# An Evaluation of Multi-User OFDMA Performance in WiFi 6 and its Optimization for Deadline Constrained Settings

By

Harshal Dev

Under the supervision of

Dr. Mukulika Maity

Dr. Arani Bhattacharya

Indraprastha Institute of Information Technology Delhi

December, 2021

# An Evaluation of Multi-User OFDMA Performance in WiFi 6 and its Optimization for Deadline Constrained Settings

By

Harshal Dev

Submitted in partial fulfillment of the requirements for the degree of

**Master of Technology**

To

Indraprastha Institute of Information Technology Delhi

December, 2021

# Certificate

This is to certify that the thesis titled *An Evaluation of Multi-User OFDMA Performance in WiFi 6 and its Optimization for Deadline Constrained Settings* being submitted by *Harshal Dev* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Master of Technology, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

December, 2021

Dr. Mukulika Maity

Dr. Arani Bhattacharya

Indraprastha Institute of Information Technology Delhi

New Delhi 110020

*Everything can be taken from a man but the last of the human freedoms—to choose one's attitude in any given set of circumstances, to choose one's own way.*

---

Viktor E. Frankl

*The world will ask you who you are, and if you don't know, the world will tell you.*

---

Carl Jung

*Meaning is only found in the journey uphill, and never in the fleeting sense of satisfaction awaiting at the next peak.*

---

Unknown

# Acknowledgements

I would like to sincerely thank my advisors, Dr. Mukulika Maity and Dr. Arani Bhattacharya, whose constant guidance, availability, and support have enabled me to navigate every obstacle encountered in the course of this project. I have learned and grown immensely through their constructive criticism and feedback on my work.

I am also thankful to Dr. Syamantak Das for his patient guidance on my work 'Driven by Deadlines'.

I would also like to acknowledge my brother, Prateek, for being a consistent source of support through exceptionally trying times in my life.

Lastly, I am grateful for my time at IIIT Delhi, which has proved to be a turning point in my life and has provided me with the opportunity to learn from exceptionally knowledgeable professors to better myself at my craft. Every lecture has been a privilege.

# Abstract

IEEE 802.11ax, popularly known as WiFi 6, introduces OFDMA (Orthogonal Frequency Division Multiple Access) that allows multiple users to transmit or receive frames concurrently via a more flexible utilization of the available bandwidth. Due to its ability to perform concurrent transmissions, prior studies suggest that OFDMA will provide reduced latency and increased throughput compared to OFDM. As our first work, we investigate this claim under various downlink traffic loads using the latest 802.11ax models supplied by the widely used open-source network simulator ns-3 (version 3.34). Our simulation results show that the actual benefits of OFDMA over OFDM can only be extracted under intermediate traffic loads. Motivated by this finding, we compare the performance of OFDMA and OFDM by simulating various application settings involving intermediate traffic rates. We find that OFDMA provides considerable improvements over OFDM under such traffic conditions for different application-specific parameters of interest, e.g., OFDMA delivers a more consistent bitrate for live video streaming applications and reduced jitter for video conferencing applications. Furthermore, for applications involving small payloads such as web-based applications and factory IoT-based motion control applications, OFDMA results in $7 - 10\times$ lower average latency when compared to OFDM.

We observe from our experiments that OFDMA experiences a substantial increase in latency as the payload size becomes small (90-30 B). In an IoT-based factory setting, applications frequently exchange small payloads with stringent deadline constraints; missing these deadlines for certain critical applications such as human-safety monitoring and equipment control have a huge impact or penalty. Our experiments make it evident that such constraints cannot be met by simple schedulers such as round-robin. Thus there is a need for intelligent scheduling techniques in such deadline-based environments. As our second work, we propose a deadline-aware scheduler for factory environments that maximizes the number of packet deadlines met for critical applications and show that our scheduler significantly reduces the overall penalty incurred due to missed deadlines compared to other deadline-based schedulers.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Over the last decade, the demand for continuous connectivity has sky-rocketed. As a consequence, there has been a significant increase in the number and variety of wireless devices. WiFi has emerged as the leading wireless technology to cater to this demand, and it has provided connectivity in crowded public spaces such as stadiums, airports, classrooms, and even inside large residential complexes. Furthermore, the integration of wireless networks into traditional wired factory networks is being driven by the emerging concept of a Flexible Factory [1]. The Flexible Factory report [2] outlines many wireless applications that are in current use in factories or expected to be of use in the near future. Thus, WiFi is contending for space in industrial settings as well.

WiFi uses standard Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to mitigate collisions, where each node has to contend for channel access before each transmission. Only on winning the contention can a device transmit data. Such a contention based channel access scheme incurs huge overheads of waiting for each transmission. Moreover, on winning a contention, only a single device can (successfully) transmit, which leads to reduced network throughput and high latency. As a consequence of this, empirical evidence suggests that WiFi performance suffers in dense settings [3]. In fact, one of the primary obstacles to the integration of wireless networks in industrial environments has been the non-determinism associated with this randomised contention based channel access scheme, resulting in intense competition by multiple wireless technologies in the industrial space [4]. Nevertheless, continued efforts by the research community have kept WiFi in the fight [5, 6].

The recent WiFi standard IEEE 802.11ax, popularly known as WiFi 6, further advances these efforts with an aim to improve WiFi performance in dense scenarios by replacing Orthogonal Frequency-Division Multiplexing (OFDM) based single-user transmissions with Orthogonal Frequency-Division Multiple Access (OFDMA) based multi-user transmissions. It also supports a shift from randomised contention based channel access to centralized access via the introduction of Multi-User EDCA parameters. OFDMA allows parallel transmissions by breaking the channel into orthogonal sets of sub-carriers (or tones) called resource units (RU) and assigning them to multiple users for each transmission [7, 8], as shown in Figure 1.1. By allowing multiple users to access the channel simultaneously, OFDMA can improve the throughput per area significantly by reducing the channel contention in the wireless system (see grey bars in Figure 1.1). Moreover, the introduction of a Multi-User (MU) EDCA procedure to suppress station contention for channel access via an advertised MU EDCA parameter set can further reduce collisions by completely relegating channel access decisions to the access point.

On account of these features, the 802.11ax standard claims to improve the average performance by up to $4\times$ [7]. There are however, a few challenges in realizing this gain:

- As OFDMA breaks the available bandwidth into smaller RU's, some tones are converted into DC, Guard, and Null tones, reducing the overall channel capacity.
- There are additional overheads associated with OFDMA transmissions. For downlink transmission, a common preamble is sent before the OFDMA transmission specifying information such as the duration of the frame, tones assigned to the users, etc. Similarly, for uplink, a trigger frame is sent that carries this information. Further, after the OFDMA transmission, MU-Block Acknowledgements are solicited. Avallone et al. [9] have shown that compared to single user frames, acknowledgment of multi-user frames introduces significant overheads. Moreover, these overheads increase with an increase in the number of users.
- Overheads are also associated with obtaining buffer status reports (BSR) and channel state

information.

- Selecting the optimal sub-carrier allocation for users is also a challenging task, because of multiple possible ways in which the channel can be split into RUs.

Thus, while OFDMA offers potential benefits, there is a need to understand when these benefits can be extracted or how the associated overheads can be mitigated.

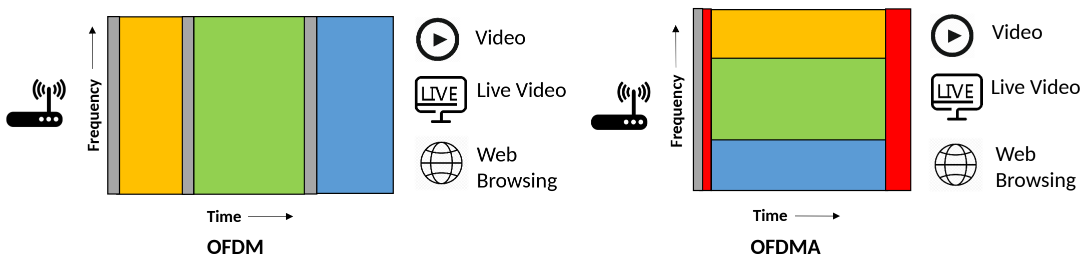


Figure 1.1: Sequential OFDM vs. Parallel OFDMA Transmissions. Grey bars show contention overhead while red bars show additional overheads due to the longer Preamble, MU-BAR Aggregated Trigger and MU-Block Acks.

Most previous work on IEEE 802.11ax [10, 11, 12] has primarily focused on scheduling and resource allocation. Each of these works implicitly assumes that OFDMA based multi-user transmissions would always provide better performance compared to OFDM based single-user transmissions. However, some recent works [13, 9] have showcased that these assumptions are not always true. Specifically, [13] showcased by running iperf utility that OFDMA provides performance benefits only in dense WiFi scenarios with over 20 clients. [9] have shown that under saturated channel conditions, OFDMA provides reduced network throughput compared to OFDM. Hence, universal assumptions about OFDMA performance should not be made. However, a deeper investigation based on realistic application traffic is needed to understand the realistic benefits.

Thus, as our first work (Chapter 3), we utilize the latest 802.11ax models supplied by the open-source network simulator ns-3 (version 3.34) to investigate the validity of the assumptions about the performance of OFDMA. To this end, we compare OFDM and OFDMA performance under various traffic loads in an attempt to find the scenarios where multi-user transmissions outperform single-user transmissions. Our simulation based experiments in ns-3 (Section 3.1) show that the true benefits of OFDMA over OFDM in terms of aggregate throughput and overall average packet latency can only be extracted under intermediate traffic loads. Overheads associated with OFDMA transmissions (shown in Figure 1.1) under high traffic loads decrease the time duration available for transmitting aggregated frames, which leads to a significant reduction in aggregate throughput compared to OFDM. These results corroborate the results presented by [9] for saturated channel conditions. Under low traffic loads, no significant performance gains are observed by using OFDMA. That is because, under such conditions, single-user transmissions easily catch up to multi-user transmissions since in the latter case most time is spent waiting for the arrival of packets in the queue.

Motivated by these findings, we take a closer look at the performance of OFDMA compared to OFDM for various application settings involving intermediate traffic loads in a dense scenario with many stations (Section 3.2). We find that OFDMA provides considerable improvements over OFDM under such traffic conditions for different application-specific parameters of interest, e.g., OFDMA delivers a more consistent bitrate for live video streaming applications, and reduced jitter for video conferencing applications. Furthermore, for applications involving small payloads such as web-based applications and factory IoT-based motion control applications, OFDMA results in $1.6\times$ less packet queuing and approximately $7$ - $10\times$ lower average latency when compared to OFDM.

Although the experiments in our first work establish the clear supremacy of OFDMA over OFDM for intermediate traffic load scenarios, we observe that as the payload size becomes small ($90$-$30$ B), OFDMA experiences a substantial increase in latency due to an increased packet arrival rate (Section 3.2.3). The Flexible Factory report [2] illustrates several wireless applications in current use in various factory settings. Many outlined applications generate time-bound traffic composed of small payloads ($30$-$500$ B). Some of these applications handle critical operations such as human safety monitoring or equipment control. Our simulation results make it evident that requirements for such time-constrained traffic cannot be met by simple schedulers, such as round-robin (implemented by ns-3) for small payload sizes.

The concept of a 'Flexible Factory' is primarily driven by the demand for shorter product development cycles which require greater flexibility in the layout of machines and the order of manufacturing processes [1]. Apart from this, there is a natural desire to reduce installation costs and collect useful information from IoT sensors to better allocate resources and monitor the status of humans and machines. In order to meet these requirements, traditional wired factory networks must evolve to incorporate wireless networks.

IEEE 802.11ax also introduces a shift from a distributed contention-based channel access scheme to a centralized scheme where the access point can be in charge of scheduling uplink traffic along with downlink traffic. This paradigm shift is facilitated by the introduction of Multi-User (MU) EDCA procedure. In this procedure, the access point advertises a set of EDCA parameters composed of channel contention parameters and a countdown timer. On performing a successful uplink OFDMA transmission, the stations switch their contention parameters to MU EDCA parameters for a duration specified by the countdown timer. If the timer is long enough, station contention for channel access can be completely suppressed and thus channel access can be completely centralized by relegating all access responsibilities to the access point. This feature can be utilized the counter non-determinism associated with the randomised channel access procedure (DCF/EDCA) employed by stations in a WiFi network which is a primary obstacle to the deployment of WiFi in factory settings.

As our second work, (Chapter 4), we utilize the paradigm shift towards centralized channel access supported by 802.11ax to design a scheduler for deadline-based factory settings which attempts to maximize the number of packet deadlines met for the most critical applications (Section 4.2) by enforcing the stations to follow packet transmission schedules provided by the access point via a trigger frame. We show that our scheduler significantly reduces the overall penalty incurred due to missed packet deadlines compared to other deadline-based schedulers (Section 4.5).

# Chapter 2

# Related Work

We first discuss prior works related to the evaluation of OFDMA performance in WiFi 6 and then focus our attention on works investigating the use of WiFi in industrial settings.

## 2.1 Evaluation of OFDMA Performance

Existing works on the performance of OFDMA fall into two categories:

- Obtaining better sub-carrier allocation to increase efficiency.
- Performance evaluation via modeling-based and empirical experiments.

The first category of works attempts to allocate sub-carriers to users to optimize a defined objective function. The second category attempts to gain insight into the performance of various 802.11ax features via different experimentation techniques without a focus on sub-carrier allocation.

### 2.1.1 Optimal Subcarrier Allocation:

Multiple works have looked at Scheduling and Resource Allocation problem of IEEE 802.11ax. Hence, various scheduling and resource allocation strategies have been proposed.

Wang et. al [10] considers the objective of maximizing a sum utility under three modes of operation: OFDMA, MU-MIMO and OFDMA+MU-MIMO jointly. The authors propose a divide & conquer algorithm and prove its optimality for the relaxed assumption of one or more users being assigned the same RU. Next, they propose two practical algorithms that do not violate the aforementioned assumption, called Greedy and Recursive. The performance of the optimal algorithm serves as an upperbound for these two algorithms. The key idea in these algorithms is to break a RU into exactly two smaller RUs, creating a tree structure. The complexity of the Recursive algorithm is $O(4^L N log N)$ for simple OFDMA mode where $L$ cannot exceed 7 based on the available RU configurations in 802.11ax. The complexity of the Greedy algorithm $O(N^2)$. Moreover, the Greedy algorithm exhibits much more poor performance than the Recursive algorithm.

In [14], the authors propose a scheduler that maximizes throughput while preventing starvation for stations with low link rates. In order to achieve this they add an ageing factor in the formulation of their optimization problem (aimed at maximizing throughput) and update it after each transmission based on the transmission rate in the last round and current assigned MCS value. The authors evaluate the performance of their proposed scheduler in ns-3 and utilize an external library to directly solve their formulated optimization problem. They compare the performance of their scheduler against the performance of legacy random access in WiFi (DCF/EDCA) and report higher throughput and lower per station delay achieved in compared to scenarios where the stations only use DCF/EDCA for channel access.

DeepMux [15] proposes a Deep Neural Network (DNN) based procedure for obtaining Channel State Information (CSI) and for obtaining a near-optimal solution to the resource allocation problem in Joint MU-MIMO and OFDMA. The CSI information is inferred by receiving quantized CSI information on a small subset of tones from the stations and then passing them to a well-trained DNN. The scheduling and resource allocation problem is modeled as a Mixed Integer Non-Linear Programming (MINLP) problem. To solve it via DNNs, the problem is divided into two sub-problems, finding optimal RU assignment and allocating power. The DNN is used to find a sub-optimal RU assignment which converts the MINLP problem into an LP problem, which is then solved. To evaluate the performance of DeepMux the authors utilize a test-bed with 1 AP and 4 stations and implement DeepMux ontop of the 802.11ax protocol and train DNNs at the AP by utilizing the PyTorch library. The authors show that DeepMux provides higher Net Throughput compared to other 802.11 defined channel sounding protocols.

The work in [16] proposes an uplink scheduler for minimizing the upload time (MUTAX). The authors first formulate their scheduling problem as an optimization problem that maximizes the throughput, which indirectly minimizes the upload time. The authors then propose an algorithm for the optimization problem that utilizes the Hungarian algorithm to find the optimal user-to-RU allocation for a given RU configuration and MCS value. To find the global optimal RU allocation, the complete algorithm performs an exhaustive search over all possible RU configurations that can be input to the Hungarian Algorithm. To evaluate the performance of their algorithm, the authors implement their scheduler in ns-3 and also implement adaptions of known OFDMA schedulers in LTE such as Proportionally Fair (PF), Shortest Remaining Time First (SRTF) and Max Rate (MR). The authors report that when the stations are close to the AP and the channel conditions are perfect, MUTAX and SRTF yield the same performance and perform $30\%$ better than PF and MR. When the stations are far apart, both MUTAX and PF peform better than SRTP and MR on account of their efficient channel splitting mechanisms.

The authors then extend this work in [12] and propose a generic sum maximizing utility function to show how popular schedulers proposed for LTE like Max Rate, Proportional Fair, Shortest Remaining Processing Time (SRPT) can be adapted for use in IEEE 802.11ax. The authors continue to utilize the Hungarian algorithm to find the optimal RU allocation by performing an exhaustive search over all available RU configurations and MCS values. Performance evaluation is once again conducted in ns-3 and the authors show that when stations are arranged in a large circle around the AP, due to variety in MCS values, stations are able to gain an advantage by dividing the channel via OFDMA. And so, non-11ax schedulers are $2\times$ less efficient compared to the channel splitting proposed schedulers.

Inam et al. [17] considered the direct problem of designing a scheduler for minimizing the number of packet drops in a time-sensitive environment. Their work first provides an algorithm for estimating the deadlines of packets using Little's Theorem and the Buffer Status Reports collected by the Access Point. Once the deadlines have been estimated, they suggest a simple heuristic that sorts the stations based on their deadlines (earliest first) and then perform an exhaustive search over all possible RU configurations to find the configuration that results in the least number of packets dropped. The exhaustive search uses a recursive expression to estimate the packet drops for a given RU configuration by taking into account the estimated deadlines and the waiting times for the stations. The authors first evaluate the performance of their deadline estimation algorithm against an ORACLE that knows the actual deadlines. The deadline values estimated by their algorithm periodically converge to the actual deadline with a fixed maximum error. Then they compare the performance of their scheduler against the MINUL scheduler which minimizes the upload time by maximizing the throughput. The authors report a $6\times$ reduction in the observed packet drops compared to MINUL.

Although the above-discussed works have provided solutions for the design of various OFDMA schedulers in different WiFi settings, none of them, apart from the work by Inam et al. [17], has attempted to design a scheduler with deadlines as the primary constraint.

### 2.1.2 Performance Evaluation and Optimization:

The performance of OFDMA has been evaluated under different conditions for a long time for different protocols such as cellular networks [18, 19] and WiMAX [20]. However, these evaluations have not been conducted for channel models or environments typically seen in WiFi networks.

802.11ax introduces 1024-QAM and DL OFDMA which is expected to provide a 25% increase in throughput. The work in [13] investigates this claim using a test-bed with two different AP vendors. They report low DL OFDMA usage in their traffic profiling data for a one vendor because of a complex scheduler that takes into account the packet sizes, buffer size and the number of stations exchanging packets. Another vendor always forces OFDMA transmissions. Regardless, both the vendors utilize the channel resources poorly and suffer throughput losses as a result. The authors also report that the 1024-QAM feature is only usable when the range of operation is less than 6 m. Finally the authors note that the poor implementation of the schedulers by both the vendors result in higher latency. For the given test-bed, the authors conclude that the benefit of OFDMA is limited unless the number of clients is large. The work thus establishes the important role of the scheduler in realizing the gains offered by OFDMA in 802.11ax.

Naribole et al. [21] investigated the impact of Multi-User (MU) EDCA parameters on network performance in OFDMA for saturated traffic and video traffic. The authors implement their 802.11ax models in ns-3 for conducting the evaluations. For the saturated traffic conditions and a high MU EDCA timer value, they report a $4\times$ increase in throughput. For video traffic, a non-zero MU EDCA timer value results in a significant drop in latency, more so for HD quality video traffic. Finally, the authors note that an MU EDCA timer value of $200$ ms can support a network of $2000$ nodes for a high MCS value. Thus, the authors conclude that by utilizing varying MU EDCA parameters to alter the channel access priority of the stations, fairness to the Access Point can be guaranteed in the network. This work has shown that non-determinism in a WiFi network can be limited via the newly introduced MU EDCA procedure. In our work, we utilize this feature when we design a deadline-based scheduler in Chapter 4.

Extensive simulation-based evaluation of OFDMA for WiFi became much easier with the release of ns-3 version 3.34, which offers WiFi models validated against an analytical model of OFDMA transmissions across many stations [22]. The work [9] leveraged these models to show that although OFDMA provides lower latency compared to OFDM under unsaturated channel conditions, its use results in lower throughput under saturated channel conditions. The authors also evaluated the effect of MU EDCA parameters with a very long time value (2 ms) on performance in a scenario with both uplink and downlink traffic. They note that overheads associated with UL OFDMA transmission increase with an increase in the number of users, which in turn increase the overheads for DL OFDMA. Thus, although MU EDCA parameters allow elimination of collisions, when the number of users is high, throughput degrades compared to No OFDMA case on account of these overheads. Finally, they show via simulations that when the distance between the stations and the AP is large, increased power spectral density in small RUs leads to increased throughput in UL OFDMA transmission. This work has partially motivated us to undertake a more thorough investigation of OFDMA performance in 802.11ax in Chapter 3.

## 2.2 WiFi in Industrial Settings

WiFi deployment in the industrial space suffers from two major obstacles. Non-determinism associated with the randomised contention based channel access scheme employed by WiFi and unreliability of wireless links. Multiple works have attempted to overcome or navigate these obstacles by employing different strategies. The works discussed in this section provide solutions for WiFi deployment in factory environments orthogonal to the solution proposed in our work. However, they are worth mentioning because they have allowed WiFi to remain a relevant technology for industrial wireless networks.

Seno et al. [6] attempted to enhance determinism in WiFi communications to enable soft-real time applications in industries. The authors proposed a generic framework composed of a coordinator node with supreme rights to initiate transmissions via request packets. The authors applied this proposed framework to WiFi and evaluated the performance via a test-bed composed of a personal computer acting as the network coordinator. MAC DATA frames were used to encode request and data packets. The authors report a high delivery success percentages both for the case of an internal error generator and an external error generator, and note that the primary cause for missed deadlines were transmission errors.

The work in [23] makes a case for the use of IEEE 802.11n in industrial settings. The authors first perform a thorough study of the features introduced by the standard and then evaluate the performance of the standard by deploying a test-bed which simulates an industry environments in a laboratory by injecting noise through an RF generator. The authors report a substantial decrease in the Packet Error Rate (PER) by the adoption of space-time block coding (STBC) technique in Multiple-Input and Multiple-Output (MIMO) transmissions.

The use of co-operative diversity in a factory environment was investigated in [24]. Co-operative diversity allows re-transmissions for failed packets by utilizing a relay node. The relay node overhears packets transmitted on the network and responds to failed transmissions. Selective co-operating relaying involves use of policies for timing the update of relay nodes. The authors evaluate the performance of three relaying protocols-Periodic, Adaptive and Reactive, by implementing them on 7 nodes deployed in a packaging plant. They authors report that all the three protocols outperform the conventional time diversity protocols in packet delivery ratio. Moreover, the reactive relaying protocol provides the best performance.

The work in [5] proposes Wi-Red, a parallel redundancy protocol (PRP) which aims to improve timeliness and dependability in links by transmitting duplicate packets on redundant links. It also aims to avoid unnecessary duplication on the redundant links which is a major drawback of other known PRPs. The authors compare the performance of Wi-Red by employing different duplication avoidance algorithms (PDA/RDA) against other well known PRP over WiFi (PoW) protocols. To evaluate the performance of Wi-Red, the authors utilize a test-bed comprising of a single link between a source and a destination with a number of intereferring nodes and two jammer nodes. The authors report a significant reduction in the percentage of packets drops by using Wi-Red when compared to legacy DCF transmission mechanisms and other PoW protocols.

Chapter 3

# When is Multiple Access Beneficial? Analysing Multi-User Performance in 802.11ax

In this work, we investigate the validity of various assumptions about OFDMA performance in WiFi 6. We begin by comparing the performance of multi-user OFDMA transmissions and single-user OFDM transmissions under various traffic loads (in Section 3.1). Next, we compare OFDMA and OFDM in various practical settings (in Section 3.2) involving various applications. These applications are sensitive to various performance metrics, such as bit-rate consistency for live video streaming applications, jitter for video conferencing applications and latency for web-based and factory IoT-based applications. We utilize the latest 802.11ax models supplied by the open-source network simulator ns-3 (version 3.4) to conduct our evaluations.

## 3.1 OFDMA VS. OFDM under various Traffic Loads

### 3.1.1 Simulation Setup

The latest 802.11ax models in Network Simulator 3 (version 3.34) [25] support both downlink and uplink OFDMA transmissions, and provide a Round Robin scheduler which splits the bandwidth into a user specified RU configuration. For a 40 MHz bandwidth, available RU configurations are as follows: $2 \times 242$-tones, $4 \times 106 + 2 \times 26$-tones, $4 \times 106$-tones, $8 \times 52 + 2 \times 26$-tones, $8 \times 52$-tones and $18 \times 26$-tones (See Figure 3.1 below).

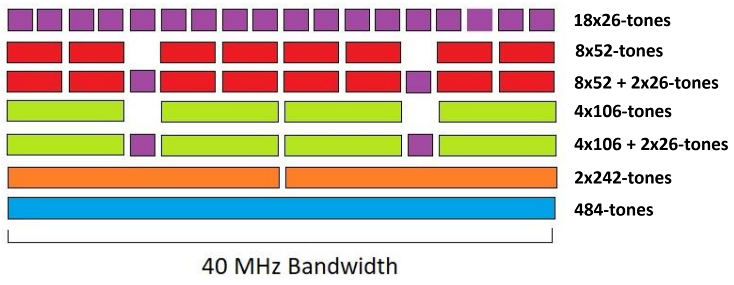


Figure 3.1: Available RU configurations for the Round Robin scheduler

The parameters for the simulation setup are summarized in Table 3.1. The stations are uniformly distributed around the AP in a radius of $5$ meters. The MAC queue sized is fixed at $6,000$ packets to allow sufficient frames for aggregation. The PHY rate at the given MCS and Guard Interval is $243.75$ Mbps. The stations start their applications at random instances. The acknowledgment sequence for OFDMA consists of a MU-BAR Trigger Frame aggregated to a downlink transmission followed by MU Block Acks. In case of OFDM, the acknowledgments are the usual Implicit Block Acks. We measure the aggregate downlink throughput achieved and the average packet latency experienced by the stations (both at the MAC Layer) as we vary the downlink traffic load from $245$ Mbps (High) to $5$ Mbps (Low).

Since an OFDMA transmission can take at most TXOP limit ($5.44$ ms) time duration, we divide the simulation duration into abstract time windows of size equal to the TXOP limit and tabulate some metrics of interest within such time windows to better interpret the results of our analysis (Tables 3.2 and 3.3). Since the simulation duration is fixed, the number of such time windows is also fixed.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Stations | 40 | Scheduler | Round Robin |
| Radius | 5 m | Bandwidth | 40 MHz |
| Transport Protocol | UDP | Guard Interval | 3200 ns |
| Payload Size | 1000 B | MCS | 11 |
| MSDU Aggregation | Disabled | MPDU Aggregation | Enabled |
| MAC Queue Size | 6, 000 packets | Simulation Duration | 10 sec |

Table 3.1: Simulation Parameters

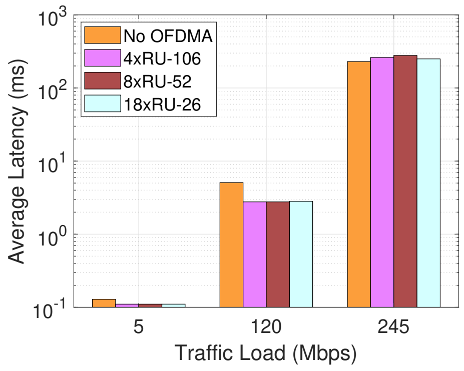


Figure 3.2: Effect of Traffic Load on Average Latency

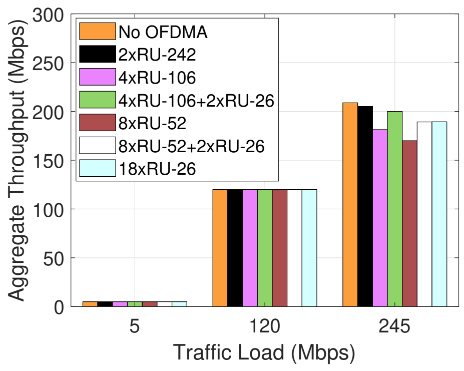


Figure 3.3: Effect of Traffic Load on Aggregate Throughput

### 3.1.2 High Downlink Traffic Load

Under a very high downlink traffic load (245 Mbps), we note from Figure 3.3 that the aggregate throughput achieved by OFDMA is less than that achieved by OFDM[1]. Since there is only downlink UDP traffic, there are no collisions in the system. Thus, throughput is only limited by the overheads associated with the contention-based channel access scheme (employed by the Access Point), the Preamble, MAC header and the Acknowledgment sequence [9].

[1]We use the terms OFDM and No OFDMA interchangeably.

| Configuration | Transmissions | Stations Served | Frames Transmitted |
|---|---|---|---|
| No OFDMA | 3 | 3 | 145 |
| 4×RU-106 | 1 | 4 | 124 |
| 8×RU-52 | 1 | 8 | 120 |
| 18×RU-26 | 1 | 18 | 126 |

Table 3.2: Simulation metrics over a TXOP limit time window
(High Downlink Traffic Load)

Avallone et al. [9] point out that whenever the bandwidth is equally split into smaller sized RUs, there is an increase in the overheads associated with the transmission of the Preamble, the MU-BAR Aggregated Trigger Frame, and the MU-Block Acks. These overheads decrease the time duration available for transmitting aggregated frames (see Figure 1.1), limiting the size of the generated A-MPDUs.

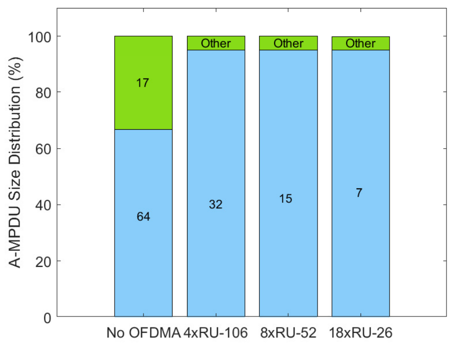


Figure 3.4: Aggregation Statistics for High Traffic Load

Since the channel is saturated (Aggregate downlink traffic load $\approx$ PHY rate $= 243.75$ Mbps), the queue quickly fills up, implying that there are always enough packets in the queue for frame aggregation. Thus, aggregation stops either because further aggregation would violate the TXOP limit with the given RU size, or because the Block Ack sequence number space $(64)$ is exhausted [26].

Figure 3.4 shows that for any OFDMA configuration, all generated A-MPDUs have size less than the maximum possible A-MPDU size $(64)$. Hence, for OFDMA, the number of MPDUs transmitted per station is bottle-necked by the TXOP limit. This implies that each OFDMA downlink transmission spans an entire TXOP limit (Table 3.2), and consequently, for each RU configuration, an equal number of transmissions are performed. For OFDM, however, Figure 3.4 shows that $66.67\%$ of the generated A-MPDUs have size equal to the maximum possible A-MPDU size. Hence, the number of MPDUs transmitted per station is bottle-necked by the sequence number space ($64$). This implies that an OFDM downlink transmission doesn't span the entire TXOP limit. In fact as mentioned in Table 3.2, compared to only $1$ OFDMA transmission, $3$ OFDM transmissions are performed within a TXOP limit time window, transmitting approximately $145$ MPDUs (verifiable from Figure 3.4, $64+64+17 = 145$). Thus, OFDM reports a higher throughput than OFDMA. This result shows that OFDMA should not be the go-to option for dense scenarios with high downlink traffic load.

### 3.1.3 Low Downlink Traffic Load

Under low traffic loads ($5$ Mbps), Figure 3.3 and 3.2 show there is no significant difference between OFDMA and OFDM transmissions in terms of aggregate throughput or overall average latency (OFDM has a latency of $0.128$ ms compared to $0.110$ ms for OFDMA). This is because, at such low data rates, OFDMA quickly performs concurrent transmissions to all the stations whose packets have arrived in the queue, and then it waits for the next set of packets to arrive. During the same time period, OFDM is able to perform sequential single user transmissions to the same stations, and as a result, is able to catch up to OFDMA in terms of performance. Hence, at low traffic loads, OFDMA offers no significant benefit over OFDM even in dense scenarios.

### 3.1.4 Intermediate Downlink Traffic Load

As can be seen from Figure 3.2 and Figure 3.3, under an intermediate traffic load scenarios ($120$ Mbps), both OFDMA and OFDM provide similar performance in terms of throughput, however, in terms of latency OFDMA always outperforms OFDM. The latency values we obtain are similar to the ones reported by [9] for an aggregate downlink data rate of 100 Mbps.

Since the channel is not saturated (Aggregate data rate $<$ PHY rate), queuing is limited. The limited queuing causes frame aggregation to now be bottle-necked by the number of MPDUs in the queue for both OFDM and OFDMA. Figure 3.5 shows that OFDM experiences more queuing compared to OFDMA. However, as indicated in Table 3.3, OFDM is able to transmit as many

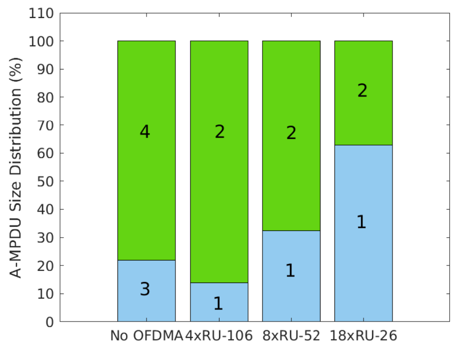


Figure 3.5: Aggregation Statistics for Intermediate Traffic Load

| Configuration | Transmissions | Stations Served | Frames Transmitted |
|---|---|---|---|
| No OFDMA | 22 | 22 | 82 |
| 4×RU-106 | 13 | 52 | 82 |
| 8×RU-52 | 10 | 80 | 82 |
| 18×RU-26 | 5 | 90 | 82 |

Table 3.3: Simulation metrics over a TXOP limit time window
(Intermediate Downlink Traffic Load)

frames as OFDMA within a time window by performing more transmissions, and therefore, achieves the same throughput.

The lower average latency observed with OFDMA is related to the number of clients served over a TXOP limit time window. As mentioned in Table 3.3, in each TXOP limit time window, OFDM serves $22 \times 1 = 22$ stations while OFDMA (18×RU-26) serves $5 \times 18 = 90$ stations. Thus, OFDMA serves more clients in each TXOP limit window than OFDM, which leads to decreased waiting times for the packets. Hence, the true benefits of multiple access in terms of aggregate throughput and overall average latency only become visible under intermediate data rates.

## 3.2 Performance Analysis of Applications under Intermediate Loads

Motivated by our findings in the previous section, we now compare the performance of OFDMA and OFDM for various applications under intermediate traffic loads.

### 3.2.1 Live Video Streaming Applications

The demand for live video content has shown consistent growth over the years [27]. Usually, applications delivering videos to the user at a latency of less than $10$ s are known as live video streaming applications. Since it is important for live video streaming applications to deliver frames that correspond to the most recent event, if a consistent bitrate is not being received at the user end, then such applications begin requesting images of lower quality. Therefore, bitrate consistency is a metric of interest for live video streaming applications, and in this section, we measure it for a typical streaming setting.

#### 3.2.1.1 Simulation Setup:

We simulate two scenarios involving 40 clients viewing a live video. In one scenario, the video is being viewed at 720p, 60 fps (High Quality), whereas in the other, it is being viewed at 1080p, 60 fps (Very High Quality). Bitrates in the range of $2,250 - 6,000$ Kbps and $4,500 - 9,000$ Kbps are required for viewing a live video at $720$p, $60$ fps and $1080$p, $60$ fps respectively [28]. Hence, to simulate the first scenario we set the per station downlink data rate to $3$ Mbps and to simulate the second scenario we set it to $5$ Mbps. The rest of the simulation parameters are the same as in Table 3.1. We then measure the aggregate throughput and average packet latency after $100$ ms intervals.

#### 3.2.1.2 High Quality Live Video:

In the case of OFDM, the box plots in Figures 3.6 and 3.7 below report an inter-quartile range (IQR) of $8.93$ Mbps and $0.33$ ms for throughput and latency respectively, compared to $0.01$ Mbps and $0.00017$ ms for the $18\times$RU-26 OFDMA case. Hence, it is evident from the box plots that OFDMA provides much more consistent performance compared to OFDM in terms of aggregate throughput and average latency. The reason for this superior performance is the ability of OFDMA to serve more clients compared to OFDM over any given time window, as explained in Section 3.1.4.

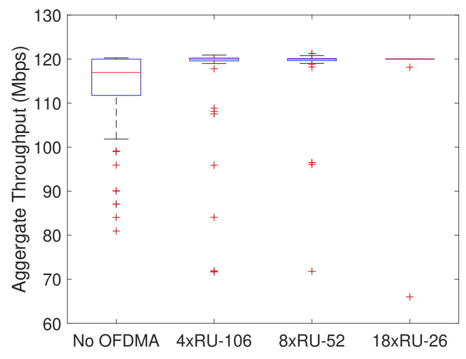


Figure 3.6: Distribution of Measured Aggregate Throughput for High Quality Live Video

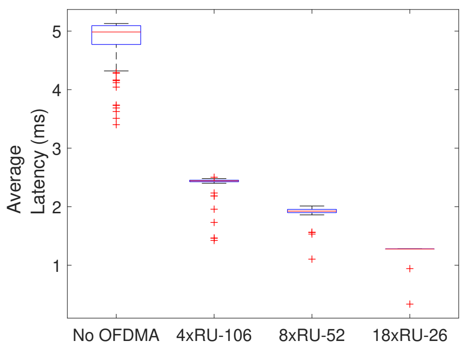


Figure 3.7: Distribution of Measured Average Latency for High Quality Live Video

#### 3.2.1.3 Very High Quality Live Video:

We notice from the box plot for this simulation (Figure 3.8) that compared to the previous scenario (Figure 3.6), the boxes are larger for OFDM and two OFDMA configurations ($4\times$RU-106 and $8\times$RU-52). However, for the $18\times$RU-26 configuration, the box is still tiny. The IQR for OFDM in the plot is $32.61$ Mbps, compared to $34.75$ Mbps, $37.27$ Mbps and $0.56$ Mbps for the $8\times$RU-52, $4\times$RU-106 and $18\times$RU-26 OFDMA configurations respectively.

As the per station (downlink) data rate increases from $3$ to $5$ Mbps, we move from the intermediate downlink traffic load scenario towards the edge of the high downlink traffic load scenario. Hence, the number of clients served by OFDMA in each TXOP limit time window reduce (Table 3.2 and 3.3), thereby leading to the observed degradation in performance for the $4\times$RU-106 and $8\times$RU-52 configurations. Furthermore, for OFDM, the queue quickly fills up, leading to packet drops, and so we do not report the latency in this case.

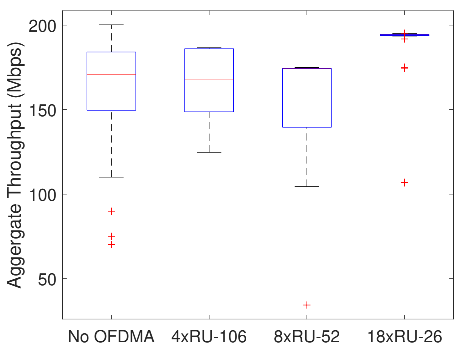


Figure 3.8: Distribution of Measured Aggregate Throughput for Very High Quality Live Video

### 3.2.2 Video Conferencing Applications

During the Covid-19 pandemic, video conferencing has become the default mode of communication [29]. These applications are more sensitive to latency and network jitter than ordinary live video streaming since any perceptible delay is noticed by the users. Here, we measure these metrics for a typical scenario involving video conferencing traffic (three-person group video conference).

#### 3.2.2.1 Simulation Setup:

Once again, we use the same simulation setup as mentioned in Table 3.1. However, in order to simulate each client engaged in a three-person group video conference, we set the per station downlink data rate to 2 Mbps and uplink data rate to 512 Kbps [30].

Since there is now uplink traffic being generated at the stations, it is essential to schedule uplink OFDMA transmissions. After every downlink OFDMA transmission, a BSRP trigger frame is sent by the AP to collect Buffer Status Reports via QoS null frames. Then, a trigger frame is sent to solicit an uplink OFDMA transmission from the same set of stations that were served in the last downlink OFDMA transmission. To support OFDMA transmissions, we implement and utilize MU-EDCA parameters. Naribole et al. [21] point out that for 32 stations, an MU EDCA timer value of 25 ms is sufficient, however since our scheduler collects Buffer Status Reports before *each* uplink OFDMA transmission, we set it to 50 ms.

#### 3.2.2.2 Three-person Group Video Conference:

For reporting jitter, we measure the difference between the $90^{th}$ and $75^{th}$ percentile values. The downlink and uplink jitter values for all configurations have been tabulated in Table 4.1. The box plots in Figures 3.9 and 3.10 show the packet latency distribution for each configuration.

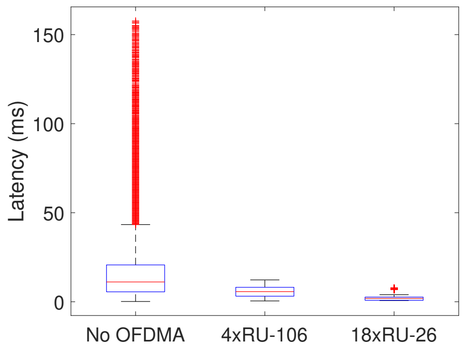


Figure 3.9: Packet Latency Distribution for Downlink

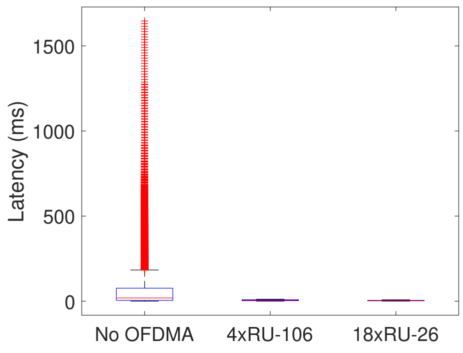


Figure 3.10: Packet Latency Distribution for Uplink

The box plots indicate that for both downlink and uplink, OFDM reports much higher latency values compared to OFDMA. Also, as can be noted from Table 3.4, OFDM shows significantly higher network jitter. The reason for the high latency and jitter values reported for the OFDM case are collisions in the system due to the presence of both uplink and downlink traffic. For the OFDMA case, there are no collisions in the system since station contention is suppressed via the advertised MU EDCA parameters. Moreover, as expected, the $18\times$RU-26 configuration provides the most superior performance in terms of latency and jitter.

| Configuration | Downlink Jitter (ms) | Uplink Jitter (ms) |
|---|---|---|
| No OFDMA | 14.54 | 145.62 |
| 4×RU-106 | 1.52 | 1.87 |
| 18×RU-26 | 0.44 | 0.65 |

Table 3.4: Network Jitter for Three-person Group Video Conference

### 3.2.3 Web-based and Motion Control Applications

Internet traffic is generally composed of packets sizes in the range of 40 - 1000 B [31]. Moreover, in IoT-based factory settings, motion control applications responsible for controlling the movement/rotation of machine components frequently exchange control information of the order of tens of bytes (≈ 30 bytes) [1, 32]. In this section, we compare the performance of OFDMA and OFDM for small payloads under an intermediate traffic load.

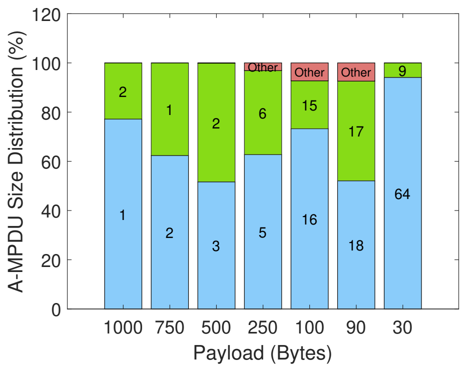


Figure 3.11: Aggregation Statistics for varying Payload Size (No OFDMA)

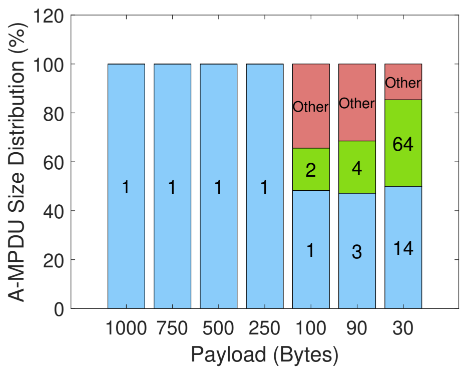


Figure 3.12: Aggregation Statistics for varying Payload Size (18×RU-26 OFDMA)

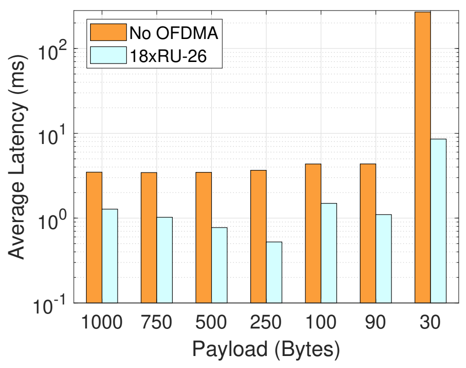


Figure 3.13: Average Latency for varying Payload Size

#### 3.2.3.1 Simulation Setup:

We once again utilize the same overall simulation parameters mentioned in Table 3.1. In order to analyze the effect of decreasing payload size on performance under an intermediate data rate, we reduce the payload size from 1000 B to 30 B while keeping the aggregate downlink data rate fixed at 60 Mbps. In doing so, we keep an eye on the per-station A-MPDU sizes being generated and also measure the average overall packet latency and aggregate throughput.

#### 3.2.3.2 Intermediate Payload Size:

Figure 3.13 shows that for payload sizes in the range 1000 - 250 B, the average overall packet latency for OFDM steadily increases, whereas for OFDMA, it gradually decreases. Figures 3.11 and 3.12 provide a distribution of *per-station* A-MPDU sizes over the duration of the simulation for each payload size. As the payload size decreases from 1000 to 250 B, the per-station A-MPDU size increases from 1 to 6 for OFDM whereas for OFDMA it remains 1. This implies that as the payload size decreases, queuing occurs for OFDM (since the packet arrival rate increases), but not for OFDMA. The underlying reason for this difference is the ability of OFDMA to serve more clients within a time window compared to OFDM (Table 3.3). Moreover, for OFDMA, as the payload size decreases, the transmission time also decreases (since no aggregation takes place). This explains why the latency decreases for OFDMA as the payload size decreases.

#### 3.2.3.3 Small Payload Size:

As the payload size decreases from $250$ B to $100$ B, the average overall packet latency now increases for both OFDM and OFDMA (Figure 3.13). Figure 3.12 indicates that this is because queuing has now also begun for OFDMA. Further, as the payload size decreases by ten more bytes to $90$ B, OFDMA again registers a decrease in latency since a lower transmission time is associated with a smaller payload size. Figures 3.11 and 3.12 also show that in the payload size region $100 - 90$ B, per-station A-MPDU sizes are smaller for OFDMA compared to OFDM. This implies that even under a high packet arrival rate, OFDMA is better able to counter packet queuing compared to OFDM due to its ability to serve more clients in a given time window.

#### 3.2.3.4 Very Small Payload Size:

For very small payload sizes ($30$ B), we observe a drastic difference in the performance of OFDMA and OFDM. At high packet arrival rates ($250,000$ packets/second), the queue completely fills up for OFDM. As a result, the average overall packet latency increases exponentially compared to the $90$ B payload case (where the arrival rate was $83,333$ packets/second). However, for OFDMA, neither does the queue completely fill up nor does OFDMA report an exponential increase in latency. Compared to the $90$ B payload case, the overall average packet latency in this case only increases by $6.7\times$. Moreover, OFDM also experiences a steep drop in throughput to $44$ Mbps, whereas no such drop is experienced by OFDMA. Thus, OFDMA significantly outperforms OFDM for the case of small packets.

#### 3.2.3.5 A Noteworthy Trend

A noteworthy observation from Figure 3.13 is that as the payload size decreases from $1000$ to $250$ B, OFDMA experiences a decrease in latency, which is attributed to no queuing and lower transmission time associated with smaller payload size. However, further reduction in payload size from $100$ to $30$ B results in a substantial increase in latency due to an increased packet arrival rate which leads to increased queuing as shown in Figure 3.12.

## 3.3 Conclusion

In this work, we investigated the performance of OFDMA in IEEE 802.11ax using the latest models supplied by the open source network simulator ns-3. We first analysed the performance of OFDMA under various downlink traffic loads to find out the scenarios in which multi-user transmissions outperformed single-user transmissions. Our experimental results establish that the true benefits of OFDMA over OFDM can only be extracted under intermediate traffic loads. At high traffic loads, overheads associated with OFDMA transmissions decrease the time duration available for transmitting aggregated frames. On the contrary, for low traffic loads single-user transmissions easily catch up to multi-user transmissions since in the latter case most time is spent waiting for the arrival of packets in the queue. We then analysed the performance of OFDMA under intermediate traffic load scenarios in various application settings and showed that for live video streaming applications, it provides more consistent bitrate and latency compared to OFDM. For typical video conferencing settings, OFDMA provides significantly lower latency and reduced network jitter. And finally, for web-based applications and factory IoT-based motion control applications, OFDMA far outperforms OFDM in terms of its ability to counter packet queuing.

Lastly, we observed that as the payload size decreases, OFDMA experiences an increase in latency due to an increased packet arrival rate. A usage survey by the Flexible Factory Project [2] outlines several wireless application in current use in factories. Many of these applications, such a safety monitoring and equipment control exchange small payloads with stringent deadline constraints and a very high communication rate. Moreover, the node density for such applications is also quite high (upto $50$ for human safety and $20$ for equipment control [1]). Therefore, for a factory environment with such applications, there is a need for intelligent scheduling techniques to ensure that the deadline requirements for critical applications are met under a high packet arrival rate. On this note, we propose a Minimize Deadline Penalty Scheduler in the next chapter that exploits knowledge of the packet arrival patterns and their deadlines in flexible factory environments to minimize the total missed packet deadlines for important applications.

Chapter 4

# Driven by Deadlines: Optimizing OFDMA for Deadline Constrained Settings

A usage survey by the Flexible Factory Project [2] categorized the various wireless applications in current use in factories. Some of the major categories include: (1) Quality Supervision, (2) Factory Resource Management, (3) Human Safety. The survey also provides the details of the communication requirements for these applications. The requirements for some of these have been tabulated in Table 4.1 below. It can be noted from the table that many applications have delay tolerance values as low as a few milliseconds, while others have comparatively relaxed tolerance values of the order of hundreds of milliseconds or few seconds. Another point to note is that for some applications, such as human safety monitoring and equipment control, the rate of communication is very high, i.e, once every one or two milliseconds.

In this work, we design a Minimize Deadline Penalty (MDP) scheduler for a flexible factory setting involving uplink traffic composed of small payloads. The scheduler utilizes known packet arrival patterns and deadline requirements for the applications in a factory setting to minimize the number of packet deadlines missed. We implement our scheduler and simulate the factory scenario in ns-3. We begin this work by first discussing the mechanisms of uplink OFDMA transmissions in 802.11ax (Section 4.1). We then formulate the scheduling problem in Section 4.2 as an Integer Linear Programming Problem. Afterwards, sections 4.3 and 4.4 outline

| **Wireless Application** | **Information** | **Data Size (Bytes)** | **Communication Rate** | **Delay Tolerance** |
|---|---|---|---|---|
| Equipment Control | | | | |
| Control of liquid injection | Water volume | 64 | Once per 1 min | 100 ms |
| AGV Control | Go signal, positioning | 100 | Once per 1 min | 100 ms |
| Bottle filling | Fill valves | 400 | Once per 1 ms | 0.5 ms |
| Warehouse | Stacker crane positioning | 10 | Once per 2 ms | 1 ms |
| Quality Supervision | | | | |
| Monitoring of equipment | State of tools, disposables | A few hundreds | Once per 1 s | 1 s |
| Counting number of wrench operations | Pulses, disposables | 64 | Once per 1 min | 100 ms |
| Detect defect state | Defect information | 500 | Once per 100 ms | 500 ms |
| Factory Resource Management | | | | |
| Movement analysis | Wireless beacon | A few tens | Twice per 1 s | A few seconds |
| Work record | Text data | 100 | Once per 1 min | 1 s |
| Human Safety | | | | |
| Detect entry in the proximity of a machine | Position of human | 10 - 30 | Once per 1 - 10 ms | 2 - 20 ms |

Table 4.1: Communication Requirements for various Wireless Applications in a Factory

two variants of our Minimize Penalty Deadline scheduler, Optimal and Heuristic, for solving the scheduling problem. Finally, in section 4.5 we evaluate the performance of our proposed Minimize Deadline Penalty scheduler by simulating a typical factory scenario in ns-3 based on the applications mentioned in Table 4.1.

## 4.1 Towards Centralized Channel Access

802.11ax introduces uplink and downlink OFDMA in WiFi, coordinated by the access point. The standard also introduces a Multi-User (MU) EDCA procedure that allows the access point to tweak the priorities of nodes contending for channel access via an advertised MU EDCA parameter set. The MU EDCA parameter set contains contention window parameters to be used by stations for channel access contention via the EDCA (Enhanced Distributed Channel Access) mechanism, along with a countdown timer.

Uplink OFDMA transmissions are solicited from the stations by the access point via the downlink transmission of a trigger frame. The trigger carries various transmission related parameters, including station-to-RU mapping related information. Upon reception of the trigger, the stations wait for a SIFS duration before participating in a synchronized uplink OFDMA transmission. After successfully performing an uplink OFDMA transmission, the stations change their contention window parameters to the MU EDCA parameters advertised by the access point for a duration specified by a countdown timer. A long value for the MU EDCA countdown timer can completely suppress station contention for channel access in the network. In such a situation, the stations become dependent on uplink OFDMA transmissions solicited by the access point to transmit data. And so, the transmission paradigm for stations shifts from being distributed contention-based, to centralized polling based.

In a network with station contention completely suppressed, all channel access decisions are made by the access point. This removes the non-determinism associated with randomised channel access schemes such as DCF/EDCA, thus removing a major obstacle limiting WiFi deployment in factory networks. We utilize these features to design a Minimize Deadline Penalty scheduler in this work. The scheduler at the access point generates a schedule specifying the station-to-RU mapping for each station. The mapping is transmitted to relevant stations via a downlink trigger frame. A SIFS duration after the trigger is received, the stations perform an uplink OFDMA transmission. Station contention for channel access is completely suppressed via the MU EDCA procedure.

## 4.2 The Deadline-based Scheduling Problem

In this section, we formulate our Deadline-based scheduling problem as an Integer Linear Programming Problem.

### 4.2.1 Notation

To aid the formulation of our problem, and to provide a more concise description of the various objects in our problem domain, we introduce some mathematical notation.

Let the time axis be divided into discrete time quanta $[t_0, t_1], [t_1, t_2] \dots [t_{n-1}, t_n]$ of fixed duration $\Delta = 1$ ms. Thus, $t_i = i * \Delta$.

And, let $m$ packets $p_0, p_1, \dots p_{m-1}$ be generated by $l$ applications $A_s = \{A_0, A_1, \dots A_{l-1}\}$ over $n$ time instants $t_0, t_1, \dots t_{n-1}$. Where, for each time instant, we have $j$ equal sized RUs available $r_0, r_1 \dots r_{j-1}$.

Moreover, each packet $p_i$ has an associated arrival time $a_i$ and deadline $d_i$. And each wireless application $A_i$ is described using the integer tuple $< T_i, s_i, d_i, P_i >$, based on the information in Table 3.4.

Where, $T_i$ is the time period of packet generation/communication rate for the application.
$s_i$ is the size of the application packets.
$d_i$ is the delay tolerance for the application packets.
$P_i$ is the penalty/priority of the application packets.

### 4.2.2 Assumptions

Before we proceed with a concrete formulation of our scheduling problem, it is important to enumerate the assumptions about the nature of our problem,

1. **Packet deadlines and arrival patterns known:** We assume that the access point is aware of the deadlines and packet arrival patterns of each wireless application deployed in the

network. This is a valid assumption in a factory setting where each application has well known characteristics as mentioned in Table 4.1.

2. **Only one application per station:** Every station has one and only one application running on it which generates packets according to a configured time period. This assumption is true for most IoT devices which only serve a single function, e.g., transmit data collected from sensors.

3. **Packets dropped on expiration of deadline:** Whenever a packet's deadline expires, it is immediately dropped from the access point queue. The applications can be easily configured to meet this requirement.

4. **No packet queuing at the stations:** A new packet for an application is not generated until the last packet's deadline has expired, and the packet has been dropped as a consequence. Mathematically, this means that $T_i > d_i$ for each application. In this work, we only consider applications that meet this requirement by generating periodic traffic.

5. **A fixed resource unit configuration:** Since the payload sizes $s_i$ are small, the best way to split the bandwidth into resource units (RUs) is to choose a RU configuration that maximizes the number of packets that can be transmitted. For example, for 40 MHz bandwidth, if 20 packets would arrive for transmission over a time window $[t_i, t_j]$, choose the $18 \times 26$-tone resource unit configuration. If 3 packets arrive, choose the $4 \times 106$-tone configuration (one resource unit would be wasted in this case). Therefore, as an input to our scheduling problem, we have a fixed RU configuration that is decided based on the number of packets that would be arriving over the considered time window for scheduling. All possible RU configurations are given in Figure 3.1.

6. **Packets arrive only at the fixed time instants:** We assume that all applications were started at the beginning of some time instant. This assumption aids in the modelling of our scheduling problem with minimal deviation from reality, since our considered value of $\Delta = 1$ ms is very small.

7. **Same MCS value for all stations:** We assume that every station observes similar channel conditions and is thus assigned the same MCS value. This assumption has been made to

simplify the design of our scheduler.

8. **Transmissions contained within a time quantum:** Every transmission starts at the beginning of a time quantum and ends before the next time quantum starts. In effect, every transmission is contained within some time quantum and lasts for a maximum duration of $1$ ms. In order to achieve this, we ensure that the transmission duration $\tau$, which is a function of the chosen RU configuration $r$, the MCS value assigned to all stations $m$ and the size of the largest payload in the transmission $s = max\ (s_k)\ \forall k$ ($k$ is the number of stations participating in the transmission), is less than the duration of a time quantum $\Delta$. Mathematically, $\tau(r, m, s) < \Delta$.

9. **Only one transmission allowed per time quantum:** Even if a transmission ends well before the end of the time quantum in which it was scheduled, the next transmission cannot start until the beginning of the next time quantum. This requirement has been imposed only to aid the design of our scheduler.

### 4.2.3 Integer Linear Programming Formulation

Let $z_p^t$ be a decision variable which is equal to 1 or 0 depending on whether the $p^{th}$ packet is scheduled at time instant $t$ or not.
Let $y_r^t$ be a decision variable which is equal to $1$ or $0$ depending on whether the $r^{th}$ RU is allocated to some packet at time instant $t$ or not.
Let $x_{pr}^t$ be a decision variable indicating that $p^{th}$ packet is allocated $r^{th}$ RU at time instant $t$.

Then, the relationship between $x_{pr}^t$ and $z_p^t$ can be defined as:

$$\sum_r x_{pr}^t = z_p^t, \forall p, r. \tag{4.1}$$

Equation 4.1 models the constraint that in a particular transmission, starting at time instant $t$, one packet can be assigned at most one RU.

Similarly, the relationship between $x_{ij}^t$ and $y_j^t$ can be defined as:

$$\sum_p x_{pr}^t = y_r^t, \forall r, t. \tag{4.2}$$

Equation 4.2 models the constraint that in a particular transmission, starting at time instant $t$, one RU can be assigned to at most one packet.

Let $\lambda_p$ denote a variable which is equal to 1 or 0 depending on whether the $p^{th}$ packet was transmitted in some time instant or not. Then, we have:

$$\lambda_p = \sum_{t \in [a_p, d_p]} z_p^t, \forall p, \tag{4.3}$$

Where $a_p$ and $d_p$ denote the arrival time instant and the deadline of that packet respectively.

Let the total number of carriers available be given by $B$, and the number of carriers required to utilize $y_{rt}$ be denoted by $b_r$. Then, the budget constraint is given by:

$$\sum_r y_r^t b_r \leq B, \forall t. \tag{4.4}$$

If $I_p$ is the penalty for not transmitting the $p^{th}$ packet. Our objective is to minimize the sum of total penalties. We write this as:

$$\textbf{Minimize} \sum_p (1 - \lambda_p) I_p. \tag{4.5}$$

or equivalently,

$$\textbf{Maximize} \sum_p \lambda_p I_p. \tag{4.6}$$

The objective function in Eq. 4.5 subject to the given constraints is an integer linear programming problem (ILP). A solution to this ILP would provide a schedule for packet transmission that minimizes the total penalty incurred due to dropped packets.

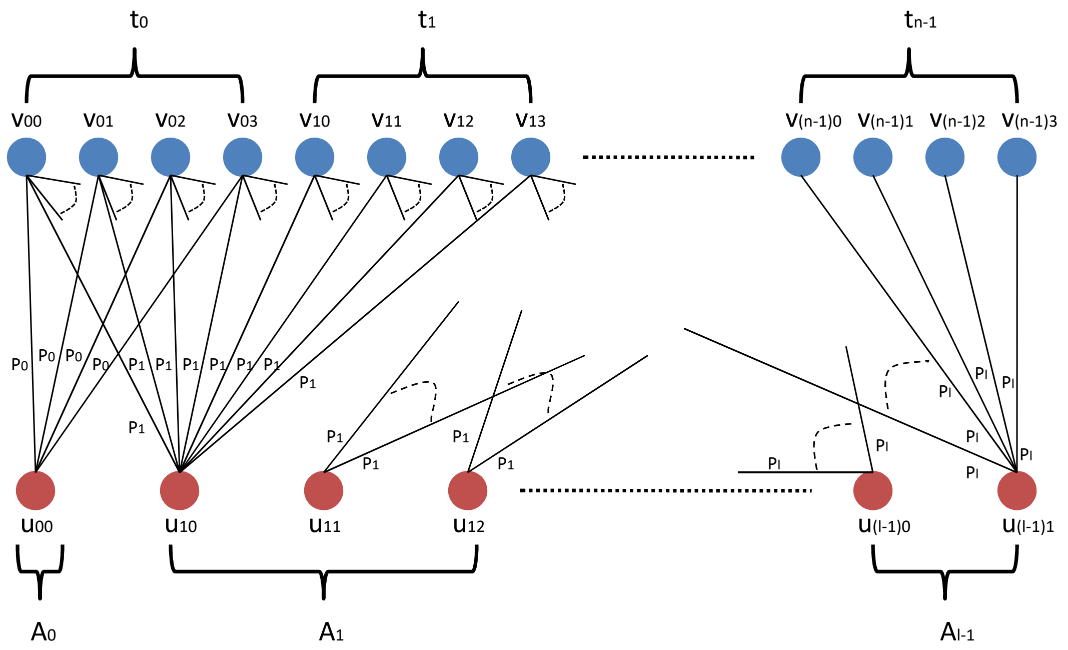
In general, ILPs are NP-Hard, thus, it is undesirable to solve the formulated ILP via conventional optimization solvers. Therefore, in the next section, we propose a polynomial time algorithm to solve the scheduling problem.

## 4.3 An Optimal Algorithm for the Scheduling Problem

The scheduling problem defined earlier can be mapped to the problem of finding the Maximum Weighted Matching in an Undirected Bipartite Graph. A matching in an undirected graph is a set of edges with no common vertices. Optimal algorithms for solving the Maximum Weighted Bipartite Matching problem are well known, and take almost linear time $O(n^2logn + ne)$ to execute.

### 4.3.1 Mapping Procedure

We now describe the mapping procedure between our scheduling problem and the Maximum Weighted Bipartite Matching problem.



t0
t1
tn-1
v00 v01 v02 v03 v10 v11 v12 v13
v(n-1)0 v(n-1)1 v(n-1)2 v(n-1)3
P0 P0 P0 P0 P1 P1 P1 P1 P1 P1
P1
Pl
u00 u10 u11 u12 u(l-1)0 u(l-1)1
A0 A1 Al-1

Figure 4.1: Graph Constructed by Minimize Deadline Penalty Scheduler

As an input to our scheduling problem, we are given the set $C_s = \{< T_0, s_0, d_0, P_0 >, <$

$T_1, s_1, d_1, P_1 >, \cdots < T_{l-1}, s_{l-1}, d_{l-1}, P_{l-1} >\}$ that describes the $l$ wireless applications $A_s = \{A_0, A_1, \ldots A_{l-1}\}$ in our factory setting, and a fixed RU configuration containing $j$ equal sized RUs.

**Algorithm 1:** Optimal Minimize Deadline Penalty Scheduling Algorithm

```
def OptimalMinimizeDeadlinePenaltyScheduling($C_s$, $A_s$, $j$, $t_i$):
 $n = LCM\ (T_0, T_1, \ldots T_{l-1})\ -1$
 Create $t_s = \{t_i, t_{i+1}, \ldots t_{i+n-1}\}$
 $m = \sum_i n + 1/T_i$
 Create $p_s = \{p_0, p_1, \ldots p_{m-1}\}$

 $G =< V_1 \cup V_2, E, W >=$ ConstructBipartiteNullGraph($m$, $n * j$)

 for each $A \in A_s$:
   for each $p \in p_s$:
     for each $t \in t_s$:
       if $t \in [a_{Ap}, d_{Ap}]$:
       for each $r \in r_s$:
         $u_{Ap} = V_1$.find($A, p$)
         $v_{tr} = V_2$.find($t, r$)
         $E$.add($u_{Ap}, v_{tr}$)
         $W$.add($u_{Ap}, v_{tr}$, $P_A$)

 $E^{'}$ = MaximumWeightedBipartiteMatching($G$)

 return $E^{'}$
```

Based on the time periods of packet arrival for the different applications, $T_0, T_1, \ldots T_{l-1}$, it evident that the pattern of packet arrival repeats are every $n + 1 = LCM\ (T_0, T_1, \ldots T_{l-1})$ time instants.

Moreover, application $A_i$ generates $n + 1/T_i$ packets before its packet arrival pattern repeats at time instant $t_{i+n+1}$. Hence, over $n$ time instants $t_s = \{t_i, t_{i+1}, \ldots t_{i+n-1}\}$, total $m = \sum_i n+1/T_i$ packets $p_s = \{p_0, p_1, \ldots p_{m-1}\}$ arrive. And for each time instant, we have $j$ equal sized RUs available $r_s = \{r_0, r_1 \ldots r_{j-1}\}$.

We construct the weighted undirected bipartite graph $G =< V_1 \cup V_2, E, W >$ such that, $|V_1| = m$ and $|V_2| = n * j$ shown in Figure 4.1. That is, the nodes in $V_1$ represents packets and the nodes in $V_2$ represent available RUs at different time instants.

A node $u_{Ap}\ \epsilon\ V_1$ represents the $p^{th}$ packet of the $A^{th}$ application. Similarly, a node $v_{tr}\ \epsilon\ V_2$

represents the $r^{th}$ RU available at the time instant $t$. Also, $[a_{Ap}, d_{Ap}]$ denotes the time window in which the $p^{th}$ packet of the $A^{th}$ application can be scheduled, since $a_{Ap}$ is the arrival time of the packet and $d_{Ap}$ is the deadline of the packet.

Then, in graph $G$, $\exists\ (u_{Ap}, v_{tr}) \in E$ if and only if $t \in [a_{Ap}, d_{Ap}]$.

That is, if the $p^{th}$ packet of the $A^{th}$ application is available for scheduling in the time window $[a_{Ap}, d_{Ap}]$, then there would be an edge from $u_{Ap}$ in $V_1$ to all $v_{tr}$ in $V_2$ where $t\ \epsilon\ [a_{Ap}, d_{Ap}]$. Hence, if a packet arrives at time instant $t_0$ and has a deadline at time instant $t_n$, then there would be an edge from the node corresponding to the packet in $V_1$ to *every* node in $V_2$. On the other hand, if the packet arrives in $t_0$ and has the deadline in $t_0$ itself, then there would be an edge from the packet's node to the nodes $v_{00}, v_{01}, \ldots v_{0j}\ \epsilon\ V_2$, given that $j$ RUs are available. For example, in Figure 4.1 packet $u_{10} \in V_1$ arrives at time instant $t_0$ and has a deadline at $t_1$, so there is an edge from $u_{10}$ to $v_{00}, v_{01}, \ldots v_{13}$ reflecting the fact that $u_{10}$ can be assigned any available RU in those two time instants.

Finally, $\forall\ (u_{Ap}, v_{tr})\ \epsilon\ E$, we assign a weight $P_A$ which represents the penalty/priority associated with each packet of the $A^{th}$ application.

At last we input our constructed graph $G$ to the $MaximumWeightedBipartiteMatching$ algorithm, which returns an optimal matching map $E'$ assigning each packet in $V_1$ to at most one RU in $V_2$ for $n$ future time instants.

### 4.3.2 Pitfalls

The careful reader would note that the value of the variable $n$ in Algorithm 1 could take exceptionally large values if the time periods $T_0, T_1, \ldots T_{l-1}$ are not chosen carefully. Even when the time periods are chosen carefully, a manageable but still sufficiently large value of $n$ would result in an undesirably long execution time since it would result in a very large and possibly dense graph $G$ being passed as input to the $MaximumWeightedBipartiteMatching$ algorithm.

To overcome this drawback, we propose a modified variant of the Optimal Minimize Deadline

Penalty Scheduling Algorithm in the next section. Needless to say, the modification trades the optimal solution provided by the current algorithm with better execution time.

## 4.4 A Heuristic for the Scheduling Problem

An simple remedy for the pitfalls discussed for our Optimal Minimize Deadline Penalty Scheduling algorithm is to fix the value of the variable $n$ by providing it as input to the algorithm. This has the direct effect of fixing the number of future time instants for which our scheduler generates a schedule. In order to better understand the challenges this remedy brings to the design of the scheduler, consider the following scenario:

Let us assume the following time window $[t_0, t_{3n}]$, where $3n + 1 = LCM\ (T_0, T_1, \ldots T_{l-1})$. A packet $p_i$ arrives at time instant $t_0$ and has a deadline at $t_{3n}$. Our scheduler is invoked at time instants $t_0$, $t_n$ and $t_{2n}$ to generate schedules for the considered time window. Now, a node corresponding to packet $p_i$ must appear in the graph $G_1$ constructed for the first of the three sub-windows, $[t_0, t_n]$. If $p_i$ is not scheduled in this sub-window (perhaps because it had a low penalty/priority compared to other packets competing for transmission), a node for it must appear in the graph $G_2$ for the second sub-window $[t_n, t_{2n}]$. Continuing its bad luck, if the packet is not scheduled even in the second sub-window, a node for it must appear in the graph $G_3$ for sub-window $[t_{2n}, t_{3n}]$. Finally, if the packet is not scheduled even in the last sub-window, it must be dropped from the queue. On the other hand, if the packet was scheduled in the first sub-window $[t_0, t_n]$, then a node corresponding to it must *not* appear in the graphs $G_2$ and $G_3$. Hence, we must now maintain a record $X$ of packets that have already been scheduled in an earlier time window.

This also has the indirect effect of modifying the value of the variable $m$ on **line** 4 of Algorithm 1. We now depend on procedure $getPackets$ to set the value of $m$, which tells the number of packets arriving in the next $n$ time instants. $getPackets$ consults the time periods of packets from $C_s$ along with a record $X$ of packets already scheduled at earlier time instants to compute this information. Simply put, it ensures that packets that were scheduled in earlier rounds, do not re-appear as nodes in the graph $G$ being constructed.

However, the most significant effect of the suggested remedy is a deviation from optimality. We discuss this in further detail in Section 4.4.2.

### 4.4.1 Modified Mapping Procedure

We now describe our modifications to the Optimal Minimize Deadline Penalty Scheduling Algorithm.

**Algorithm 2:** Heuristic Minimize Deadline Penalty Scheduling Algorithm

```
$X = \phi$

def HeuristicMinimizeDeadlinePenaltyScheduling($C_s$, $A_s$, $j$, $t_i$, $n$):
 Create $t_s = \{t_i, t_{i+1}, \ldots t_{i+n-1}\}$
 $m$ =getPackets($C_s$, $X$, $t_i$)
 Create $p_s = \{p_0, p_1, \ldots p_{m-1}\}$

 $G =< V_1 \cup V_2, E, W >=$ ConstructBipartiteNullGraph($m$, $n * j$)

 for each $A \in A_s$:
   for each $p \in p_s$:
     for each $t \in t_s$:
       if $t \in [a_{Ap}, d_{Ap}]$:
       for each $r \in r_s$:
         $u_{Ap} = V_1$.find($A, p$)
         $v_{tr} = V_2$.find($t, r$)
         $E$.add($u_{Ap}, v_{tr}$)
         $W$.add($u_{Ap}, v_{tr}$, $P_A$)

 $E^{'}$ = MaximumWeightedBipartiteMatching($G$)

 for each $A \in A_s$:
   for each $p \in p_s$:
     if $E^{'}$.isMapped($u_{Ap}$):
       $X$.add(p)

 return $E^{'}$
```

Our Heuristic Minimize Deadline Penalty Scheduling Algorithm 2 has the following differences when compared to the Optimal version.

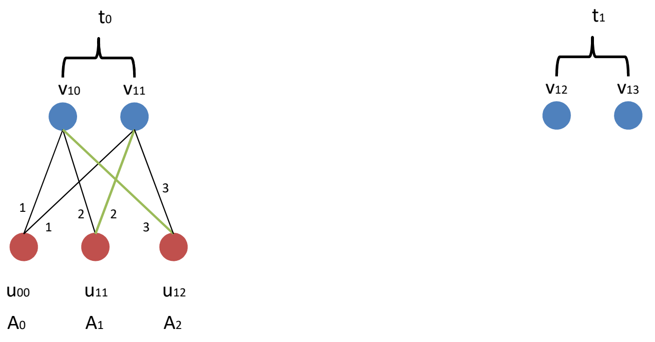
- The number of future time instants $n$ for which a schedule has to be generated is now provided as input.

- After $MaximumWeightedBipartiteMatching$ returns the map $E^{'}$, we add a record corresponding to each packet that was assigned an RU into $X$.
- The method $getPackets$ now computes the number of packets arriving in the next $n$ time instants by consulting $C_s$, $X$ and the current time instant $t_i$.

### 4.4.2 When Do We Deviate From Optimality?

Consider the following example scenario. We have three applications described by the integer tuples $< T_i, s_i, d_i, P_i >$, specifically, $A_0 =< 2, 10, 0, 1 >$, $A_1 =< 2, 30, 1, 2 >$ and $A_2 =< 2, 100, 1, 3 >$. Let, $C_s = \{A_0, A_1, A_2\}$. We set $j = 2$ to represent the $2x242$-tone RU configuration.

We first provide this input to the Heuristic Minimize Deadline Penalty Scheduler with $t_i = t_0$, and $n = 1$ to imply that our scheduler generates a schedule only for the next time instant. Observe what happens over the time window $[t_0, t_1]$.



(a) Graph $G_1$ for time instant $t_0$ (b) Graph $G_2$ for time instant $t_1$

Figure 4.2: Sub-Optimal Packet-to-RU assignment by Heuristic Minimize Deadline Penalty Scheduler

At the time instant $t_0$, one packet is generated by each application and so $m = 3$. The scheduler generates the graph $G_1$ shown in Figure 4.2a above and gives it as input to the $MaximumWeightedBipartiteMatching$ algorithm, which outputs the matching shown via green edges. A penalty of 1 is incurred since the packet for application $A_0$ is dropped.

At time instant $t_1$, no new packet arrives, $m = 0$, and so the output $E'$ is empty. No penalty is incurred.

We now provide the original input to the Optimal Minimize Deadline Penalty Scheduling Algorithm with $t_i = t_0$ and again observe what happens over the time window $[t_0, t_1]$.

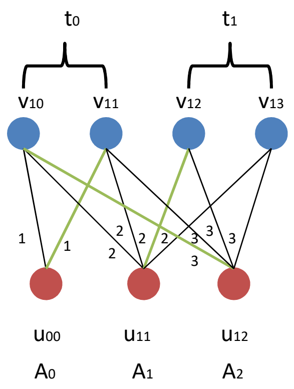


Figure 4.3: Optimal Packet-to-RU assignment by Optimal Minimize Deadline Penalty Scheduler

Clearly, $n = LCM(T_0, T_1, T_2) = LCM(2, 2, 2) = 2$ and $m = 3$. The scheduler generates the graph $G_3$ shown in Figure 4.3 and $MaximumWeightedBipartiteMatching$ generates the matching shown via green edges. No penalty is incurred.

One can quickly infer from this example that presence of applications with long packet deadlines and a high packet drop penalty along with a small value of the input $n$ cause the Heursitic variant of our scheduler to deviate from optimality. Had we given $n = 2$ as input to our Heuristic Minimize Deadline Penalty Scheduler, we would have been provided with the same optimal value of $0$ penalty incurred as provided by the Optimal Minimize Deadline Penalty Scheduler.

Fortunately, for most practical factory settings, rarely do we encounter applications with relaxed delay tolerances and high packet drop penalties. In most cases, a stringent delay tolerance is itself an indicator of high packet priority. Therefore, we expect our Heuristic scheduler to provide good performance in typical factory environments. With this expectation, in the next section, we evaluate the performance of our scheduler by implementing it and simulating a factory scenario in ns-3.

## 4.5 Results and Analysis

In this section, we simulate a factory setting consisting of the applications listed in Table 4.2 below, and evaluate the performance of our Optimal and Heuristic Minimize Deadline Penalty Schedulers against three other simple deadline-based schedulers, namely, Earliest Deadline First (EDF), Largest Penalty-To-Deadline Ratio First (LRF) and a Non-Starving variant of Largest Penalty-To-Deadline Ratio First (NLRF).

The NLRF scheduler sorts the stations based on the quantity,

$$\frac{Penalty/Deadline}{(BytesTransmitted + 1)/CurrentTimeInstant} \tag{4.7}$$

to ensure that the Penalty-To-Deadline ratio eventually increases for applications that are continually being starved. This does not always mean that the NRLF scheduler would result in lower total packet drop penalty incurred compared to the LRF scheduler, this is because if the simulation runs long enough, the NRLF scheduler will eventually start prioritizing packets will lower penalty over the ones with higher penalty and more frequent arrival time.

| **Application** | **Time Period (ms)** | **Size (B)** | **Delay Tolerance (ms)** | **Penalty** | **Nodes** |
|---|---|---|---|---|---|
| Equipment Control | | | | | |
| Bottle filling | 1 | 400 | 0.5 | 90 | 3 |
| Warehouse | 2 | 10 | 1 | 100 | 6 |
| Quality Supervision | | | | | |
| Monitoring of equipment | 1000 | 100 | 1000 | 50 | 4 |
| Detect defect state | 100 | 500 | 100 | 40 | 10 |
| Factory Resource Management | | | | | |
| Movement analysis | 500 | 20 | 500 | 10 | 10 |
| Human Safety | | | | | |
| Detect entry in the proximity of a machine | 10 | 30 | 2 | 200 | 7 |

Table 4.2: Characteristics of the 40 applications in our Simulated Factory Settings

Note from Table 4.2 that we have a total of 40 applications running on 40 stations, and packet drop penalties have been assigned in the order Human Safety > Equipment Control > Quality Supervision > Factory Resource Management. Since applications with longer delay tolerances, particularly Quality Supervision and Factory Resource Management, have been assigned lower penalties, we can hope that our Heuristic scheduler would provide good performance. Also, the largest payload size we deal with has a length of $500$ bytes, therefore, the RU configuration and the MCS value must be chosen such that the transmission duration does not exceed the duration of a time quantum $\Delta$.

### 4.5.1 Simulation Setup:

The parameters for our simulation are provided in the Table 4.3 below. We set the duration of the time quantum $\Delta = 1$ ms to minimize the idle time of our scheduler. We run our simulation for a duration of $LCM(T_0, T_1, \dots T_{n-1}) = LCM(1, 2, 1000, 100, 500, 1) = 1000$ ms because the packet arrival pattern repeats after this duration, and so the same schedule is generated by the Optimal Minimize Deadline Penalty Scheduler. Frame Aggregation is disabled since there is no packet queuing at the stations (A new packet arrives only after the deadline for the last packet has expired, implying that it was either transmitted or dropped). We first choose an RU configuration of composed of all $26$-tones to maximize the number of packets transmitted at each time instant, and then change the RU configuration to $4 \times 106$-tones to force more packet drops and thus analyze the effects of lower available RUs on performance. We set the MCS value to $11$ because it allows a data rate of approximately $1000$ bytes per milliseconds in each $26$-tone RU, and $6000$ bytes per milliseconds in each $106$-tone RU with the given Guard Interval (verified via simulations). Hence, we can be sure that within the duration of a time quantum, both the trigger frame, carrying the station-to-RU mapping, along with the Uplink OFDMA data frame are able to finish transmission.

| **Parameter** | **Value** | **Parameter** | **Value** |
|---|---|---|---|
| Stations | 40 | Radius | 5 m |
| MCS | 11 | Bandwidth | 40 MHz |
| Transport Protocol | UDP | Guard Interval | 3200 ns |
| Frame Aggregation | Disabled | Simulation Duration | 1 sec |

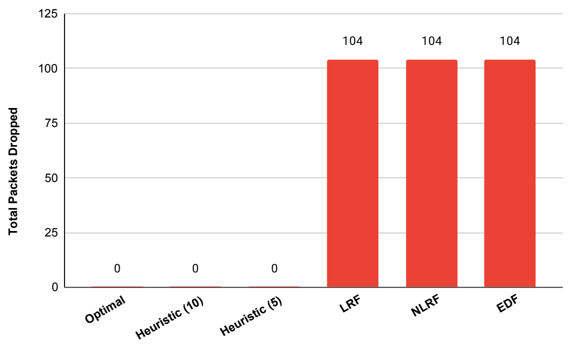
Table 4.3: Simulation Parameters

### 4.5.2 Results

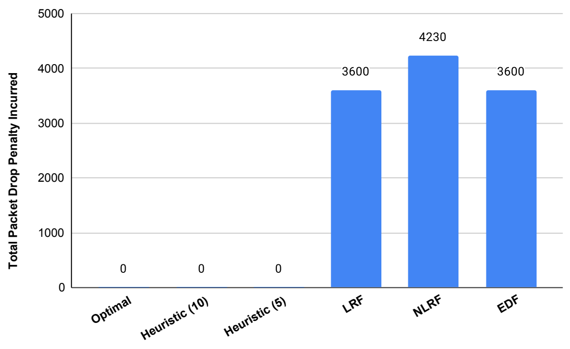
The metric of interest for us is the total number of packets dropped and the total packet drop penalty incurred. Figures 4.4 and 4.5 provide these metrics for the $18 \times 26$-tone and $4 \times 106$-tone RU configurations respectively.

*Heuristic ($x$)* in Figures 4.4 and 4.5 denotes the Heuristic Minimize Deadline Penalty Scheduler which generates a schedule for $x$ future time instants. *Optimal* denotes the Optimal Minimize Deadline Penalty scheduler, while *LRF* and *NLRF* denote the starving and non-starving variants of the Largest Penalty-To-Deadline Ratio First scheduler. *EDF* denotes the Earliest Deadline First Scheduler.

#### 4.5.2.1 $18 \times 26$-tone RU configuration



(a) Total Packets Dropped



(b) Total Penalty Incurred

Figure 4.4: Comparison of Scheduler Performance for $18 \times 26$-tone RU Configuration

Figure 4.4a shows that both the Optimal and Heuristic variants of our scheduler result in zero packet drops compared to 104 packet drops by Earliest Deadline First, Largest Penalty-To-

Deadline Ratio First (LRF) and its non-starving variant NLRF.

Figure 4.4b shows that both the Optimal and Heuristic variants also result in zero total packet drop penalty incurred. This is in stark contrast to 3600, 4230, and 3600 penalty values reported by the LRF, NRLF and EDF schedulers respectively.

#### 4.5.2.2 $4 \times 106$-tone RU configuration

Figure 4.5a shows that both the Optimal and Heuristic variants of our scheduler result in 2824 packet drops compared to 3324 packet drops by Earliest Deadline First, Largest Penalty-To-Deadline Ratio First (LRF) and its non-starving variant NLRF. A 15% reduction in dropped packets, even though only 4 RUs are available for transmission.

Figure 4.5b shows that both the Optimal and Heuristic variants result in a total packet drop penalty of 250400 compared to 384400, 339730, and 394400 penalty values reported by the LRF, NRLF and EDF schedulers respectively. Thus, the Minimize Deadline Penalty Scheduler saves at least 34.8% in packet drop penalty payment.

#### 4.5.2.3 Execution Times

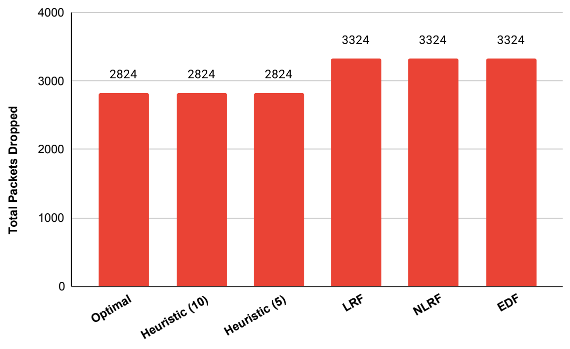
Table 4.4 provides execution times for both the variants of our Minimize Deadline Penalty scheduler on an Intel(R) Core i7-9700K 3.60GHz CPU. It can be clearly seen that the Heuristic versions execute significantly faster compared to the Optimal version since the Optimal variant constructs and finds a matching in a much bigger graph.

| **Variant** | **Execution Time (ms)** |
|---|---|
| $18 \times 26$-tone RU configuration | |
| Optimal Minimize Deadline Penalty | $2,584$ |
| Heuristic (5) Minimize Deadline Penalty | 3 |
| Heuristic (10) Minimize Deadline Penalty | 6 |
| $4 \times 106$-tone RU configuration | |
| Optimal Minimize Deadline Penalty | 574 |
| Heuristic (5) Minimize Deadline Penalty | 2 |
| Heuristic (10) Minimize Deadline Penalty | 4 |

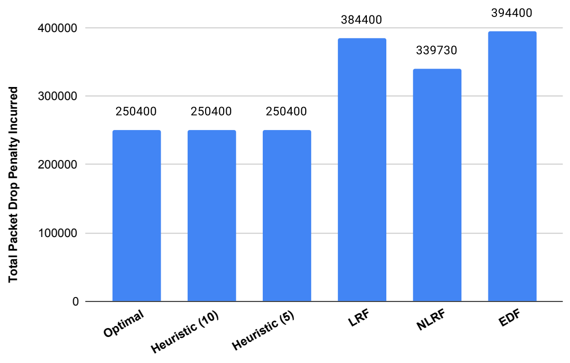
Table 4.4: Execution Times for the Minimize Deadline Penalty Scheduler variants

### 4.5.3 Analysis and Insights

Here, we first discuss the differences between our Minimize Deadline Penalty scheduler and the state-of-the-art deadline-based scheduler by Inam et al.[17]. (1) The scheduler proposed by [17] uses a deadline estimation approach that utilizes the station queue length provided by the buffer status reports collected by the access point from the stations. On the other hand, our scheduler assumes that the access point is already aware of the deadlines for each application running on a station.(2) Once deadlines have been estimated, their scheduler sorts the stations in an earliest-deadline-first order and then performs an exhaustive search over all RU configurations to find the one that results in the least expected packet drops. It utilizes a recursive mathematical expression to estimate the number of packets dropped when the first $m$ sorted stations are assigned the first $m$ RUs (RUs sorted by their size, where $m$ is the number of available RUs) for a given RU configuration. Our Minimize Deadline Penalty Scheduler does not perform an exhaustive search since it always chooses an RU configuration that splits the bandwidth into



(a) Total Packets Dropped



(b) Total Penalty Incurred

Figure 4.5: Comparison of Scheduler Performance for $4 \times 106$-tone RU Configuration

equal sized RUs to maximizes the number of packets that can be transmitted. This choice is justified since we only consider small payloads in our uplink traffic.

Based on the above two differences one can infer that for the small payload and known deadline scenario we are considering, the scheduler by [17] would simply sort the stations based on the known deadlines and then assign the first $m$ stations to the first $m$ RUs of the fixed RU configuration input to the scheduler. Thus, their scheduler reduces to a simple Earliest Deadline First (EDF) scheduler.

It is evident from Figures 4.4 and 4.5 that our scheduler far outperforms the Earliest Deadline First scheduler for the factory IoT setting. Both the Heuristic and the Optimal variants of our scheduler provide zero packet drops and result in zero penalty incurred when the $18 \times 26$-tone RU configuration is used, even for the $4 \times 106$-tone RU configuration, a $36.5\%$ reduction in penalty is observed compared to EDF. This is a consequence of the fact that our scheduler exploits knowledge of the packet arrival patterns to generate a schedule not just for the subsequent transmission but for several future transmissions. Similarly, the poor performance of the LRF and NLRF schedulers can be attributed to the fact that their scheduling decisions are based on local instantaneously available information only. Moreover, both the LRF and NLRF schedulers continue to exhibit starvation for certain applications, particularly for the ones with very long time periods.

Another interesting point to note is that neither $Heuristic(10)$ nor $Heuristic(5)$ deviates from the optimal value generated by $Optimal$. This is due to the fact that the our simulation does not include any application with long packet deadlines and high penalty, which could cause deviation from Optimality, as explained in Section 4.4.2. Table 4.3 points out that the execution time for the Heuristic Minimize Deadline Penalty scheduler is smaller than the number of time instants for which it is generating a schedule ($n = 5$ and $n = 10$ ms). This is a lucky co-incidence, for some settings it is possible that the execution time of the scheduler turns out to be longer than $n$ which would cause the scheduler to finish its scheduling operations before a schedule for the next $n$ time instants is generated. In such a scenario, the suggested remedy is to gain a head-start of $12 - 18$ ms by generating a schedule for 2 to 3 time windows when the *very first* schedule is being generated. Afterwards, schedule generation should proceed in parallel along with scheduling operations at the access point to ensure that a schedule is always available. This approach has the small disadvantage that newly associated stations must wait for $10$ to $20$ ms before a schedule that considers them is generated.

## 4.6 Conclusion

In this work, we implemented and evaluated the performance of our proposed Minimize Deadline Penalty Scheduler in ns-3 using the latest 802.11ax models supplied by the simulator. We proposed two variants of the scheduler, Optimal and Heuristic. For the Optimal variant, the time window for which a schedule is generated is governed by the time periods of the application's packet arrival patterns, while for the Heuristic variant, the time window for schedule generation is provided as input to the scheduler. We showed via simulations involving two RU configurations ($18 \times$ 26-tones and $4 \times$ 106-tones) that for a typical factory scenario, the Minimize Deadline Penalty scheduler far outperforms other deadline-based schedulers in terms of total penalty incurred due to missed packet deadlines. For $18$ RUs, Minimize Deadline Penalty scheduler provides a $100\%$ reduction in packet drop penalty compared to other considered schedulers. Even when only $4$ RUs are available for scheduling at every time instant, the Minimize Deadline Penalty Scheduler saves at least $34.8\%$ in packet drop penalty payment. Moreover, the performance of the Heuristic variant coincides with the performance of the Optimal variant for the factory scenario considered since it does not involve applications with long packet deadlines and high penalty.

# Chapter 5

# Conclusion and Future Work

In this work, we have utilized the latest 802.11ax models supplied by the open source network simulator ns-3 to investigate the performance of OFDMA in WiFi 6 in various settings, and have provided a solution for its optimization in deadline constrained factory settings. In Chapter 3, our investigations have provided insight into the behavior of OFDMA under varying traffic loads and have established its utility for intermediate downlink traffic scenarios. Further exploration of OFDMA performance under intermediate traffic loads has shed light on its superior performance in typical everyday application settings compared to OFDM. Even though OFDMA outperforms OFDM in most everyday settings, it cannot be adopted as-is for factory environments where the payload sizes are small, and the packet arrival rates high, along with stringent deadlines to meet.

Based on these insights, we proposed a Minimize Deadline Penalty scheduler in Chapter 4 to maximize the number of packet deadlines met in factory environments. Even though our proposed scheduler was designed under the assumption of small payload sizes, our experiments were able to showcase its ability to provide near optimal performance in a typical factory setting composed of a subset of the applications mentioned in Table 4.1.

Integration of wireless networks in factory environments has been long overdue, and WiFi standards have been taking concrete steps in this direction. For example, the newly emerging IEEE 802.11be standard (known as WiFi 7) is expected to tackle the problem of wireless link unreliability by introducing Multi-Link operation [33]. In line with these efforts our work provides reason for the consideration of WiFi as a feasible technology in the industrial space.

## 5.1 Future Work

The evaluation of OFDMA performance under varying traffic loads in this work has only focused on UDP as the underlying transport protocol. It is well known that a large number of application layer protocols including HTTP, SMTP, Telnet and FTP use TCP for transport [34]. Therefore, in future works, we plan to evaluate TCP performance in WiFi 6 via extensive simulations to understand what benefits can OFDMA offer over OFDM under varying traffic loads and for various applications settings that utilize TCP.

The scheduling problem for the Minimize Deadline Penalty scheduler proposed in this work was formulated under two important assumptions, that the duration of a transmission is always less than $1$ ms, and that the best RU configuration is the one that maximizes the number of packets transmitted. These assumption holds true for the applications in a factory that transmit only small payloads, primarily Human Safety and Equipment Control applications. However, a large number of applications related to Quality Supervision and Resource Management in a flexible factory setting generate bigger payloads, of the order of thousands of bytes, that require more than a millisecond to transmit. Moreover, when there is a large variation in the size of packets, the best RU configuration is not always the one that maximizes the number of RUs available for transmission. Therefore, as a next step, we plan to extend the scope of our Minimize Deadline Penalty Scheduler to allow transmission of bigger payload sizes via an intelligent choice of the RU configuration.

# Bibliography

[1] "IEEE 802 nendica report: Flexible factory IoT: Use cases and communication requirements for wired and wireless bridged networks," pp. 1–48.

[2] Flexible Factory Project, "Wireless use cases and communication requirements in factories," NICT, Tech. Rep., February 2018. [Online]. Available: https://www2.nict.go.jp/wireless/en/ffpj-wp.html

[3] M. Maity, B. Raman, and M. Vutukuru, "Tcp download performance in dense wifi scenarios: Analysis and solution," *IEEE Transactions on Mobile Computing*, pp. 213–227, 2016.

[4] W. Liang, J. Zhang, H. Shi, K. Wang, Q. Wang, M. Zheng, and H. Yu, "An experimental evaluation of WIA-FA and IEEE 802.11 networks for discrete manufacturing," vol. 17, no. 9, pp. 6260–6271, conference Name: IEEE Transactions on Industrial Informatics.

[5] G. Cena, S. Scanzio, and A. Valenzano, "Seamless link-level redundancy to improve reliability of industrial wi-fi networks," vol. 12, no. 2, pp. 608–620, conference Name: IEEE Transactions on Industrial Informatics.

[6] L. Seno, G. Cena, S. Scanzio, A. Valenzano, and C. Zunino, "Enhancing communication determinism in wi-fi networks for soft real-time industrial applications," vol. 13, no. 2, pp. 866–876, conference Name: IEEE Transactions on Industrial Informatics.

[7] "Ieee p802.11ax - 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."

[8] "Status of project ieee 802.11ax high efficiency (he) wireless lan task group," [accessed 09-July-2020]. [Online]. Available: http://www.ieee802.org/11/Reports/tgaxupdate.htm

[9] S. Avallone, P. Imputato, G. Redieteab, C. Ghosh, and S. Roy, "Will ofdma improve the performance of 802.11 wifi networks?" *IEEE Wireless Communications*, 2021.

[10] K. Wang and K. Psounis, "Scheduling and resource allocation in 802.11 ax," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018.

[11] ——, "Efficient scheduling and resource allocation in 802.11 ax multi-user transmissions," *Computer Communications*, 2020.

[12] D. Bankov, A. Didenko, E. Khorov, and A. Lyakhov, "Ofdma uplink scheduling in ieee 802.11 ax networks," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018.

[13] D. Weller, R. D. Mensenkamp, A. v. d. Vegt, J.-W. v. Bloem, and C. d. Laat, "Wi-fi 6 performance measurements of 1024-qam and dl ofdma," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[14] M. S. Kuran, A. Dilmac, O. Topal, B. Yamansavascilar, S. Avallone, and T. Tugcu, "Throughput-maximizing OFDMA Scheduler for IEEE 802.11ax Networks," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–7.

[15] P. K. Sangdeh and H. Zeng, "Deepmux: Deep-learning-based channel sounding and resource allocation for ieee 802.11 ax," *IEEE Journal on Selected Areas in Communications*, 2021.

[16] D. Bankov, A. Didenko, E. Khorov, V. Loginov, and A. Lyakhov, "Ieee 802.11 ax uplink scheduler to minimize, delay: A classic problem with new constraints," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2017.

[17] M. Inamullah, B. Raman, and N. Akhtar, "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*, ser. MSWiM '20. Association for Computing Machinery, pp. 181–189.

[18] H. Rohling and R. Gruneid, "Performance comparison of different multiple access schemes for the downlink of an ofdm communication system," in *1997 IEEE 47th Vehicular Technology Conference. Technology in Motion*, vol. 3, 1997, pp. 1365–1369 vol.3.

[19] M. Moeneclaey, M. Van Bladel, and H. Sari, "Sensitivity of multiple-access techniques to narrow-band interference," *IEEE Transactions on Communications*, vol. 49, no. 3, pp. 497–505, 2001.

[20] C. Tarhini and T. Chahed, "On capacity of ofdma-based ieee802. 16 wimax including adaptive modulation and coding (amc) and inter-cell interference," in *2007 15th IEEE Workshop on Local & Metropolitan Area Networks*. IEEE, 2007, pp. 139–144.

[21] S. Naribole, W. B. Lee, and A. Ranganath, "Impact of mu edca channel access on ieee 802.11 ax wlans," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–5.

[22] D. Magrin, S. Avallone, S. Roy, and M. Zorzi, "Validation of the ns-3 802.11 ax ofdma implementation," in *Proceedings of the Workshop on ns-3*, 2021, pp. 1–8.

[23] F. Tramarin, S. Vitturi, M. Luvisotto, and A. Zanella, "On the use of IEEE 802.11n for industrial communications," vol. 12, no. 5, pp. 1877–1886, conference Name: IEEE Transactions on Industrial Informatics.

[24] N. Marchenko, T. Andre, G. Brandner, W. Masood, and C. Bettstetter, "An experimental study of selective cooperative relaying in industrial wireless sensor networks," vol. 10, no. 3, pp. 1806–1816, conference Name: IEEE Transactions on Industrial Informatics.

[25] ns-3.34 released | ns-3. [Online]. Available: https://www.nsnam.org/news/2021/07/14/ns-3-34-released.html

[26] E. Perahia and R. Stacey, *Next Generation Wireless LANs: 802.11n and 802.11ac*, 2nd ed. Cambridge University Press.

[27] Research: Conviva's state of streaming q2 2021. [Online]. Available: https://www.conviva.com/state-of-streaming/

[28] Specifications for live video on facebook | facebook business help centre. [Online]. Available: https://en-gb.facebook.com/business/help/162540111070395?id=1123223941353904

[29] H. Chang, M. Varvello, F. Hao, and S. Mukherjee, "Can you see me now? a measurement study of zoom, webex, and meet," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21. Association for Computing Machinery, pp. 216–228.

[30] How much bandwidth does Skype need? | Skype Support. [Online]. Available: https://support.skype.com/en/faq/FA1417/how-much-bandwidth-does-skype-need

[31] R. Sinha, C. Papadopoulos, and J. Heidemann, "Internet packet size distributions: Some observations," USC/Information Sciences Institute, Tech. Rep. ISI-TR-2007-643, May 2007, orignally released October 2005 as web page http://netweb.usc.edu/%7ersinha/pkt-sizes/. [Online]. Available: http://www.isi.edu/%7ejohnh/PAPERS/Sinha07a.html

[32] 3GPP, "Study on Communication for Automation in Vertical Domains," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 22.804, 12 2017.

[33] A. Garcia-Rodriguez, D. López-Pérez, L. Galati-Giordano, and G. Geraci, "IEEE 802.11be: Wi-fi 7 strikes back," vol. 59, no. 4, pp. 102–108, conference Name: IEEE Communications Magazine.

[34] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach (6th Edition)*, 6th ed. Pearson.