A Deadline-Aware Scheduler for Smart Factory using WiFi 6

Mohit Jain
mohit20221@iiitd.ac.in
IIIT Delhi, India

Anis Mishra
anis20026@iiitd.ac.in
IIIT Delhi, India

Syamantak Das
syamantak@iiitd.ac.in
IIIT Delhi, India

Andreas Wiese
andreas.wiese@tum.de
TU Munich, Germany

Arani Bhattacharya
arani@iiitd.ac.in
IIIT Delhi, India

Mukulika Maity
mukulika@iiitd.ac.in
IIIT Delhi, India

ABSTRACT

A key strategy for making production in factories more efficient is to collect data about the functioning of machines, and dynamically adapt their working. Such smart factories have data packets with a mix of stringent and non-stringent deadlines with varying levels of importance that need to be delivered via a wireless network. However, the scheduling of packets in the wireless network is crucial to satisfy the deadlines. In this work, we propose a technique of utilizing IEEE 802.11ax, popularly known as WiFi 6, for such applications. IEEE 802.11ax has a few unique characteristics, such as specific configurations of dividing the channels into resource units (RU) for packet transmission and synchronized parallel transmissions. We model the problem of scheduling packets by assigning profit to each packet and then maximizing the sum of profits. We first show that this problem is strongly NP-Hard, and then propose an approximation algorithm with a 12-approximate algorithm. Our approximation algorithm uses a variant of local search to associate the right RU configuration to each packet and identify the duration of each parallel transmission. Finally, we extensively simulate different scenarios to show that our algorithm works better than other benchmarks.

CCS CONCEPTS

• **Networks** → **Packet scheduling**; **Wireless local area networks**; • **Applied computing** → *Computer-aided manufacturing*.

ACM Reference Format:
Mohit Jain, Anis Mishra, Syamantak Das, Andreas Wiese, Arani Bhattacharya, and Mukulika Maity. 2024. A Deadline-Aware Scheduler for Smart Factory using WiFi 6. In *International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '24), October 14–17, 2024, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641512.3686387

1 INTRODUCTION

The increasing demand for more efficient and more precise manufacturing has led to the demand for more intelligent or "smart factory" [17, 25]. With manufacturing processes getting more complex, wired connectivity is gradually getting replaced by wireless technologies to avoid clutter on the factory floor, reduce maintenance costs and enable easier planning [3, 17].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MobiHoc '24, October 14–17, 2024, Athens, Greece

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0521-2/24/10…$15.00
https://doi.org/10.1145/3641512.3686387

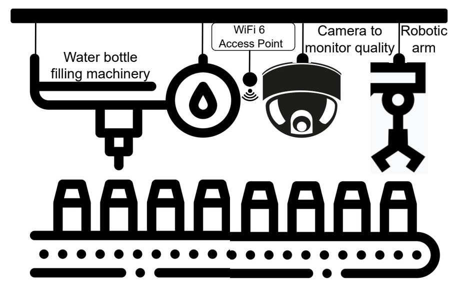


Figure 1: An illustrative example of a bottle-filling factory being controlled over WiFi. The bottle-filling equipment and the robotic arm have much stringent deadlines and higher profit in case of failure than the camera monitoring quality.

Background on Smart Factory: A smart factory is able to dynamically adapt the operation of machines in response to changes in the condition of machines, thus leading to more predictable and efficient production. Such a smart factory would collect data deployed from different sensors within the premises, and utilize them to schedule the operation rate of different machines. Thus, different machines are connected by a network, and a controller needs to decide how to schedule and operate them. A key challenge of such smart factory settings is that they handle a mix of critical and non-critical tasks. For example, Figure 1 shows a bottle-filling factory setting being controlled by WiFi. Here the *water bottle filling machinery* and *robotic arm* are very crucial to the operation of the factory. They have a very stringent deadline as well compared to the camera which is monitoring quality. Here, critical operations need to coexist with basic quality supervision and factory resource management. Missed deadlines of the most critical applications could potentially lead to the shutdown of factory operations or fatal accidents. On the other hand, such missed deadlines for non-critical applications might only lead to a temporary slowdown of production. Such settings require strong performance guarantees on latency from the wireless network that is used to connect the factory equipment [3]. Thus, wireless network technologies that are used need to ensure that they provide such performance guarantees.

A technique typically used for such applications is to first implement such protocols in the digital twin of a factory [34]. Once the digital twin shows acceptable performance, the wireless technology is deployed. This step of integrating the simulation with digital twin leads to a better understanding of bandwidth requirements, the number of packets that miss their latency, and consequences of such misses. Thus, simulation of algorithms used to schedule packets in a specific technology is crucial to understand its suitability for deployment.

WiFi for Smart Factory: Although WiFi is one of the most widely used wireless network technology, it was traditionally not used in factory settings because of its decentralized nature. This decentralized contention-based channel access scheme of WiFi led to the possibility of collisions and thus a lack of any real-time guarantees. Furthermore, WiFi was traditionally designed for a limited number of users/devices in a small area, thus limiting its scope in factory settings.

What makes WiFi 6 Suitable for Smart Factory? Compared to traditional WiFi, recent standards of WiFi such as IEEE 802.11ax, popularly known as WiFi 6, have made a number of changes that enable its use in factory settings. The first change is to enable its use 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. 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 [1]. This feature enables support for a smaller but more number of packets per unit time, which is commonly observed in factory settings [3]. In addition, with the OFDMA based transmission, WiFi enables centralized control where the access point (AP) schedules both downlink and uplink transmissions. Furthermore, WiFi 6 introduces more deterministic and centralized channel access scheme called a procedure called Multi-User Enhanced Distributed Channel Access (MU-EDCA). Using MU-EDCA, it is possible to suspend contention for channel access by setting the EDCA timer [28] for a specific duration. The stations switch their contention parameters to MU-EDCA parameters for a duration specified by the timer, thus moving to a centralized scheme. These changes enable WiFi 6 to be used in industrial settings [4, 33].

Although running smart factories over cellular networks is more widely discussed due to its centralized structure, utilizing WiFi 6 in smart factories has other advantages over cellular networks: (1) WiFi 6 APs are already commercially available and are cost-effective, making it easy to use in any factory. In contrast, using 5G requires acquiring of license from the government or approaching a telecom operator which provides specific ultra reliable low latency communication (URLLC) service. Such services today are only available via commercial agreements and are not easily accessible to small factories [24]. (2) Practical evaluations using 5G and WiFi suggested that both can provide latencies required for smart factory [32], with WiFi providing occasionally lower latency than 5G.

Challenges: In this paper, we utilize the paradigm shift towards centralized channel access supported by 802.11ax to design a scheduling framework for deadline-based flexible factory settings that involve uplink traffic. Although 802.11ax enables support for such factory settings, the actual scheduling of packets by the AP is still crucial to ensure that the factory is both safe and efficient. Such scheduling must prioritize the delivery of the most critical packets within the deadline, while ensuring that the non-critical ones are not ignored. The protocol itself also has its own characteristics not found in other technologies, such as 5G. For example, (1) the standard provides a specific way of splitting the bandwidth into RUs, (2) one RU can be allocated to at most one user and vice-versa, and (3) the starting and ending time of the transmissions must be synchronized. These characteristics of 802.11ax make the scheduling of packets a challenge, as they do not allow us to utilize any algorithm proposed by prior works in scheduling (details in §2).

Our Approach: Our solution utilizes known packet arrival patterns and deadline requirements for the applications. Further it defines a 'profit' for each job that indicates the relative criticality with more critical packets having higher profit. For example, in case of the bottle filling factory (shown in Fig. 1), *water bottle filling machinery* and *robotic arm* are the critical applications and thus have higher profit compared to *Camera to monitor quality* application. Such presence of critical applications co-existing with less critical ones is frequently seen in modern industrial systems [16]. The algorithm schedules packet transmissions provided by the access point via a trigger frame. The objective of the scheduler is to maximize the profit of packets that finish transmission on or before the respective deadline (§ 3.1).

We reduce the above problem to a novel variant of classical parallel machine scheduling which we call *Deadline-aware Parallel Machines Scheduling with Synchronized start (DPMSS)*. The two crucial constraints of scheduling jobs in 'batches' and bandwidth restrictions on the RUs to be used pose new challenges over existing techniques for deadline-aware scheduling. We argue that our problem is strongly NP-hard even in the simplest settings and design a $(12+\varepsilon)$-approximation algorithm, namely *Local Search Deadline Scheduling (LSDS)*. At a high level, our technical contribution involves developing an 'interval scheduling viewpoint' of the DPMSS problem (see § 3.2). We show that any solution to the DPMSS problem can be thought of as (1) selecting disjoint sub-intervals of a scheduling time-horizon, (2) for each selected sub-interal, determining an assignment of packets to the RUs under certain bandwidth restrictions such that all assigned packets start transmission at beginning and finish on or before the end of the sub-interval. We develop a *local search* based algorithm (LSDS) for (1) in conjunction with a *budgeted version of the classical maximum weight bipartite matching* for (2) (see § 4). Below, we summarize our main contributions:

(1) We utilize WiFi 6 for efficient operation of a smart factory. We model such an operation through a novel variant of classical scheduling problem with release times and deadlines on heterogeneous machines, which we call DPMSS. The scheduler aims to maximize total profit of packets successfully transmitted, where profits indicate relative criticality of packets.

(2) We show that DPMSS is *strongly NP-hard* and use an *admission control* strategy based on *iterative search* to get the *first* approximation algorithm (LSDS) with an approximation ratio $(12+\varepsilon)$, for any given positive ε. In fact, for a special case when the RU configurations for each transmission are given and fixed, our algorithm (namely LSDSF) is a 12-approximation.

(3) We create a simulation framework of factory IoT settings. We reproduce 4 different use cases under practical channel conditions. We evaluate the performance of LSDS, LSDSF and compare with three other benchmarks. We observe that LSDS outperforms all the benchmarks in terms of profit achieved and packet drops (including critical packets). Further, we also observe that LSDS succeeds in determining the schedules in real time.

An extended version of this paper with complete proofs is available at [23]. We have also made the source code available at [22] to reproduce all our experiments.

2 RELATED WORK & MOTIVATION

2.1 Related Work

WiFi in Industrial settings: Multiple works have attempted to overcome the problem of non-deterministic access and unreliable wireless links seen in WiFi by modifying the WiFi protocol and/or better scheduling of packets. For example, Seno et al. [33] attempted to enhance determinism in WiFi communications by introducing a coordinator node to enable soft real time applications in industries. Wi-Red [12] proposes a parallel redundancy protocol which aims to improve timeliness and dependability in links by transmitting duplicate packets on redundant links. These works are orthogonal to our approach of utilizing OFDMA for deadline constrained settings. Similarly, a time division multiplexing-based modification of the WiFi protocol, known as RT-WiFi, was proposed in [36]. None of these techniques utilize OFDMA for such purposes. Although the work [7] performs the job of resource allocation using OFDMA in WiFi 6, it uses a heuristic without providing any performance guarantees.

Optimal sub-carrier allocation: Scheduling and resource allocation in IEEE 802.11ax has gained a lot of attention in recent times [20, 21, 35], where the authors formulate this as an optimization problem and propose various solutions for the same. Works such as [20, 35] propose various algorithms to solve the problem of maximizing the weighted sum rate. Unlike our paper, none of these studies consider the deadlines of each user. The closest work to ours is by Inam et al. [21], where they designed a scheduler to minimize the number of packet drops in a deadline-driven environment. However, lack of theoretical guarantees makes their heuristic less suitable for safety-critical settings.

Scheduling with Deadlines: Scheduling jobs with deadlines to maximize profit (more commonly called weighted throughput in literature) has been widely studied under a variety of settings. The general setting is consists of a set of jobs, each with a release time, a deadline and a profit. The task is to schedule these jobs on a set of parallel heterogeneous machines to maximize profit of jobs that finish on or before their deadlines. The problem had been established to be strongly NP-hard even in the basic setting of a single machine and unit profit for all jobs [18]. The work of Bar-Noy et al. [9] gives the first constant factor approximation algorithms for the problem on multiple machines and arbitrary profits, which was later improved to 2-approximation independently by Bar-Noy et al. [8] and Berman et al. [11]. For the case with uniform profits, Chuzhoy et al. [14] improves it further to 1.582. Throughput maximization on unrelated machines with machine dependent release dates and deadlines has also been studied in the context of wireless sensor networks [5].

Note that 5G in cellular networks also uses OFDMA, and deadline scheduling in the context of 5G's ultra-reliable low latency communication (URLLC) solves a similar problem. However, the resource blocks/units can be independently used in 5G [6, 30], without synchronizing their start times of transmission for each user, unlike in WiFi 6 [1]. Requiring such synchronized transmission effectively requires a new class of scheduling algorithms [28]. Unlike 5G, which has access to servers, WiFi's scheduling happens entirely on access points with small amounts of compute power. This requires algorithms with low computational overhead [31]. Finally, WiFi 6 provides a specific way of splitting the channel into smaller RUs and maximum one RU can be allocated to a node, making it impossible to directly use the algorithms designed for scheduling in URLLC.

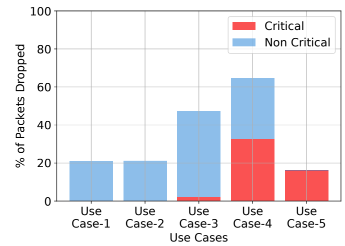


Figure 2: Percentage of total and critical packets dropped in a total of four different use cases using Earliest Deadline First (EDF). Use Case 1 does not have any critical packets.

To the best of our knowledge, the exact scheduling problem DPMSS has not been studied so far. In particular, two major components in our problem makes the existing algorithms difficult to be used - (1) In DPMSS, the processing of jobs needs to happen in batches in a synchronized fashion on each machine and (2) There is a bandwidth restriction on the machines processing the jobs in a particular batch. We carefully design a novel algorithm that can handle both these constraints effectively with only a constant loss in the approximation factor.

2.2 Requirement of a Scheduling Algorithm

We now motivate the requirement of new algorithms to schedule WiFi packets with deadline by demonstrating that naive ideas lead to a significant drop in critical packets in an industrial setting. We consider a single WiFi AP and a total of four different use cases, with the total number of devices (apart from the AP) ranging from 40-90, drawing from scenarios mentioned in the research literature (further details in §5.1). The devices send data to collect control information or monitor the working (additional details in §4). We use the earliest deadline first (EDF) technique as proposed in prior work [21], and observe the number of packets dropped (Figure 2). In the last three use cases, the packets from some users are more critical than the other packets. In these cases, we also plot the percentage of the most critical packets dropped. We observe that the number of packets dropped exceeds 10% in all cases. Furthermore, for Use Case-4, we find that all the packets dropped by EDF are critical. This indicates the need for a more intelligent scheduler. Furthermore, a significant number of total packets are dropped in each case. This is because EDF only prioritizes the earliest deadline first without considering how much parallelism can be used. Thus satisfying the deadline of one packet can lead to a cascade of other packets missing their deadlines, as there is a requirement of synchronized transmission.

Thus, designing an intelligent scheduler is non-trivial, as it needs to take into account the overall latency requirements of all the packets. The scheduler must also run in real-time, i.e, the total execution time should not exceed the time-window available for transmission.

3 PROBLEM DEFINITION

In this section, we define a new variant of classical parallel machine scheduling problem which we call Deadline-aware Parallel Machines Scheduling with Synchronized start (DPMSS) and show that our WiFi packet scheduling problem reduces to the above.

3.1 Problem Model

We have a single WiFi 6 Access Point that allows parallel OFDMA transmissions. As mentioned earlier, WiFi 6 allows synchronized parallel transmissions, where multiple users can transmit in parallel but the transmissions must begin together. Furthermore, each such transmission must end within a fixed duration of TXOP = δ.

We also have a fixed set of users, where each user generates packets periodically. We denote the set of packets (*jobs*) as J. A job $j \in J$ has a specific generation time (*release time*) r_j and a *deadline* d_j before which the packet must be delivered. The relative importance of delivery packet j is modelled by assigning non-negative *profits* $w_j, \forall j \in J$.

The RU's are modeled as a set of *machines* M that can *process* jobs J. We denote the time taken using RU (machine) $i \in M$ to deliver (process) a packet (job) $j \in J$ by p_{ij}. Further, each machine occupies a *bandwidth* $b_i, \forall i \in M$, during the entire time for which it is active and B denotes the total available bandwidth

Definition 1. *DPMSS: The DPMSS problem requires to schedule jobs non-preemptively in batches such that*

(1) Each batch consists of at most $|M|$ jobs which are to be processed non-preemptively on individual machine (one can think of a batch as one parallel transmission of packets)
(2) For each batch, all jobs need to start processing at exactly the same time (hence synchronized start). The endtime of a batch is defined by the latest time for a job in the batch to finish processing
(3) The total bandwidth requirement of machines active within a batch must not exceed B.

The objective is to maximize the total profit of jobs that finish at or before their respective deadlines. One possible approach of solving DPMSS is to greedily group jobs according to deadlines/profits and assign each group to suitable machines. However, such myopic algorithms have arbitrarily bad approximation ratio. In § 3.2, we develop an alternate viewpoint of DPMSS based on scheduling the parallel transmission intervals. This allows us to use a combination of admission control based iterative search.

3.2 An Interval Scheduling Viewpoint

We now formalize the interval scheduling viewpoint of DPMSS defined in § 3.1. Let $\mathcal{T}$ be a time-horizon which is pre-defined such that the scheduling needs to happen in the interval $[0, \mathcal{T}]$ (we assume $\mathcal{T}$ is an integer). An interval $\mathcal{I}(t_1, t_2)$ is a subset of $[0, \mathcal{T}]$, where both t_1, t_2 are non-negative integers and $t_1 < t_2$. An interval of unit length of the form $[t, t+1], t \in \mathbb{N}$ is called a *time-slot*. We assume that all release dates and deadlines are non-negative integers inside the time-horizon $[0, \mathcal{T}]$.

Definition 2. *Admissible jobs to a machine: A job j is* admissible *to machine $i \in M$ in interval $\mathcal{I}(t_1, t_2)$ if $r_j \leq t_1$ and $t_1 + p_{ij} \leq \min\{t_2, d_j\}$.*

The first condition comes from the fact that the job can only start after it is released. The second condition follows from the fact that if a job is processed by a machine in an interval, then it must complete by the end-time of the interval or it's deadline, whichever is smaller.

Definition 3. *Admissible jobs to an interval: A job j is admissible to an interval $\mathcal{I}$ if there exists a machine $i \in M$ such that j is admissible to i in interval $\mathcal{I}$.*

Intuitively, one can think of a solution to DPMSS as a collection of intervals within which each batch of parallel transmission happens.

Observation 4. *Any feasible solution to DPMSS consists of*

(1) Disjoint intervals of $[0, \mathcal{T}]$, $\mathcal{I}_1, \mathcal{I}_2, \cdots \mathcal{I}_k$, and
(2) for each interval $I_r, r = 1, 2, \cdots k$, a set of admissible jobs $J(\mathcal{I}_r)$ and a set of active machines whose total bandwidth does not exceed B, along with an assignment of the jobs to active machines inside $\mathcal{I}_r$.

such that for any interval $\mathcal{I}_r$, no machine is assigned more than one job in the set $J(\mathcal{I}_r)$ and no job belongs to both $J(\mathcal{I}_r), J(\mathcal{I}_{r'}), r \neq r'$.

Hence, we need to design an algorithm that determines (1) and (2) such that the total profit of jobs scheduled is maximized. We shall adopt this point-of-view of DPMSS in the following discussion.

Theorem 5. *The DPMSS problem is strongly NP-hard*

Proof. We consider the simplest setting of DPMSS where we have one machine of unit bandwidth and the total bandwidth available $B = 1$. Then the DPMSS problem reduces to the classical decision problem of single machine non-preemptive scheduling with release dates and deadlines, which decides whether all jobs can be scheduled within their respective deadlines or not. This problem has been proved to be strongly NP-hard in [18]. Hence DPMSS is strongly NP-hard. □

We next establish that even a sub-problem to our main problem is computationally hard.

Theorem 6. *For a fixed interval $\mathcal{I}$ and a given bandwidth B, determining the maximum profit subset of admissible jobs and their assignment to machines such that total bandwidth of machines involved in the assignment does not exceed B is weakly NP-hard.*

The above theorem follows via a reduction of the Knapsack Problem to this problem. On the positive side, the above problem admits a PTAS [10].

4 ALGORITHM FOR DPMSS

Given the above hardness results, we give approximation algorithms for DPMSS. We first consider an easier setting where the set of machines (RUs) to be used for processing is the same fixed one for each batch - we call this DPMSS with *fixed* configuration (DPMSSF). In other words, we assume that we are given a valid configuration of machines that satisfies constraint (3) in Definition 1. Note that Theorem 5 directly implies that even this simpler setting is strongly NP-hard. In §. 4.3, we show how to extend this algorithm to the general setting where the algorithm must decide which machines to engage for each batch under bandwidth constraint.

4.1 Algorithms for DPMSSF

We need a few more notations to describe our algorithm. For any subset of jobs $J^{Alg} \subseteq J$, let $w(J^{Alg}) = \sum_{j \in J^{Alg}} w(j)$ denote the total profit of all jobs in J^{Alg}. Let δ be the maximum length of any interval in any feasible solution. In general, $1 \leq \delta \leq \mathcal{T}$, for our applications, δ (duration of one transmission) is a small constant.

Definition 7. *Conflicts : An interval $\mathcal{I}_1 = [t_1, t_2]$ is defined to be in conflict with an interval $\mathcal{I}_2 = [t_3, t_4]$ if either $t_1 \leq t_3 \leq t_2$ or $t_3 \leq t_1 \leq t_4$.*

Algorithm 1 Algorithm LSDSF $(J,M,\mathcal{T},\delta)$

1: $S^{Alg}=\{\},J^{Alg}=\{\}$
2: **for** $\ell=1,2,\cdots\delta$ **do**
3:   **for** $t=0,1,2,\cdots\mathcal{T}-\ell$ **do**
4:     Let $\mathcal{I}$: Interval $[t,t+\ell]$
5:     $(\mathcal{M},J(\mathcal{I}))\leftarrow$ Max-Profit $(J\setminus J^{Alg},\mathcal{I},M)$
6:     Let $\mathcal{S}'=\{\mathcal{I}'\in S^{Alg}:\mathcal{I}'$ conflicts with $\mathcal{I}\}$
7:     Let $J'=\bigcup_{\mathcal{I}'\in\mathcal{S}'}J(\mathcal{I}')$
8:     **if** $w(J(\mathcal{I}))>2w(J')$ **then**
9:       $J^{Alg}\leftarrow J^{Alg}\setminus J'\cup J(\mathcal{I})$
10:       $S^{Alg}\leftarrow S^{Alg}\setminus\mathcal{S}'\cup\{\mathcal{I}\}$
11:       Assign job j to machine i inside interval $\mathcal{I}$ if and only if (ji) is in matching $\mathcal{M}$
12:     **end if**
13:   **end for**
14: **end for**

Algorithm 2 Algorithm for Max-Profit $(J',\mathcal{I},M)$

1: $J(\mathcal{I},i)=\{j\in J':j$ is admissible to i in $\mathcal{I}\}$
2: $J'=\bigcup_{i\in M}J(\mathcal{I},i)$
3: Construct bipartite graph G where
(a) J' and M are the two partite sets of vertices
(b) For $j\in J'$ and $i\in M$ add edge (ji) with weight w_j if and only if $j\in J(\mathcal{I},i)$
4: Solve Maximum Weight Bipartite Matching on G - let $\mathcal{M}$ be the matching and $\tilde{J}$ be the set of jobs matched
5: Return $\mathcal{M},\tilde{J}$

We show our main algorithm named as LSDSF (Local Search Deadline Scheduling Fixed split) in Algorithm 1. The main idea is the following. We start with an empty set of intervals S^{Alg} and empty set of jobs J^{Alg} – both of which would eventually form our solution. The outer loop of Line 3 iterates over all possible integral lengths of the intervals 1 to δ. For a fixed length ℓ, Line 4 of the algorithm iterates over all possible intervals of length ℓ. While considering a particular interval, say $\mathcal{I}=[t,t+\ell]$, the algorithm first determines the admissible set of jobs in $\mathcal{I}$ with *maximum possible profit* that have not yet been added to J^{Alg} – let $J(\mathcal{I})$ denote this set of jobs. Solving this sub-problem itself is non-trivial since there are exponentially many subsets to be considered. However, it reduces to an instance of the classical *maximum weight bipartite matching problem.* We invoke standard algorithms to determine the maximum profit subset of jobs $J(\mathcal{I})$ and also the assignment of each job $j\in J(\mathcal{I})$ to individual machines $i\in M$ such that j is admissible to i in interval $\mathcal{I}$ (Line 5) . This sub-procedure is listed in Algorithm 2. Lines 6-11 form the heart of our algorithm. We need to decide whether to add the new interval $\mathcal{I}$ to $\mathcal{S}$. To this end, we check whether $w(J(\mathcal{I}))$ is strictly bigger than twice the total profit of already selected jobs in intervals that are in conflict with $\mathcal{I}$ and are currently in $\mathcal{S}$ (Line 8). If this is true, then we discard all the conflicting intervals from $\mathcal{S}$, remove all jobs assigned to such intervals from J^{Alg} while adding new interval $\mathcal{I}$ to $\mathcal{S}$ and a new set of jobs $J(\mathcal{I})$ (Lines 9-10). Otherwise, we ignore the interval $\mathcal{I}$.

4.2 Analysis

Runtime. LSDSF shown in Algorithm 1 runs in time $O(\mathcal{T}\cdot\delta\cdot\rho(|J|,|M|)$ where, $\rho(|J|,|M|)$ is the runtime of any algorithm for maximum weight bipartite matching. The currently best known theoretical bound for ρ is nearly linear in $|J|\cdot|M|$ [13]. However, one can use a more practical algorithm running in similar time with only an $(1+\varepsilon)$-factor loss in the approximation ratio (where ε is a tiny positive quantity) [15].

Approximation Ratio. We now prove the following theorem in this section. We defer the proof of some of the lemmas to the extended version of the paper due to lack of space [23].

Theorem 8. *LSDSF is a 12-approximation for the DPMSSF problem.*

Lemma 9. *LSDSF always computes a feasible solution to a given DPMSSF instance.*

The more non-trivial part is to prove the approximation ratio. First, we need a lemma about the Max-Profit subroutine.

Lemma 10. *Given an interval $\mathcal{I}$, a subset of jobs J' and a set of machines M, the algorithm Max-Profit returns the subset of admissible jobs of J' in $\mathcal{I}$ with maximum possible profit*

Lemma 11. $w({J^\star}_1)\leq 5((w({J^\star}_2)+w({J^\star}_3)+w(\tilde{J}))$

We do so by proving a more general statement that will imply the above lemma as a corollary.

Lemma 12. *Let $\tilde{\mathcal{S}}$ be the set of all intervals that were added but later discarded by Algorithm 1. Then $w(J^{Alg})\geq w(\bigcup_{\mathcal{I}\in\tilde{\mathcal{S}}}J(\mathcal{I}))$*

The proof of the above lemma requires a delicate 'charging' argument which we are going to develop over a few steps. We first define a collection of tree structures that models the behavior of the algorithm and allows us to bound the profit of intervals we discard.

Definition 13. *Charging Forest: We define a node for each interval in S^{Alg} and $\tilde{\mathcal{S}}$. We make an interval node $\mathcal{I}$ the parent of another node $\mathcal{I}'$ if and only if $\mathcal{I}'$ was discarded by the algorithm from J^{Alg} due to conflict with $\mathcal{I}$.*

Observation 14. *The roots of the charging forest are precisely the intervals in J^{Alg} upon termination, while the internal nodes and the leaves form the set S^{Alg}.*

The following lemma forms the heart of our analysis,

Lemma 15. *For any internal node of the charging forest, $\mathcal{I}$, the total profit of the jobs corresponding to the sub-tree rooted at $\mathcal{I}$ excluding the node I is at most the profit of $\mathcal{I}$.*

Proof. We define levels of nodes in the charging forest $\mathcal{F}$ in the natural way. The leaf nodes are level 0. An internal node is of level $j+1$, where j is the maximum level of any of it's children. We prove the lemma by induction on the level. The hypothesis is clearly true for level 0. Now let it be true for all level $0,1,2,\cdots j$ and consider a node $\mathcal{I}$ at level $j+1$. Let $\mathcal{I}'_1,\mathcal{I}'_2\cdots\mathcal{I}'_p$ be it's children. By construction of the charging forest and Line 8 of the algorithm,

$$w(J(\mathcal{I}))>2\sum_{i=1}^{p}w(J(\mathcal{I}'_i))$$

Further, applying induction hypothesis, the total profit of jobs corresponding to the sub-trees rooted at the nodes $\mathcal{I}'_1,\mathcal{I}'_2,\cdots\mathcal{I}'_p$ excluding the nodes themselves is at most $\sum_{i=1}^{p}w(J(\mathcal{I}'_i)$. Hence, adding the total profit of all jobs in the subtree rooted at $\mathcal{I}$ excluding itself is at most $2\cdot w(J(\mathcal{I}))/2=w(J(\mathcal{I}))$ ◻

Table 1: Simulation Parameters

Parameter	Value	Parameter	Value
Stations	use-case specific	MCS	11
Transport Protocol	UDP	Bandwidth	40 MHz
Frame Aggregation	Disabled	Guard Interval	3200 ns

Hence, using Lemma 15 on the roots of the charging forest and Observation 14, we have Lemma 12. As a corollary of Lemma 12, we get the following lemma:

LEMMA 16. $w(J^{\star}{}_2) \leq w(J^{Alg})$

Now we finish the proof of Theorem 8. Using Lemmas 11, 16,

$$\begin{aligned} w(J^{\star}) &= w(J^{\star}{}_1) + w(J^{\star}{}_2) + w(J^{\star}{}_3) \\ &\leq 5(w(J^{\star}{}_2) + w(J^{\star}{}_3) + w(\tilde{J})) + w(J^{\star}{}_2) + w(J^{\star}{}_3) \\ &\leq 10w(J^{Alg}) + 2(J^{Alg}) = 12w(J^{Alg}) \end{aligned}$$

4.3 Extension to DPMSS.

We briefly discuss how to extend LSDSF to DPMSS. The only difference lies in the sub-routine MAX-PROFIT . In case of DPMSSF, this subproblem could be solved using maximum weight bipartite matching over the set of machines and admissible jobs. In case of DPMSS, we need to solve a budgeted version of this problem since the scheduler has the additional task of allocating the total bandwidth B among machines. Formally, we again construct the bipartite graph G exactly as in Algorithm 2. However, we need to ensure that the total bandwidth of nodes selected by the matching $\mathcal{M}$ from the set M does not exceed B. As stated in Theorem 6, this problem itself is weakly NP-hard (unlike the unbudgeted case) and there is a PTAS [10] known for this problem. Plugging in the latter gives us the following theorem in a straightforward fashion.

THEOREM 17. *There is a* $(12+\varepsilon)$*-approximation algorithm (LSDS) for DPMSS, for any fixed* $\varepsilon > 0$.

Runtime. The runtime of LSDS with the above modification is $O(\mathcal{T} \cdot \delta \cdot \rho'(|J|,|M|)$ Here ρ' is the runtime of the budgeted maximum weighted bipartite matching algorithm used. Using the state-of-the-art [10], this can be implemented in time $(|J| \cdot |M|)^{O(1/\varepsilon)}$.

5 EVALUATION SETUP

In this section, we describe our evaluation setup, evaluation test cases, metrics, and the benchmarks.

Simulation Paramters: The parameters for our simulation are provided in Table 1. Note that we simulated for 200 ms because the periodicity of transmissions allows the WiFi access point to reuse the same schedule repeatedly. While this simulation horizon ensures packets cannot be scheduled across rounds of 200 ms, a deadline above 200 ms is considered easily achievable over WiFi and thus this does not hurt our performance. We primarily focus on settings where deadlines are ≤ 200 ms, as these are most challenging for WiFi to handle. Note that this also implies that in case of any changes in the number of nodes, a new schedule can be generated and utilized with a delay of 200 ms. This delay is considered relatively small, since WiFi nodes often take up to 1 s [29] to establish a connection.

Additionally, our simulation disables frame aggregation 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 evaluate under both good and poor channel conditions. We also assume that the trigger frame, carrying the station-to-RU mapping, along with the Uplink OFDMA data frame is able to finish transmission within TXOP.

Implementation: We have implemented the code using C++, and compile it using gcc compiler with the optimization flag "O2". In §.4.3, we had implemented the MAX-PROFIT subroutine using an existing PTAS for maximum weight budgeted matching [10] which leads to the desired approximation ratio. However, PTAS-es are prohibitive in practice. In the actual implementation, we use a much faster iterative heuristic which cleverly navigates through the search space of all possible RU configurations. The algorithm starts with an infeasible configuration containing the maximum number of RU's of each type (essentially the set of all machines M), ignoring the bandwidth restriction and greedily determining the best possible assignment of M jobs to the machines. This step prunes the possible set of jobs to size $|M|$. The next idea is to iterate over all possible configurations and only these pruned set of jobs and output the best combination. The algorithm can be implemented in time $O(\mu \cdot |M| \log |M|)$, where μ is the total number of possible feasible RU configurations. Since in practice, both M and μ are quite small (at most 36 for our use cases), it makes our algorithm much faster in practice compared to the theoretical bounds. Except for the cases where we separately mention, this is typically equal to 40MHz.

Evaluation Metrics: The metrics of interest for us are (1) profit ratio which is defined as the ratio between the profits obtained vs total profit that could be achieved if all the packets got scheduled in due time, (2) drop percentage which is defined as the percentage of total packets dropped, and (3) critical drop percentage which is defined as the percentage of critical packets dropped. For each use case, the criticality is decided by the application with the highest profit, and (4) runtime which is defined as the time needed to execute the algorithm.

5.1 Factory Use Cases

We now describe various simulated use cases, each of which represents different real-world scenarios. Some of these use cases generate packets with sizes following certain distributions, we repeat the simulation 100 times and plot the 95% confidence interval. We assign the profit of each application based on its criticality, i.e. applications with higher reliability requirements get higher profits, as in [30].

Use Case-1 (UC-1): We consider a factory scenario with a set of sensors and controller [19]. The network simulates real-time communication between the sensors and the controller. The communication between the sensor and the controller is based on the application profiles presented in Table. 2. Each profile includes a periodic transmission rate, a minimum, and maximum frame size, and an end-to-end latency requirement for frames. We create a topology with 50 sensors and 1 controller as AP. For each device, we set the deadline as the maximum latency corresponding to its profile. We retain the same profit for all nodes, as no specific values are reported.

Use Case-2 (UC-2): Here, we consider an industrial IoT system with different application settings [26]. Table. 3 shows the application characteristics. The control traffic has the maximum profit.

Use Case-3 (UC-3): As in [27], we consider another industrial IoT (IIoT) network. Table. 4 represents various applications in this network and their characteristics. Here, the arrival of packets is modeled

Table 2: Use Case-1 (UC-1) Application Characteristics [19]

Appl.	Gen. rate (pkts/s)	Size (B)	Deadline (ms)	Profit	#Nodes
Profile 1	4000	$U(64, 128)$	0.25	10	10
Profile 2	2000	$U(128, 256)$	0.5	10	10
Profile 3	1000	$U(256, 512)$	1	10	10
Profile 4	500	$U(512, 1024)$	2	10	10
Profile 5	250	$U(1024,1522)$	4	10	10

Table 3: Use Case-2 (UC-2) Application Characteristics [26]

Appl.	Gen. rate (pkts/sec)	Size (B)	Dead-line (ms)	Profit	# Nodes
Smart meters	1.25	100	16	10	15
Status info	2.5	100	16	20	15
Reporting & logging	0.75	500	1000	30	15
Data polling	1	500	16	10	15
Control traffic	937.5	100	16	160	20
Video surveil-lance	2000	1500	1000	10	10

Table 4: Use Case-3 (UC-3) Application Characteristics [27]

Appl.	Gen. rate (pkts/s)	Size (B)	Deadline (ms)	Profit	# Nodes
Motion control	40000	50	1	30	10
Collaborative AGV	40000	50	4	20	10
Robotic control	40000	50	10	30	10
Asset/process monitoring	40000	50	100	5	10

as a Poisson process with an average arrival rate λ packets/sec as mentioned in 2$^{\text{nd}}$ column of Table. 4. We assign motion-control and robotic-control applications the highest profit followed by collaborative AGV (Automated Guided Vehicle). The asset/process monitoring has the least profit.

Use Case-4 (UC-4): As in [3], we now consider a metal processing site. It has a set of machines and tools managed in a certain area. Table 5 represents the set of applications for the metal processing site, with a total of 10 applications running on 59 stations.

Benchmarks. We consider three benchmarks namely, Earliest Deadline First (EDF), Largest Profit-To-Deadline Ratio First (LRF), and a Non-Starving variant of Largest Profit-To-Deadline Ratio First (NLRF). All three benchmarks sort the stations based on a scheduling metric and assign the first M stations to M RUs. It exhaustively searches over all RU configurations and picks the one that provides the maximum profit. The scheduling metric (sm) used for EDF and LRF are the deadline (d_j) and profit-to-deadline ratio (p_j/d_j) respectively, whereas for NLRF it is given by:

$$sm_{nlrf} = \frac{p_j/d_j}{(T_{packets}+1)/(G_{packets}+1)}, \tag{1}$$

where $T_{packets}$ and $G_{packets}$ represent the number of transmitted and generated packets respectively. This is to ensure that the profit-to-deadline ratio eventually increases for applications that are continually being starved. For this, it balances out the ratio by the total number of packets transmitted to the total number of packets generated till the current time. Note that here we do not only consider the total number of packets transmitted, instead we also consider

Table 5: Use Case-4 (UC-4) Application Characteristics [3]

Appl.	Gen. rate (pkts/s)	Size (B)	Deadline (ms)	Profit	# Nodes
Size inspection by line camera (line sensor)	1	30000	5000	30	3
Detect defect state	10	500	500	50	4
Sensing for managing AC	0.016	64	6000	45	1
Preventive maintenance	0.02	30	1000	50	2
Monitoring of equipment	1	20	1000	30	2
Counting number of wrench operations	0.016	64	100	40	10
Movement analysis wireless beacon	2	4000	4000	10	6
Racking assets (beacon transmission) information of equipment and things	1	200	1000	15	20
Tracking parts, stock RFID tag	0.028	1000	1000	45	10
Techniques, knowhow from experts video, torque waveforms	50	24000	200000	1	1

number of packets generated as they are not uniform across the stations. Hence, over long run the NRLF scheduler will start prioritizing packets with lower profits over those with higher profits.

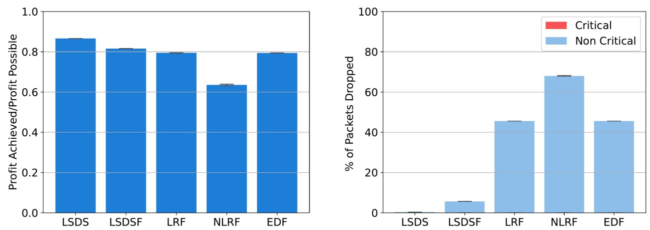
6 EVALUATION RESULTS

We compute all the metrics for both versions of our algorithm i.e., LSDSF and LSDS. We then compare it with other benchmarks i.e., EDF, LRF, and NLRF for all 4 use cases and plot the results in Figures 3- 6. Here, we observe that both LSDSF and LSDS outperform all benchmarks for all 3 metrics across all use cases. The primary reason for the improved performance can be attributed to the interval scheduling viewpoint which allows us to de-couple the task of selecting the correct set of transmission intervals and the correct combination of packets and RUs within each interval. Furthermore, our algorithm avoids myopic decisions unlike local greedy approaches adopted by the benchmarks. For instance, a particular job might be added and discarded by our algorithm several times till termination. On the other hand, the benchmarks naively sort the stations based on some intuitive scheduling metric and groups packets greedily. This is also why the benchmarks are much faster, but highly sub-optimal. Essentially, they fail to find the combinations which might be sub-optimal 'locally' but earns a lot of benefit from a global point of view. Further, LSDS outperforms LSDSF as the latter only uses a fixed RU split, while the former picks the optimal RU config.

6.1 Evaluation Results for Different Use Cases

We now discuss each use case-specific result.

Use Case-1: Figure 3 shows the profit-ratio and drop percentage for UC-1. In this case, we have about 8K packets/sec, hence even LSDS



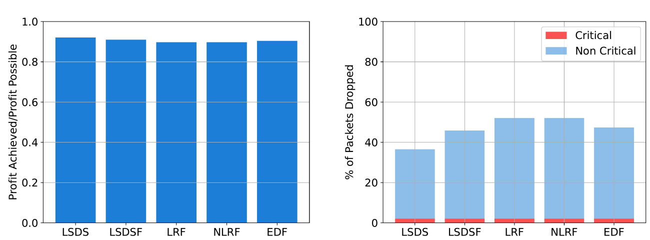
(a) (b)

Figure 3: Use Case-1 (UC-1): LSDS, LSDSF, EDF, LRF, and NLRF; (a) Ratio of profit obtained to total profit, (b) Percentage of total and critical packets dropped.

was not able to schedule all packets within their deadline. Note that there are no critical packets as the use case does not provide any application with higher profit. However, it outperforms all the baselines. Here, EDF and LRF also performed well and were able to obtain a profit ratio close to 0.8. Note that in this case, all the applications have the same profit. Hence, the scheduling metric turns out to be the same for both EDF and LRF for all applications and they behave exactly the same. However, NLRF suffers. As observed from Table. 3 the applications i.e., Profiles 1 and 2 with higher packet generation rates also have lower delay tolerance. This causes applications with relatively larger delay tolerance to starve and hence NLRF tries to schedule these applications. While scheduling packets of Profiles 3-5, NLRF misses out on Profiles 1 and 2, as they have a high generation rate with the same profit. Hence, overall NLRF achieves a lower profit compared to LRF and incurs a higher loss ratio as well.

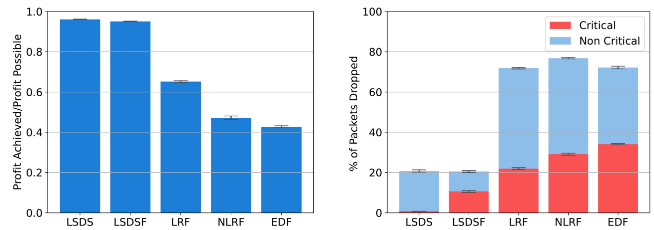
Use Case-2: Figure 4 shows the ratio for UC-2. There are a few interesting observations. The ratio for LSDS is a little less than 1. Here, the number of packets for the most critical application i.e., the control traffic is high. It generates almost 1 packet every ms and they have a deadline of 16ms. However, there are 20 nodes that run these control traffic application. Hence, LSDS is not able to schedule them all and had to drop some critical packets too ($\simeq 2\%$). Further, video surveillance also has a very high generation rate leading to high non-critical packet drops. Here, EDF prioritizes all 4 applications with shorter deadlines i.e., smart-meter, status information, data polling, and control traffic. However, LRF correctly prioritizes the most critical application i.e., control traffic. The performance of all the algorithms is very similar as other applications like smart-meter, status information, and data polling have a very lower packet generation rate i.e., $1-2$ packets/sec. Thus, all the algorithms end up allocating resources to control and video surveillance applications only.

Use Case-3: Figure 5 shows the ratio for UC-3. The total capacity requirement for this use case is $16{,}00000 \times 50 \times 8 = 640 Mbps$. A bandwidth of 40 MHz cannot handle this much load (as the maximum



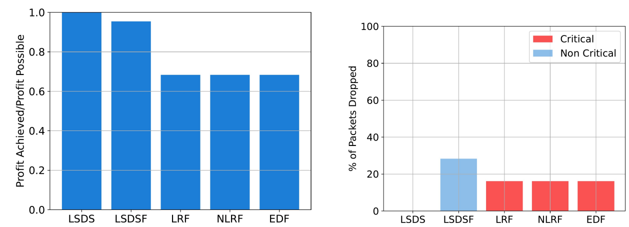
(a) (b)

Figure 4: Use Case-2 (UC-2): LSDS, LSDSF, EDF, LRF, and NLRF; (a) Ratio of profit obtained to total profit, (b) Percentage of total and critical packets dropped.



(a) (b)

Figure 5: Use Case-3 (UC-3): LSDS, LSDSF, EDF, LRF, and NLRF; (a) Ratio of profit obtained to total profit, (b) Percentage of total and critical packets dropped.



(a) (b)

Figure 6: Use Case-4 (UC-4): LSDS, LSDSF, EDF, LRF, and NLRF; (a) Ratio of profit obtained to total profit, (b) Percentage of total and critical packets dropped.

bitrate is about 510 Mbps at MCS11), hence, we run this use case for higher bandwidth of 160MHz. Interestingly, here LSDSF and LSDS have similar packet drop percentages, however, the number of critical packet drops is greater for LSDSF. Unlike LSDSF, LSDS not only generates optimum scheduling intervals but also finds out the optimum RU split to be used for the packet-to-interval mapping. This results in lower critical drops in LSDS than LSDSF. Notably, here EDF performs poorly as it prioritizes the applications with lower deadlines i.e., motion control and collaborative AGV. It misses out on robotic control that has a very high profit. Hence, it obtains a poor profit overall. Here, LRF correctly prioritizes all 3 important applications i.e, motion control, collaborative AGV, and robotic control. Hence, it obtains overall higher profit compared to EDF. Further, the critical loss ratio for LRF is lower compared to EDF. However, this lower ratio comes with a trade-off of higher overall loss rate for non-critical packets. Since this causes starvation for asset/process monitoring application, NLRF prioritizes that. But, due to that, it misses packets from all the critical applications as the asset/process monitoring application as well has a very high generation rate. Hence, overall NLRF obtains a poor profit ratio and high loss percentage.

Use Case-4: Figure 6 shows the ratio for UC-4. Here, we observe that LSDS incurs 0% packets drops and hence a profit ratio of 1. Even the fixed split version LSDSF incurs 0% critical drop percentage and hence a profit ratio very close to 1. The other three benchmarks incur about 18% critical packet drop percent. The reason is, EDF prioritizes Application 6 and then Application 2 for their smaller deadline. Interestingly, in this use case, two applications that have a small deadline have a high profit too. Hence, both EDF and LRF perform the same. Here, the starvation will be for applications 1, 3, 7, 10 as their scheduling metric is very less. Hence, NLRF prioritizes them. But, the generation rate of these applications is very small, atmost 1 packets/sec for Application 1, 3, and 7. Though the generation rate for Application 10 is relatively higher, it has only 1 node and the delay tolerance value

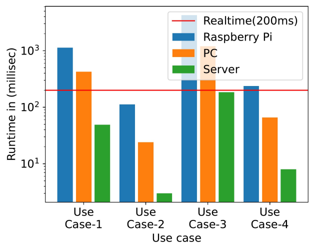
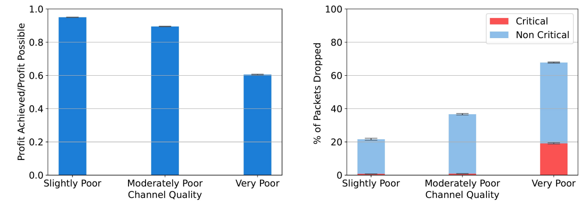


Figure 7: Runtime of LSDS on three different platforms.



(a) (b)

Figure 8: Performance of LSDS in poor channel in terms of (a) Ratio of profit obtained to total profit, and (b) Percentage of total and critical packets dropped.

is really high. Hence, the effect of scheduling these applications is minimal. All the benchmarks perform similarly. Now, EDF prioritizes Application 6 and 2 and Application 6 has a very low generation rate. As a result, the schedule created by EDF will have large gaps, that will cause another critical applications i.e., Application 4 to drop packets. The same reasoning is true for LRF too as for LRF the scheduling metric is highest for Application 6 followed by Application 2. The drop is caused by Application 4 which is critical. Hence, all the benchmarks incur $\simeq 18\%$ critical packet drops. We note here that both LSDS and LSDSF, by design prioritize packets that are of significantly high profit and schedules them in the appropriate interval. Hence, both obtain a higher profit ratio with no critical packet drops. Finally, several noncritical applications have higher packet sizes i.e., Applications 1,7,10. LSDSF was not able to schedule them in small 26 tone RUs within TXOP duration, and hence incurs noncritical packet drops.

These results show that LSDS outperforms all the other baselines in each case. While LSDSF performs worse than LSDS due to its fixed configuration of RU's, it still outperforms LRF, NLRF and EDF. This shows that both local search to schedule packets optimally (used in both LSDSF and LSDS) and search of RU configuration (used only in LSDS) significantly improve the solution to the DPMSS problem.

6.2 Runtime Analysis

We now plot the runtimes of our LSDS scheduler on three different platforms – a Raspberry Pi 4 having Quad core Cortex-A72 of frequency 1.5GHz, an AMD Ryzen 7 laptop with maximum frequency 3.2 GHz and an Intel Core i7-11700B processor server with maximum frequency 3.2 GHz (in Figure 7). We omit the benchmarks since their metrics are trivial to compute. We have different time horizons in each case, in each case smaller than 200 ms. We note that for two of the use cases UC-2 and UC-4, the execution time is smaller than 200 ms across all platforms. For UC-1 and UC-3, the runtime is smaller than 200 ms on the server. This indicates that the LSDS scheduler can run in real-time for all cases on a server. Even on less compute-intensive systems, the easier use cases (such as UC-2) run in real-time, which makes integration of LSDS less expensive in such situations.

6.3 Performance Under Poor Channel Condition

We show that LSDS performs well under challenging conditions where path losses reduce throughput. We simulate three scenarios with clients at 5 m, 15 m, and 40 m from the AP, labeled as slightly poor, moderately poor, and very poor conditions, respectively. Using the IEEE 802.11ax path loss model [2], we focus on the critical UC-3 use case. We observe in Figure 8 that LSDS does not drop any critical packets in slightly or moderately poor conditions, but does so in very poor conditions due to reduced MCS. This reduced MCS increases transmission time and prevents all packets from being scheduled by their deadline.

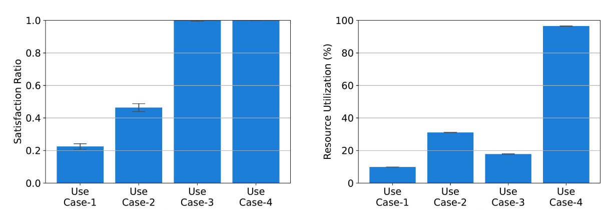
6.4 Support for Best-effort Traffic

In a factory, apart from the factory-based applications, the same WiFi network can be used to support best-effort Internet applications such as web browsing and video streaming. Hence, we propose a mechanism where the network is shared by both factory applications and best-effort ones. If a best-effort packet arrives at the i^{th} round, we try to schedule it into either the i^{th} or $(i+1)^{th}$ round's free RUs i.e., after allocating RUs to factory applications. If a packet does not get scheduled in either i^{th} or $(i+1)^{th}$ round, we convert it into a deterministic packet for the $(i+1)^{th}$ round and continue the process. Also, to make sure the packet gets scheduled eventually, the profit is increased after every round as follows: $p_j^i = (p_j^{i-1} + Threshold)/2$ where p_j is the profit of packet p. In addition to assigning best-effort traffic to free resources in the frequency domain, we utilize free time slots that are left unoccupied after assigning factory applications.

We show both the satisfaction ratio for best-effort applications and their resource utilization in Figure 9. We define the satisfaction ratio as the throughput obtained to the offered load. We create 3 best-effort nodes that generate a load following Poisson distribution with a mean of $20 Mbps$ for UC-1, 2, and 4. Since UC-3 was run at 160 Mhz, the mean is 80 Mbps. We observe that both UC-1, UC-2 have less free RUs and hence the satisfaction ratio is about 0.2 and 0.5 respectively. Further, for UC-4, best-effort traffic utilizes close to 90% resources and obtain a satisfaction ratio of 1. This corroborates our previous observation that LSDS does not drop any packets for UC-4, as it does not have high traffic load and hence most RUs were free. Since UC-3 is run at 160 MHz, even though the resource utilization by best-effort traffic is less than 20%, it was easily able to handle a load of 80Mbps. and provide a satisfaction ratio of 1. This shows that our technique of scheduling packets can efficiently accommodate additional best-effort traffic.

7 CONCLUSION

Integration of wireless networks in factory environments has been long overdue, and WiFi standards have been taking concrete steps



(a) (b)

Figure 9: Support for best-effort traffic: satisfaction ratio and resource utilization

in this direction. In line with these efforts, our work models the scheduling requirements using a system of packets and deadlines for each data packet in such scenarios. We then provide a deadline-aware scheduler for factory IoT settings. We show that the problem is NP-Hard and provide a $(12+\epsilon)$-approximation algorithm using a technique of local search. We introduce a novel variant of deadline scheduling which we hope would spark independent interest in the scheduling community. We evaluate the effectiveness of our algorithm through various simulations in terms of number of packets delivered, running time of our algorithm as well as adaptation to poor channel conditions and additional network load. We observe that our algorithm works well in practice and outperforms all the benchmarks. We believe this work provides a reason for the consideration of WiFi as a feasible technology in the industrial space. For future work, we plan to consider more dynamic situations where the deadlines and the packet arrivals are stochastic in nature.

ACKNOWLEDGMENTS

This work is supported by the Science and Engineering Research Board, Department of Science and Technology, Government of India, under Grant CRG/2022/005096. We acknowledge the help of the anonymous reviewers and the shepherd Professor Doug Blough.

REFERENCES

[1] [n.d.]. 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. https://standards.ieee.org/project/802_11ax.html [accessed 09-July-2020].

[2] [n.d.]. TGax Simulation Scenarios. https://mentor.ieee.org/802.11/dcn/14/11-14-0980-16-00ax-simulation-scenarios.docx. [accessed 01-April-2020].

[3] 2020. Flexible Factory IoT: Use Cases and Communication Requirements for Wired and Wireless Bridged Networks. *IEEE 802 Nendica Report* (Apr 2020), 1–48.

[4] May 2022. Wi-Fi 6/6E for Industrial IOT: Enabling Wi-Fi Determinism in an IoT World. https://wballiance.com/wi-fi-6-6e-for-industrial-iot-whitepaper/ Written by: "Wireless Broadband Alliance WiFi 6/6E for IIoT Project Team"; dated May 2022.

[5] Yosef Alayev, Fangfei Chen, Yun Hou, Matthew Johnson, Amotz Bar-Noy, Tom Porta, and Kin Leung. 2014. Throughput Maximization in Mobile WSN Scheduling With Power Control and Rate Selection. *IEEE Transactions on Wireless Communications* 13 (05 2014).

[6] Arjun Anand, Gustavo de Veciana, and Sanjay Shakkottai. 2020. Joint Scheduling of URLLC and eMBB Traffic in 5G Wireless Networks. *IEEE/ACM Transactions on Networking* 28, 2 (2020), 477–490.

[7] Evgeny Avdotin, Dmitry Bankov, Evgeny Khorov, and Andrey Lyakhov. 2019. OFDMA Resource Allocation for Real-Time Applications in IEEE 802.11ax Networks. In *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. 1–3.

[8] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. 2001. A Unified Approach to Approximating Resource Allocation and Scheduling. *J. ACM* 48, 5 (sep 2001), 1069–1090.

[9] Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. 1999. Approximating the Throughput of Multiple Machines Under Real-Time Scheduling. *SIAM J. Comput.* 31 (01 1999), 622–631.

[10] André Berger, Vincenzo Bonifaci, Fabrizio Grandoni, and Guido Schäfer. 2011. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Mathematical Programming* 128, 1-2 (2011), 355–372.

[11] Piotr Berman and Bhaskar Dasgupta. 2000. Multi-phase Algorithms for Throughput Maximization for Real-Time Scheduling. *Journal of Combinatorial Optimization* 4 (04 2000), 307–323.

[12] Gianluca Cena, Stefano Scanzio, and Adriano Valenzano. 2016-04. Seamless Link-Level Redundancy to Improve Reliability of Industrial Wi-Fi Networks. *IEEE Transactions on Industrial Informatics* 12, 2 (2016-04), 608–620.

[13] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. 2022. Maximum Flow and Minimum-Cost Flow in Almost-Linear Time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. 612–623.

[14] Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. 2006. Approximation Algorithms for the Job Interval Selection Problem and Related Scheduling Problems. *Mathematics of Operations Research* 31, 4 (2006), 730–738.

[15] Ran Duan and Seth Pettie. 2014. Linear-Time Approximation for Maximum Weight Matching. *J. ACM* 61, 1, Article 1 (jan 2014), 23 pages.

[16] Hossam Farag, Emiliano Sisinni, Mikael Gidlund, and Patrik Österberg. 2019. Priority-Aware Wireless Fieldbus Protocol for Mixed-Criticality Industrial Wireless Sensor Networks. *IEEE Sensors Journal* 19, 7 (2019), 2767–2780.

[17] Roman Froschauer, Aljosha Köcher, Kristof Meixner, Siwara Schmitt, and Fabian Spitzer. 2022. Capabilities and Skills in Manufacturing: A Survey Over the Last Decade of ETFA. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. 1–8.

[18] M. R. Garey and D. S. Johnson. 1977. Two-Processor Scheduling with Start-Times and Deadlines. *SIAM J. Comput.* 6, 3 (1977), 416–426.

[19] Alexej Grigorjew, Michael Seufert, Nikolas Wehner, Jan Hofmann, and Tobias Hoßfeld. 2021. ML-Assisted Latency Assignments in Time-Sensitive Networking. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 116–124.

[20] Naman Gupta, Syamantak Das, and Mukulika Maity. 2022. FairSplit-An efficient near-optimal bandwidth splitting strategy for OFDMA in IEEE 802.11 ax. In *ICC 2022-IEEE International Conference on Communications*. IEEE, 285–290.

[21] Muhammad Inamullah, Bhaskaran Raman, and Nadeem Akhtar. 2020. Will My Packet Reach On Time? Deadline-Based Uplink OFDMA Scheduling in 802.11ax WLANs. In *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*. 181–189.

[22] Mohit Jain, Anis Mishra, Arani Bhattacharya, and Mukulika Maity. 2024. Source code of DPMSS. https://github.com/arani89/WiFi6-DeadlineScheduling.

[23] Mohit Jain, Anis Mishra, Andreas Wiese, Syamantak Das, Arani Bhattacharya, and Mukulika Maity. 2024. A Deadline-Aware Scheduler for Smart Factory using WiFi 6. https://arxiv.org/abs/2408.12274.

[24] Jobish John, Md. Noor-A-Rahim, Aswathi Vijayan, H. Vincent Poor, and Dirk Pesch. 2023. Industry 4.0 and Beyond: The Role of 5G, WiFi 7, and TSN in Enabling Smart Manufacturing. arXiv:2310.02379 [cs.NI] https://arxiv.org/abs/2310.02379

[25] Maria Kovacova and Elizabeth Lewis. 2021. Smart factory performance, cognitive automation, and industrial big data analytics in sustainable manufacturing internet of things. *Journal of Self-Governance and Management Economics* 9, 3 (2021), 9–21.

[26] Dániel Kozma, Gábor Soós, Dániel Ficzere, and Pal Varga. 2019. Communication challenges and solutions between heterogeneous Industrial IoT systems. In *15th IEEE International Conference on Network and Service Management (CNSM)*.

[27] Roberto Maldonado, Anders Karstensen, Guillermo Pocovi, Ali A Esswie, Claudio Rosa, Olli Alanen, Mika Kasslin, and Troels Kolding. 2021. Comparing Wi-Fi 6 and 5G downlink performance for industrial IoT. *IEEE Access* 9 (2021), 86928–86937.

[28] Sharan Naribole, Wook Bong Lee, and Ashok Ranganath. 2019. Impact of MU EDCA channel access on IEEE 802.11 ax WLANs. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 1–5.

[29] Changhua Pei, Zhi Wang, Youjian Zhao, Zihan Wang, Yuan Meng, Dan Pei, Yuanquan Peng, Wenliang Tang, and Xiaodong Qu. 2017. Why it takes so long to connect to a WiFi access point. In *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 1–9.

[30] Li-On Raviv and Amir Leshem. 2023. Joint Scheduling and Resource Allocation for Packets With Deadlines and Priorities. *IEEE Communications Letters* 27, 1 (2023), 248–252. https://doi.org/10.1109/LCOMM.2022.3211337

[31] Pedram Kheirkhah Sangdeh and Huacheng Zeng. 2021. DeepMux: Deep-Learning-Based Channel Sounding and Resource Allocation for IEEE 802.11 ax. *IEEE Journal on Selected Areas in Communications* 39, 8 (2021), 2333–2346.

[32] Matthias Schneider, Fabian Haag, Abdul Kader Khalil, and David Albert Breunig. 2022. Evaluation of communication technologies for distributed industrial control systems: Concept and evaluation of 5G and wiFi 6. *Proceedings of the 55th CIRP Conference on Manufacturing Systems* 107 (2022), 588–593.

[33] Lucia Seno, Gianluca Cena, Stefano Scanzio, Adriano Valenzano, and Claudio Zunino. 2017-04. Enhancing Communication Determinism in Wi-Fi Networks for Soft Real-Time Industrial Applications. *IEEE Transactions on Industrial Informatics* 13, 2 (2017-04), 866–876.

[34] Marko Vuković, Daniele Mazzei, Stefano Chessa, and Gualtiero Fantoni. 2021. Digital Twins in Industrial IoT: a survey of the state of the art and of relevant standards. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. 1–6.

[35] Kaidong Wang and Konstantinos Psounis. 2018. Scheduling and Resource Allocation in 802.11 ax. In *IEEE Conference on Computer Communications (INFOCOM)*. 279–287.

[36] Yi-Hung Wei, Quan Leng, Song Han, Aloysius K Mok, Wenlong Zhang, and Masayoshi Tomizuka. 2013. RT-WiFi: Real-time high-speed communication protocol for wireless cyber-physical control applications. In *2013 IEEE 34th Real-Time Systems Symposium*. 140–149.