6)$$

where the waiting time, since STAs transmit one after the other, can be recursively defined as either the deadline of the previous STA or the time when

the previous STA completes its transmission, whichever is earlier. This is given as:

$$w_i = \begin{cases} 0, & i = 1 \\ \min\left(d_{i-1}, w_{i-1} + \frac{D_{i-1}}{R_{i-1}}\right), & i \geq 2 \end{cases} \quad (7)$$

Now consider OFDMA, with an OFDMA slot duration τ. We split the channel according to the method discussed in Section 2 and find for each RU configuration the possible waiting time and drop time for each STA when STAs are assigned RUs for the duration τ, as follows. Let a picked RU configuration consist of a sorted list of m RUs. RU 1 is assigned to STA 1, RU 2 to STA 2, ..., RU m to STA m, in that order, so the first m out of n STAs ($m \leq n$) get an RU. Note that our assignment is not bipartite matching because the AP keeps the STA list also sorted according to the estimated deadlines, and we only assign the first m STAs to m RUs of the RU configuration in question. Each STA i transmits at a rate R'_i, where R'_i is the rate of STA i in RU i, in parallel with other STAs in the OFDMA slot, possibly with a higher sum rate across all RUs than the rate of the whole channel (as explained in Section 2). STA i transmits $R'_i \cdot \tau$ amount of data in the OFDMA slot out of its queue size D_i, and the further time needed to upload the remaining data is $(D_i - R'_i \cdot \tau)/R_i$. Maximizing the value $R'_i \cdot \tau$ minimizes the time to upload the remaining data for STA i and consequently minimizes the STA's drop time and the waiting time of STAs that follow STA i. The drop time with OFDMA then changes from x_i to

$$\overline{x_i} = \left(0, \overline{w_i} + \frac{D_i - R'_i \cdot \tau}{R_i} - d_i\right) \quad (8)$$

Where the waiting time with OFDMA for each successive STA i calculated recursively is:

$$\overline{w_i} = \begin{cases} \tau, & i = 1 \\ \min\left(d_{i-1}, \frac{D_{i-1} - R'_{i-1} \cdot \tau}{R_{i-1}} + \overline{w_{i-1}}\right), & i \geq 2 \end{cases} \quad (9)$$

Parallel transmission in the OFDMA slot reduces each STA's waiting time and drop time compared to the non-OFDMA case. This is explained as follows. Waiting time w_i depends on whether each STA i, i = *1, 2,...,* i-1 has completed its upload within its deadline or has continued the transmission till its deadline, at which time its remaining queue is dropped. Thus, with OFDMA, when all STAs 1, 2, ..., i - 1 complete their uploads before their deadlines, the maximum reduction in w_i in the best case is $\left(\sum_{k=1}^{i-1} \frac{R'_k \cdot \tau}{R_k}\right) - \tau$, and subsequently the best-case reduction in drop time for STA i is $\left(\sum_{k=1}^{i} \frac{R'_k \cdot \tau}{R_k}\right) - \tau$. In the worst case, when each of the STAs 1, 2, ..., i-1 consumes full time till its deadline, the reduction in drop time of STA i is $(R'_i \cdot \tau / R'_i) - \tau$. Note that since the sizes of the RU configurations in the number of RUs differ, it is possible in a particular UL MU transmission that some STAs at the end of the sorted list do not get any RUs.

We use the deadlines thus estimated and the drop times in our UL MU OFDMA scheduling algorithm, which we describe next. In the first step of our UL OFDMA scheduling algorithm, the AP collects queue sizes from the BSRs of all STAs, calculates the delay of each STA's HOL packet using Equation 5, and estimates the deadlines by subtracting the delays from all STAs' deadline values it maintains. The AP then sorts the STAs according to the deadlines. In the second step, the AP uses Equation 9 to calculate the drop times each RU configuration of m RUs would result in when assigned to the first m STAs from the list. The AP picks the RU configuration that results in minimum total drop time across all STAs.

The sorting of n STAs needs O (n log n) time. The second step performs an exhaustive search for the best RU configuration, and thus, its time complexity is equal to the number of possible RU configurations M. Considering our discussion in Section 2 about splitting the channel, M is equal to the total number of allowed ways a channel can be divided, removing the identical RU configurations. Thus, the worst-case complexity is O (n log $n + M$).

4 SIMULATION AND RESULTS

This section evaluates our OFDMA scheduling that works with our deadline estimation algorithm. The algorithm consists of two parts, as explained in Section 3: In the first part, the AP estimates the deadlines based on the queue size information, and in the second part, the AP uses the estimated deadlines to sort the STAs and assigns an RU configuration that minimizes the drop times. Accordingly, our evaluation also goes in two parts: 1) we understand how much the deadlines estimated with our deadline-estimation method conform to the actual HOL deadlines, 2) we evaluate the performance of the scheduling algorithm in terms of

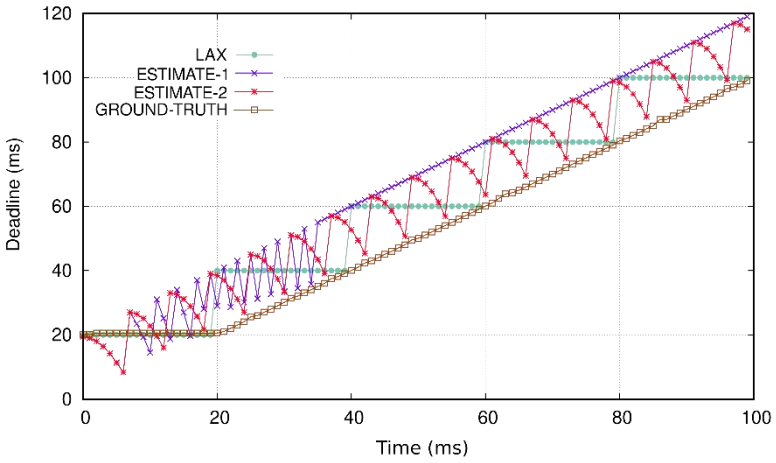


Figure 6: Estimated deadlines with three approaches. The fourth (GROUND-TRUTH) is for comparison.

number of packets dropped at STAs when the AP runs it to schedule RUs, and the STAs send their UL traffic in their allotted RUs, while also dropping any packets that cannot be sent within their deadlines. We take a simple one STA - one AP network for the first part of the evaluation. The traffic model we take is simply CBR traffic with the goal of keeping the STA's buffer full all the time so that packets experience queuing delay. This goal is achieved by keeping the arrival rate higher than the link rate. We set the delay tolerance value at 20 ms, a typical value for IoT traffic. Every packet that enters the queue at the STA has a deadline associated with it, which is set equal to the current time plus delay tolerance. We refer to the deadline value of the HOL packet in the queue as the actual deadline. The AP maintains a deadline variable for the traffic coming from the STA, initializes it to the current time plus the delay tolerance of the STA's traffic type, and updates the value by subtracting the delay using Equation 5 whenever it receives a BSR. When the current time exceeds the current value of the deadline variable at AP, the AP reinitializes the variable again to the current time plus delay tolerance value. At this time, it also infers that the packets are dropped in the STA's queue due to the passing of the deadline and resets all the variables, such as the previous queue size and the arrival rate till now, used in Equation 5 to estimate the deadline.

Figure 4 shows, as time passes, how the values of the deadline estimated by AP with our estimation algorithm (depicted as ESTIMATE-1) compare with the actual deadline value of the HOL packet (denoted GROUND-TRUTH). There are two more variants. In LAX, the AP does not consider the time the packet has spent in the queue to modify the deadline at AP; it simply re-initializes the deadline value whenever the current time surpasses the stored value of the deadline.

ESTIMATE-2 is the same as ESTIMATE-1, with a slight variation in that the variables are not reset when the deadline is reinitialized. Thus, ESTIMATE-2 bases its estimation on previous values of queue sizes, while ESTIMATE-1 calculates the estimates anew every time the deadline is re-initialized. The curves show that ESTIMATE-1 tries to converge towards the GROUND-TRUTH more often. In contrast, ESTIMATE-2, after initially showing some fluctuations, steadily follows the GROUND-TRUTH with a constant offset, thus maintaining a fixed error in the estimated deadline. We will return to their effect on scheduling performance when we analyze our results after discussing the following second and central part of the evaluation.

4.1 Simulation with custom simulator and results:

To evaluate the performance of our OFDMA scheduling, we set up a 20-node network, where each node wants to send UL traffic to the AP. For now, we assume all nodes can transmit at MCS 8. This gives a maximum link rate of 97.5 Mbps in 20 MHz channel. Later, we will see the performance by varying the MCS values. We run the simulations for one second, and each simulation is divided into 0.1 ms ticks, which are used to synchronize various simulation events like packet arrivals, sending, and drops. One second simulation time gives 10,000

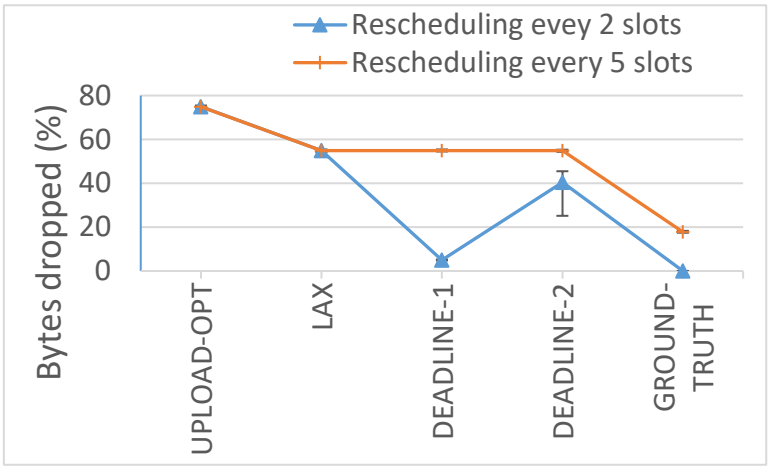


Figure 7: Fraction of bytes dropped due to missed deadlines when delay tolerance of the AC is 5 ms, with different scheduling intervals for full buffer traffic.

ticks, a sufficiently large value for observing the behavior of various events.

We evaluate two different traffic models. The first traffic model is full buffer UL traffic from each node, where we want to determine how different delay tolerance values and BSR periods affect the packet drop. The second is the real IoT traffic model, where each node generates the traffic as specified in [14].

1) Results with full buffer traffic model, varying the delay tolerance and BSR period: We set our first traffic model as follows. All the nodes generate the same traffic, with packets of mean size 200 B arriving with a mean interarrival time of 0.1 ms, thus ensuring that, on average, one packet arrives every tick. This traffic model ensures a full buffer at each STA by setting the arrival rate well above the link rate (97.5 Mbps) of the whole channel. (Note that the sum of rates of RUs after splitting the channel might be larger than the whole channel rate, so we need to keep the arrival rate accordingly high). The mean packet size is arbitrary and can be anything that, coupled with the interarrival rate, can keep the STA queues full. As described below, all nodes have the same delay tolerance and BSR period values, which we vary for different simulations. Figure 7 gives the percentage of total bytes dropped due to missed deadlines across all STAs out of the total packets sent for two different rescheduling intervals

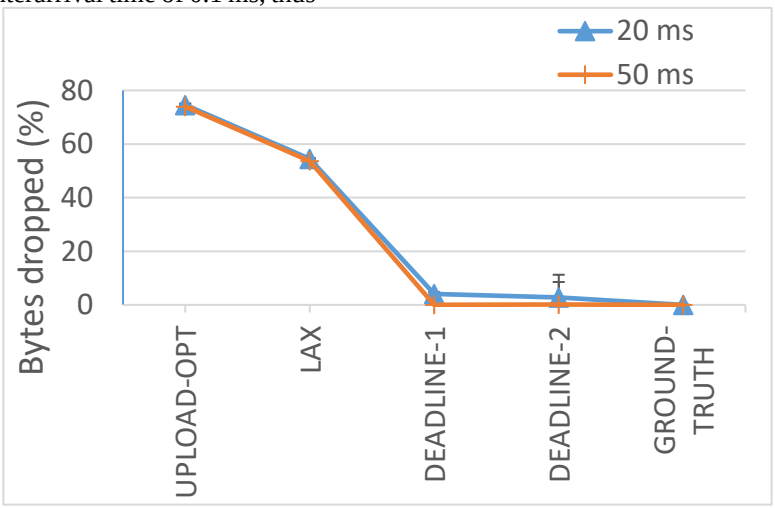


Figure 8: Percentage of bytes dropped due to missed deadlines, for 20 ms and 50 ms as AC's delay tolerance, when the AP sends a new schedule after every 5 OFDMA slots for full buffer traffic.

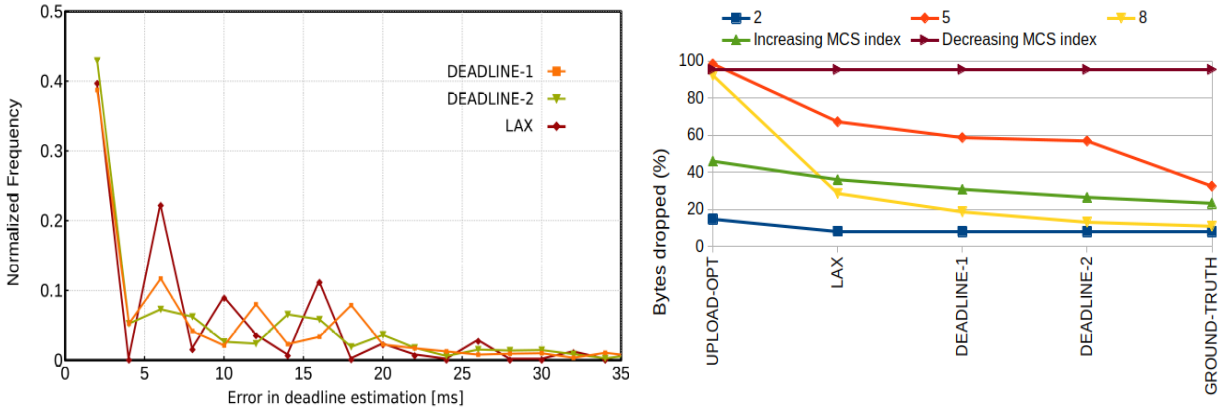


Figure 9: IoT Traffic: (a) PDF of magnitude of error in deadline estimation, and (b) Fraction of bytes dropped due to crossed deadlines, across different MCS schemes.

for all traffic having the same delay tolerance. Similarly, Figure 8 gives the bytes dropped for two values of delay tolerance for a fixed schedule update period. DEADLINE-1 and DEADLINE-2 show the results for our algorithm with the two methods of deadline estimation, ESTIMATE-1, AND ESTIMATE-2, respectively, as discussed above. GROUND-TRUTH shows, for comparison, the best possible performance when we separately provide the AP with the ground truth – the actual deadlines of HOL packets in STA queues. Note that such provision of deadline values to the AP is only possible in our custom simulator but not in the real world. The leftmost UPLOAD-OPT is our implementation of [3], for which the goal of scheduling is to minimize the total upload time of the traffic and does not consider packet deadlines.

Figures 9a and 9b show the median percent of packets dropped (with error bars showing the minimum and maximum values) each for 100 runs of simulations with the traffic described above, with delay tolerance values of 20 ms and 50 ms, respectively. With more considerable delay tolerance values, our algorithms perform the same way as GROUND-TRUTH. Then, in Figure 6, we reduce the delay tolerance value to 5 ms. Though still better than UPLOAD-OPT, our algorithms seem to perform similarly to the LAX when OFDMA scheduling (i.e., assignment of RUs as per the deadline-based algorithm) is done every 5 OFDMA slots (Figure 7). When we reduce re-scheduling periodicity to 2 OFDMA slots, DEADLINE-1 outperforms LAX and DEADLINE-2. Thus, our proposed algorithms can reduce the number of packets of the delay-bound traffic that miss their deadline and get dropped, compared to when scheduling is done merely to minimize the overall upload time. This happens better when OFDMA is rescheduled frequently with a period less than the delay tolerance of the traffic being scheduled.

2) Results with factory IoT traffic model: Next, we describe simulation where ten nodes send real factory IoT traffic, with delay tolerance values from as low as 0.5 ms to as high as 200 ms, and low data rates ranging from 40 Kbps to 4.6 Mbps, as per the specifications in [15]. Apart from IoT traffic, we also include ten other nodes to send video, voice and best-effort traffic in the form of VoIP, video conferencing and FTP traffic. The VoIP is a low rate traffic (around 13 Kbps) where packets have a delay tolerance of 50 ms. The video conferencing packets are bigger (around 7 KB) but have no delay requirements, and the traffic has a data rate of 2 Mbps. FTP traffic is a single big file upload and does not have any deadline, while data e rate may reach up to 16 Mbps. Detailed specifications can be found in [14] and [16]. Our simulation aims to evaluate the performance of our scheduling algorithm in a typical factory setting, where the same Wi-Fi supports the factory IoT traffic and the employees' internet browsing traffic. Again, we use 20 node network with each node transmitting at MCS 8, but this time, 14 nodes send delay-bound traffic, and six nodes send FTP and video conferencing traffic that is not delay-sensitive, as described in detail in [14-16]. The simulation is divided into time ticks of 0.1ms.

Packets following a lognormal distribution to select their sizes with the mean values given for each application arrive at each STA with an inter-arrival time following an exponential distribution for each STA [14]. The packets also possibly get dropped from the front of the queue at every tick if they have surpassed their deadline. Packets are sent in every OFDMA slot, with the OFDMA slot taken as 1 ms. In this simulation, BSR reports are sent every 5 OFDMA slots.

Figure 9 shows the bytes dropped by all the nodes with delay-bound IoT traffic by employing various scheduling algorithms for various MCS indexes.

Analysis and Explanation of Results:

To understand how our deadline estimation algorithms affect the packet drops, we show the PDF of the absolute error value in deadline estimation (Estimated Deadline minus Actual Deadline) for the above simulation in Figure 9a. The PDF graphs show that error magnitudes are more prevalent at specific values for LAX (indicated with higher peaks), whereas curves are smoother for the two variants of our algorithms. Our algorithm takes on intermediate estimated values before updating the deadline to the next value. In contrast, LAX jumps to the next deadline value whenever the current time crosses the deadline. This means that the LAX's estimation of the deadline deviates more often for some STAs than others, resulting more often in incorrect sorting of STAs by the scheduling algorithm.

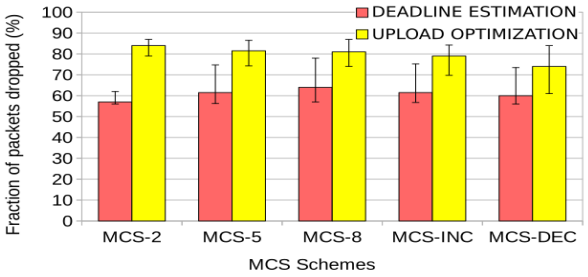
We also want to see how our proposed algorithm performs with different channel conditions. We investigate 5 MCS schemes: 1) *MCS-Increasing*: STAs use MCS values from 2 to 9, some with the same MCS value. The STAs with most stringent delay constraints have the lowest MCS, 2) *MCS-Decreasing*: STAs are assigned MCS in decreasing order. Those STAs with the earliest deadlines have the best channel conditions, 3) MCS-2, 4) MCS-5, and 5) MCS-8, where all STAs transmit at MCS 2, 5, and 8, respectively. Figure 9b shows these results, consistent with our previous finding, that our algorithm performs very close to the GROUND-TRUTH. In each case, the number of packet drops due to the deadline with our algorithms is less than that with the UPLOAD-OPT algorithm.

For *MCS-Decreasing*, the STAs with the earliest deadline have the best MCS (9). This renders the minimization of drop time to the minimization of upload time; hence, the results are the same for our algorithms and the UPLOAD-OPT algorithm. The reason lies with the selection of RU configuration that, the algorithm decides, would give the smallest drop-time, as discussed in Section 3. With this MCS scheme, and also for scheme MCS-5, the RU configuration our algorithm picks is {RU106, RU106, RU26}. The algorithm splits and moves to a higher RU configuration only if the resulting RUs while transmitting at the same transmit power in each RU, allow for higher MCS, which is not always the case, as can be seen by examining the 802.11ax SNR-MCS table [17]. Thus, the selected RU configuration, which our simulation selects most of the time, allows only the first three STAs in the sorted list, therefore making all other STAs miss deadlines and drop data.

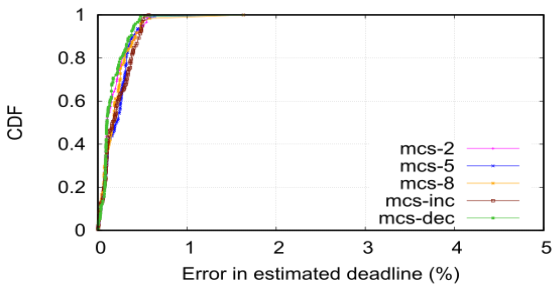
A similar observation is made with MCS-5. The scheduling algorithm chooses the same RU configuration as above more frequently. Still, it occasionally chooses other RU configurations with more RUs this time. Therefore, the packet drop rate remains higher than other MCS schemes, as shown in Figure 9b.

Simulations with our implementations of deadline estimation perform better than UPLOAD-OPT. But the performance is always worse than GROUND-TRUTH across all MCSs. This is because our estimation sometimes deviates from the actual value due to the AP's dependence for estimation solely on the queue size the STA reports in BSR. The final value of the queue size visible to the AP is the number of packets arrived minus the sum of the packets dropped and sent by the STA, but the AP separately does not know how many packets the STA has dropped. Thus, based on the queue size it sees, the AP underestimates the number of arrivals and, hence, the arrival rate at STAs, which subsequently results in an error in the estimation of the average delay of packets in the STA queue.

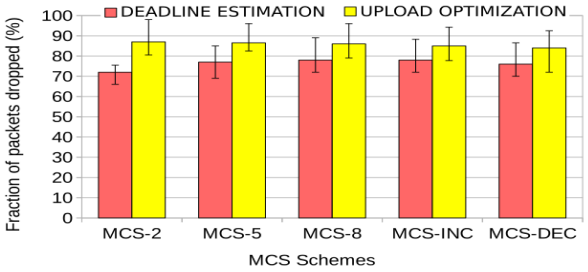
Another observation worth mentioning from our simulations of random full-buffer (Figures 7 and 8) and IoT (Figure 9) traffics is that there is a trade-off between DEADLINE-1 and DEADLINE-2 variations. DEADLINE-1 works better when deadlines are identical across the competing traffic flows (for example, in Figures 7 and 8, we select only one of 5, 20, and 50ms delay tolerance values for all the nodes). At the same time, DEADLINE-2 is superior (Figure 9) when the deadlines are diverse



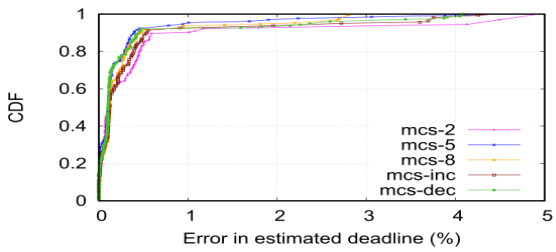
(a) Maximum delay 20 ms for each STA.



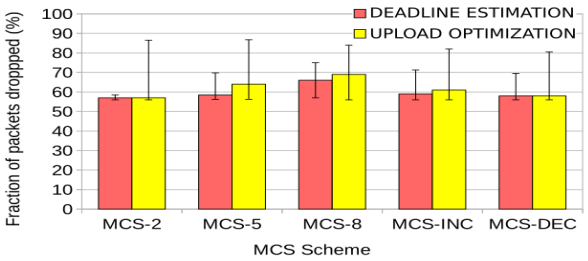
(b)



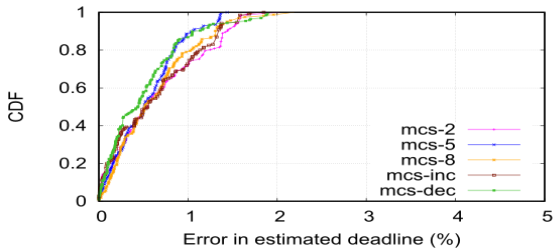
(c) With small maximum delay (4 to 6 ms) for each STA.



(d)

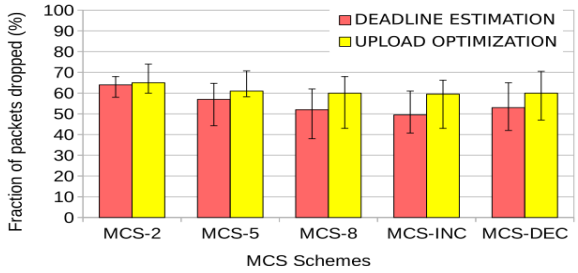


(e) With big maximum delay (40 to 60 ms) for each STA

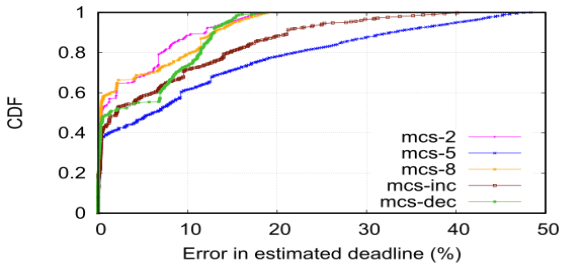


(f)

Figure 10: ns-3 simulations with full-buffer UL traffic: Fraction of packets expired (a, c, and e), and CDF of error in deadline estimation (b, d, f) for a particular run of the bars of plots a, c, and e, respectively.



(a) Fraction of packets expired.



(b) CDF of percentage error.

Figure 11: ns-3 simulations with IoT traffic: Packets dropped and CDF of percentage error

across the flows. To understand why, let us revisit Figure 6. The DEADLINE-1 follows the GROUND-TRUTH deadline curve with a constant offset after time 0.04 ms. This means that when all the flows have the same delay tolerance values, either of the two deadline estimation methods, DEADLINE-1 or GROUND-TRUTH, gives the same ordering in the sorted list of STAs after 0.04 ms. This makes the

results obtained with the two methods very close in Figures 7 and 8. On the other hand, the deadline curve ESTIMATE-2 in Figure 6 repeatedly takes intermediate values between ESTIMATE-1 and GROUND-TRUTH, due to which it becomes more suitable for the cases when traffic flows have diverse delay tolerance values, and the AP gets the estimate of the deadline of each flow closer to its actual values, as shown in Figure 7.

Reserving the RUs for STAs that have deadlines may cause starvation of others with no deadline. We simulate the latter as STAs with huge deadlines. Figure 10 shows that the six no-deadline STAs could send little or no data; in the same simulation, we have shown the packet drops for the delay-bound traffic nodes in Figure 7. DEADLINE-2 variation of our algorithm allows sending some no-deadline traffic, as does the GROUND-TRUTH. This happens because our algorithm and the GROUND-TRUTH allow the STAs to send most of the deadline-bound traffic on time, which causes the queues of STAs having deadlines occasionally empty, giving some chance to no-deadline STAs. Starvation can be easily handled if we allocate one or a few RUs for random access and allow the STAs with best-effort (no deadline) traffic to contend for the channel as usual within the random access RUs.

4.2 Ns-3 Implementation and Results:

We implemented the proposed OFDMA scheduling scheme and the deadline estimation algorithm in the ns-3 [18] branch maintained by Stefano Avallone [19].

The current ns-3 implementation has two assumptions that we removed. One is that all RUs should be of equal sizes, taken for simplicity of implementation. The other is that RUs are assigned in a round-robin manner to the STAs requesting UL traffic.

Multiple ways of splitting the channel allowed in 802.11ax, as described in Section 2, give rise to various possible RU configurations where each RU configuration has RUs of different sizes. We implement our scheduling by replacing the round-robin allocation with our allocation, where we pick the best RU configuration according to our criterion of lowest drop time, as discussed in Section 3. We then assign the RUs from the selected RU configuration to the STAs in the order of the earliest deadline.

In the following sections, we show the results for full buffer traffic and IoT traffic.

1) Results with full buffer traffic model. We take nine nodes and consider three maximum delays that define the deadline: with maximum delays of 20 ms for each STA, with small maximum delays (four nodes with 4 ms max delay, three 5 ms delay, and three 6 ms delay), and with large maximum delays (similarly with 40, 50, and 60 ms delay values). These values are set to the MSDU Life Time attribute of the Wi-Fi Mac Queue; the queue drops an MSDU whose MSDU Life Time value has expired. Each node sends 1472-byte UDP packets to the AP at a 20 Mbps rate. This makes a full buffer UL traffic from each STA, assuming each STA uses VHT MCS 8 that allows for a PHY rate of 86.7 Mbps in a 20 MHz channel at a guard interval (GI) of 0.4 μs.

Figure 10 shows the fraction of packets that expired out of the total of expired and received packets due to passed deadlines in STA queues. Each of the bars in the figures shows results when each STA transmits with increasing MCS values, decreasing MCS values, or with MCS 2, 5, or 8, respectively. For each MCS scheme, we compare the results for the simulation with our algorithm of minimization of drop time with deadline estimation (labeled DEADLINE ESTIMATION) and the existing algorithm of minimization of upload time according to [3] (denoted UPLOAD OPTIMIZATION that stands for minimum upload time). Results with our algorithm are, in general, better than UPLOAD OPTIMIZATION. We also show the error variation in estimating the deadline for each of the three cases.

2) Results with IoT traffic model. The IoT traffic model consists of the same traffic model [14] used above in our custom simulation, which also shows the per-STA deadlines, which are the same as the maximum delays after which a packet in the queue expires. Figure 11 shows the fraction of packets that expire when IoT traffic is scheduled with deadline estimation (DEADLINE ESTIMATION) and without (UPLOAD OPTIMIZATION), both for different MCS schemes described above.

Analysis and Explanation of ns-3 Results:

Our results with ns-3 simulation (Figures 10 and 11) conform to our hypothesis that the deadline estimation algorithm (DEADLINE ESTIMATION)

performs much better than the throughput optimization (UPLOAD OPTIMIZATION). For some cases, the deadline estimation shows worse results than the minimum upload time first algorithm, for example, at MCS 8 for the IoT traffic in Figure 11. The reason is that the buffers are full more often for no-deadline STAs (VoIP and file transfer) than for IoT STAs with small deadlines, and the deadline estimation, which heavily depends on reported queue sizes, could accurately calculate the deadlines when queues are substantially full. Our results demonstrate that the highest number of delay-bound packets can successfully meet their deadlines when the AP performs two tasks: 1) it gives priority to the HOL packets with the earliest deadlines while assigning OFDMA RUs, and 2) it selects RUs to assign after searching the list of RU configurations for an RU configuration that gives the least packet drops (for which we designed the metric drop time).

The best possible deadline-based OFDMA scheduling is achieved when AP knows the exact deadlines of the waiting HOL packets. We recommend modifying the 802.11 standard to include the HOL packet's time of entry into the queue or HOL delay in MPDU headers. The headers have reserved fields that can be utilized for this purpose. Nevertheless, our metric of drop time will still be relevant, which the AP will need for searching for the best RU configuration.

***A Remark about Starvation*:**

Reserving the RUs to provide scheduled UL OFDMA access to STAs that have deadlines may cause starvation of STAs without deadlines. We model the latter as STAs with arbitrarily large deadlines in all our simulations. DEADLINE-2 variation of our algorithm allows sending some no-deadline traffic, as does the GROUND-TRUTH. This happens because our algorithm and the GROUND-TRUTH allow sending most of the deadline-bound traffic on time, which makes the queues of STAs with deadlines occasionally empty, giving some chance to no-deadline STAs. Reservation was better handled when, from every RU configuration that we assigned, we allocated one or a few RUs for random access and allowed contention-based access to the STAs with best-effort (no-deadline) traffic.

5 CONCLUSION

Guaranteeing packet delivery within deadlines is essential to support time-sensitive networking in 802.11 WLANs. We convert the problem of ensuring the packet deadlines to the problem of minimizing packet drops due to deadlines, for which OFDMA emerges as an enabler technology. Since the deadlines are not readily available, our algorithm of deadline estimation enables the AP to estimate the deadlines from the queue sizes it receives in BSRs from STAs, a feature introduced in 802.11ax. We find that more frequent reporting of BSRs is required to assess the deadlines correctly for the flows with smaller delay tolerance values.

We calculate per-STA drop times, the time needed to transmit the packets that would have been dropped due to expired deadlines, after arranging STAs in earliest-deadline-first order using our estimated deadlines. We simulate both with our custom simulator and with ns-3 and find that OFDMA scheduling with minimization of drop times reduces the number of packets dropped due to past deadlines at STAs.

The AP can best schedule according to deadlines when it knows the ground truth – the deadline of every HOL uplink packet, which can be realized only when the 802.11 standard includes some way of reporting to the AP the amount of time the HOL packet has spent in the STA queue. In that case, as we have shown in our implementation of the scheduling with the knowledge of ground truth, the number of packets dropped due to expired deadlines will be lower than that obtained with our estimated deadlines. Nevertheless, our results with the scheduling that minimizes drop times using estimated deadlines are close to those obtained with the knowledge of ground-truth deadlines.

In future work, we will investigate the performance of our scheme when applied with aggregation and block ACK that prevents starvation of non-delay bound STAs, for which we proposed one solution as reserving the random access RUs so that the same Wi-Fi can be used for both the delay-bound traffic and the best effort traffic.

REFERENCES

[1] IEEE Draft 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 Enhancements for High-Efficiency WLAN. IEEE P802.11ax/D6.0, November 2019 (2019), 1–780.

[2] Jon Kleinberg and Eva Tardos. 2005. Algorithm Design. Addison-Wesley Longman Publishing Co., Inc., USA.

[3] D. Bankov, A. Didenko, E. Khorov, and A. Lyakhov. 2018. OFDMA Uplink Scheduling in IEEE 802.11ax Networks. In 2018 IEEE International Conference on Communications (ICC). 1–6.

[4] O. S. Aboul-Magd. 2008. IEEE Standard 802.11 Overview.

[5] R. Su and I. Hwang. 2016. Efficient resource allocation scheme with grey relational analysis for the uplink scheduling of 3GPP LTE networks. In 2016 IEEE International Conference on Industrial Technology (ICIT). 599–603.

[6] K. Arshad. 2015. LTE system level performance in the presence of CQI feedback uplink delay and mobility. In 2015 International Conference on Communications, Signal Processing, and their Applications (ICCSPA'15). 1–5.

[7] L. A. M. Ruiz de Temino, G. Berardinelli, S. Frattasi, and P. Mogensen. 2008. Channel-aware scheduling algorithms for SCFDMA in LTE uplink. In 2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications. 1–6.

[8] Harold J Kushner and Philip A Whiting. 2002. Asymptotic properties of proportional-fair sharing algorithms. Technical Report. BROWN UNIV PROVIDENCE RI DIV OF APPLIED MATHEMATICS.

[9] Aswin Kanagasabai and Amiya Nayak. 2015. Opportunistic dual metric scheduling algorithm for LTE uplink. In 2015 IEEE International Conference on Communication Workshop (ICCW). IEEE, 1446–1451.

[10] H. Ferng, C. Lee, J. Huang, and Y. Liang. 2019. Urgency Based Fair Scheduling for LTE to Improve Packet Loss and Fairness: Design and Evaluation. IEEE Transactions on Vehicular Technology 68, 3 (2019), 2825–2836.

[11] Y. Chen, X. Wang, and L. Cai. 2017. Head-of-Line Access Delay-Based Scheduling Algorithm for Flow-Level Dynamics. IEEE Transactions on Vehicular Technology 66, 6 (2017), 5387–5397.

[12] K. Wang and K. Psounis. 2018. Scheduling and Resource Allocation in 802.11ax. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 279–287.

[13] John D. C. Little. 1961. A Proof for the Queuing Formula: L = λW. Oper. Res. 9, 3 (June 1961), 383–387. https://doi.org/10.1287/opre.9.3.383

[14] Muhammad Inamullah, Bhaskaran Raman, and Nadeem Akhtar. 2020. Will My Packet Reach On Time? Deadline-Based Uplink OFDMA Scheduling in 802.11ax WLANs. In Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '20). Association for Computing Machinery, New York, NY, USA, 181–189. https://doi.org/10.1145/3416010.3423232.

[15] Nader Zein et al. 2020. IEEE 802 Nendica Report: Flexible Factory IoT: Use Cases and Communication Requirements for Wired and Wireless Bridged Networks. *IEEE 802 Nendica Report: Flexible Factory IoT: Use Cases and Communication Requirements for Wired and Wireless Bridged Networks* (2020), 1–48.

[16] S. Merlin R. Porat, M Fischer and et al. 2020. 11ax evaluation methodology. (2020), 1–48. https://mentor.ieee.org/802.11/dcn/14/11-14-

0571-12-00ax-evaluationmethodology.docx

[17] François Vergès. Mcs table (updated with 802.11ax data rates). https://www.semfionetworks.com /blog/mcs-table-updated-with-80211ax-data-rates, 2020. Accessed: 2020-11-23.

[18] NS-3 Consortium. ns-3 documentation. *https://www.nsnam.org/*, n.d. Accessed: 2020-11-23.

[19] Stefano Avallone. ns-3-11ax gitlab repository. *https://gitlab.com/stavallo/ ns-3-11ax*, n.d. Accessed: 2020-11-23.