Admission Control and Scheduling of Isochronous Traffic in IEEE 802.11ad MAC

Anirudha Sahoo, Weichao Gao, Tanguy Ropitault and Nada Golmie
Email:{anirudha.sahoo,weichao.gao,tanguy.ropitault,nada.golmie}@nist.gov
National Institute of Standards and Technology
Gaithersburg, Maryland, USA

ABSTRACT

An upsurge of low latency and bandwidth hungry applications such as virtual reality, augmented reality and availability of unlicensed spectrum in the mmWave band at 60 GHz have led to standardization of the next generation WiFi such as IEEE 802.11ad and 802.11ay. Due to the stringent Quality of Service (QoS) requirement of those applications, 802.11ad/ay have introduced contention free channel access called *Service Period*, which provides dedicated channel access exclusively reserved for communication between a pair of nodes. One type of user traffic supported by IEEE 802.11ad is isochronous traffic, which is essentially periodic traffic that requires certain channel time to be allocated before its period ends. So, isochronous traffic needs guaranteed channel time allocation with stringent deadlines. In this paper, we present three Admission Control Algorithms (ACAs) which admit isochronous requests to achieve the above goals while being fair. We also present an Earliest Deadline First (EDF) based scheduling algorithm for isochronous traffic. We evaluate the performance of the three ACAs in terms of different performance metrics. Our simulation results show that, out of the three ACAs, the proportional fair allocation based algorithm offers the best tradeoff across different performance metrics.

CCS CONCEPTS

• **Networks → Wireless access points, base stations and infrastructure**; **Wireless local area networks**; *Mobile networks.*

KEYWORDS

IEEE 802.11ad, MAC, admission control, scheduling

ACM Reference Format:
Anirudha Sahoo, Weichao Gao, Tanguy Ropitault and Nada Golmie. 2021. Admission Control and Scheduling of Isochronous Traffic in IEEE 802.11ad MAC. In *Proceedings of the 24th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '21), November 22–26, 2021, Alicante, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3479239.3485698

1 INTRODUCTION

There has been an upsurge of networking applications that require high throughput and low delay service. Applications such as Virtual Reality (VR), Augmented Reality (AR), wireless backhaul, high bandwidth connectivity with 8K TVs are some of the examples that demand Quality of Service (QoS) in terms of delay and throughput. A large amount of bandwidth availability in the unlicensed millimeter wave (mmWave) band at 60 GHz has led to the formulation of next generation WiFi standards like IEEE 802.11ad and 802.11ay, also known as *Wireless Gigabit* (WiGig). These standards make use of large channel bandwidth (2.1 GHz), Multiple Input Multiple Output (MIMO) and channel bonding techniques to provide very high data rate of over 100 Gbps [6, 11]. Due to its probabilistic channel access mechanism, stringent QoS requirement of the above said applications cannot be met by contention based channel access traditionally used in older WiFi standards. Hence, in 802.11ad/ay, the WiGig standard supports contention free channel access referred to as *service period* (SP) which is suitable for QoS based applications. An SP is a dedicated channel duration exclusively reserved for communication between a pair of nodes. However, the standard does not specify how to schedule those SPs for user traffic.

In this paper, we present admission control and scheduling algorithms for a particular type of user traffic called *isochronous* traffic in a IEEE 802.11ad system. Isochronous traffic is essentially periodic traffic and needs certain amount of channel time before its period ends. The challenge for the admission control and scheduler is to be able to guarantee SP duration to admitted user traffic (or requests) before their respective deadlines, while admitting high number of requests and achieving high channel utilization. The algorithms also should be fair while allocating channel time to different users. So, we propose three Admission Control Algorithms (ACAs) which have the above properties. Since isochronous traffic has deadlines to meet, our scheduling algorithm is based on Earliest Deadline First (EDF) scheduling algorithm used in scheduling tasks in real time systems [3]. We evaluate the three admission control algorithms in terms of different performance metrics. Our simulation results show that, among the three algorithms, the Proportional Fair Admission Control (PFAAC), though computationally more complex than the other two, performs best in terms of some metrics, and slightly worse with resptect to other metrics. Hence, PFAAC is an appropriate ACA for an IEEE 802.11ad system to get an overall good performance in terms of all metrics. The main contributions of this work are: i) we mapped a deadline driven Central Processing Unit (CPU) task scheduling algorithm to IEEE 802.11ad Medium Access Control (MAC) framework, ii) proposed three fair admission control algorithms and iii) exploited static scheduling (in a beacon interval) requirement of IEEE 802.11ad MAC to propose a simplified EDF scheduler for isochronous traffic.

MSWiM '21, November 22–26, 2021, Alicante, Spain
2021. ACM ISBN 978-1-4503-9077-4/21/11.
https://doi.org/10.1145/3479239.3485698

2 DESIGN OF IEEE 802.11AD ADMISSION CONTROL AND SCHEDULING

2.1 IEEE 802.11ad Medium Access

The medium access time in IEEE 802.11ad consists of an infinite sequence of time durations called Beacon Intervals (BI). A BI is expressed in Time Units (TU), where $1\,TU = 1024\,\mu s$. Each BI consists of a Beacon Header Interval (BHI) followed by a Data Transmission Interval (DTI). The DTI is used for data exchanges and beamforming training among IEEE 802.11ad Stations (STAs) and Personal Basic Service Set (PBSS) Control Point/Access Point (PCP/AP). During a DTI, channel access is specified in two ways. A Contention Based Access Period (CBAP) duration implies STAs should access the channel using contention based scheme called Enhanced Distributed Channel Access (EDCA) [1]. A SP type of channel access is intended for communication between two STAs or between a STA and PCP/AP without any contention. CBAP and SP schedules in a DTI are announced by the PCP/AP in a Directional Multi Gigabit (DMG) Beacon frame in the Beacon Transmission Interval (BTI) or Announce frame in the Announcement Transmission Interval (ATI) of BHI (before DTI period starts) [1].

2.2 IEEE802.11ad Traffic

IEEE 802.11ad has two types of user traffic: *isochronous* and *asynchronous*. Isochronous traffic is suitable for applications that require periodic data transmission with certain QoS requirements. Asynchronous traffic, on the other hand, is a *one time* request, although the requested duration may be granted in multiple allocations. Asynchronous traffic may be *best effort* or may have certain QoS requirements. A station sends *Add Traffic Stream* (ADDTS) requests to its PCP/AP to request resources for its isochronous or asynchronous traffic. The Traffic Specification (TSpec) element in the ADDTS request carries the traffic parameters for which resources need to be allocated. Because of the periodic nature of isochronous traffic, scheduling of this type of traffic is more challenging. Hence, in this paper, we study scheduling of isochronous traffic. The most important traffic parameters of isochronous traffic are [1]:

- Allocation Period (P): Period over which allocation repeats. It can only be an integer multiple or integer fraction of the BI.
- Minimum Allocation (C^{min}): Minimum acceptable allocation in microseconds in each allocation period. If the request is accepted, the PCP/AP must guarantee at least this duration to the STA in every allocation period.
- Maximum Allocation (C^{max}): Requested allocation in microseconds in each allocation period. This is the maximum duration that can be allocated to the user in each allocation period.
- Minimum Duration: Minimum duration in microseconds in each allocation period. An allocation may be split into multiple chunks. Each chunk must be larger than or equal to this duration. The user can set this value to zero to indicate that this parameter should not be considered. In this study, we assume this parameter to be zero to keep the problem simple.

In this paper, we study scheduling of isochronous traffic using SP in the DTI period. Let C_i^{op} be the duration allocated to isochronous request T_i whose traffic parameters are $(C_i^{min}, C_i^{max}, P_i)$. The allocated duration C_i^{op} may change over the lifetime of the request, but it must always satisfy $C_i^{min} \leq C_i^{op} \leq C_i^{max}$. Since this is a periodic request, C_i^{op} must be allocated in every P_i interval. At the beginning of every period, the request is ready to be served, i.e., SP duration can be allocated anytime after the beginning of the period and the corresponding deadline is its period. We refer to the beginning of every period as the *release time* of the request. Denoting R_{i_n} as the n^{th} release time of isochronous request T_i, we have $R_{i_0} = 0$, $R_{i_1} = P_i$, $R_{i_2} = 2P_i$ and so on. In IEEE 802.11ad, when a request arrives, if admitted, it is scheduled in the next BI. Hence, release time $R_{i_0} = 0$ of a request refers to the start of the next BI. These periodic releases of a request contribute to the load on the system and are captured as its *demand*. So, demand of a request T_i is represented as a series of triples $(C_{i_n}^{op}, P_i, R_{i_n})$, $n = 0, 1, .., $ where $C_{i_n}^{op}$ is the duration to be allocated between R_{i_n} and $R_{i_{n+1}}$. Each triple in a demand is referred to as a *job* of the request.

2.3 Central Processing Unit Scheduling of Periodic Tasks

Scheduling of isochronous traffic is very similar to Central Processing Unit (CPU) scheduling of periodic tasks. CPU scheduling of periodic task has been extensively studied in the literature [3, 5, 8, 13]. In this context, a periodic task T_i is modeled with two parameters (C_i, P_i), where C_i is the duration of the task and P_i is the period as well as deadline of the task. So, the task must be allocated CPU time C_i in every P_i time duration. In [3], the authors present two preemptive scheduling algorithms for periodic tasks: Rate Monotonic Scheduling (RMS) and Earliest Deadline First (EDF) scheduling. Priority of a task in an RMS scheduler is static and a task with lower period is assigned higher priority. The EDF scheduler sets higher priority to a task with earlier deadline. The priority, in this case, is dynamic. Although RMS is a simpler scheduler than EDF, the maximum utilization that can be achieved while guaranteeing that every task meets its deadline is approximately $\ln 2 \approx 69\,\%$ for a large set of tasks [3]. EDF scheduler can achieve maximum utilization of $100\,\%$ while guaranteeing the deadline of each task. Hence, we choose EDF scheduler to schedule isochronous traffic.

An EDF scheduler can be preemptive or non-preemptive. The feasibility or admissibility of a set of n preemptive tasks for an EDF scheduler is given by [3]

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1. \tag{1}$$

The feasibility of a set of n non-preemptive tasks, in addition to the condition in Eq. (1), needs another condition to be satisfied as given in Theorem 4.1 in [8]. This second condition for non-preemptive task involves finding least upper bound on the processor demand between periods of the tasks and is more complex than the condition given in Eq. (1). Hence, the admission control for non-preemptive EDF scheduler is more complex than its preemptive counterpart. In the case of IEEE 802.11ad MAC, when a new request arrives, it is scheduled in the next BI. Hence, in a given BI, requests are scheduled as per their priority based on their deadline (which is their period). This schedule is not perturbed (i.e., does not change) by the arrival of a new request, unlike what happens in CPU scheduling of periodic tasks. This property of IEEE 802.11ad MAC leads to a static schedule in each BI (even though a EDF is a dynamic scheduling algorithm), i.e., the schedule does not change in a given BI due to arrival of a new request. So, we choose preemptive EDF as our scheduler in IEEE 802.11ad MAC to take advantage of simpler admission control and scheduling.

2.4 Admission Control

We borrow the basic principles of admission control from the *feasibility* test of CPU scheduling of periodic tasks. However, the TSpec of isochronous traffic has a range of duration from C^{min} to C^{max}, unlike the CPU scheduling of a task which has a single duration. So, in our case, an admission control algorithm not only determines whether a new request can be admitted or not, but also computes C^{op}, $C^{min} \le C^{op} \le C^{max}$, the exact operating allocation duration of the newly admitted request. Let us assume that there are $(n-1)$ requests already in the system. These requests were admitted at their corresponding C^{op}s. Depending on the admission control algorithm used, the duration to be allocated to the newly arriving n^{th} request, T_n, is computed to be C_n^{op}. The newly arriving isochronous request is admitted if and only if

$$U + \frac{C_n^{op}}{P_n} \le 1, \qquad (2)$$

where $U = \sum_{i=1}^{(n-1)} \frac{C_i^{op}}{P_i}$ is the utilization of the system due to already admitted requests. Depending on the admission control algorithm used, while admitting a new request, C^{op} of existing requests may or may not change during the life of the requests. If C^{op} of existing requests may change, then utilization of the system due to existing requests U needs to be recalculated. In this case, the complexity of the admission control algorithm depends on the complexity of computing U and C^{op}. Also note that C^{op} changes only when a new request is admitted and when an existing request leaves the system. To keep the notations simple, we do not make C^{op} a function of time. Hence, when we refer to C^{op}, it refers to its value at that instant. However, if C^{op} of existing requests cannot change throughout the life of the requests, then U does not need to be computed every time a new request needs to be admitted, but can be updated and maintained in the system as and when new requests are admitted or existing requests leave the system. The complexity of the admission control algorithm, in this case, depends only on complexity of computation of C^{op}.

2.5 Our EDF Based Scheduler

As mentioned earlier, when admitting a request, the admission control algorithm computes the C^{op} of the request. The responsibility of the scheduler is to allocate C^{op} duration to the request before its deadline (which is equal to its period). The allocation may be a contiguous duration or a set of non-contiguous fragments. The flowchart of our preemptive EDF scheduler is shown in Figure 1. The figure illustrates how the schedule is computed for a given BI. The Algorithm starts with ordering the jobs of all the requests

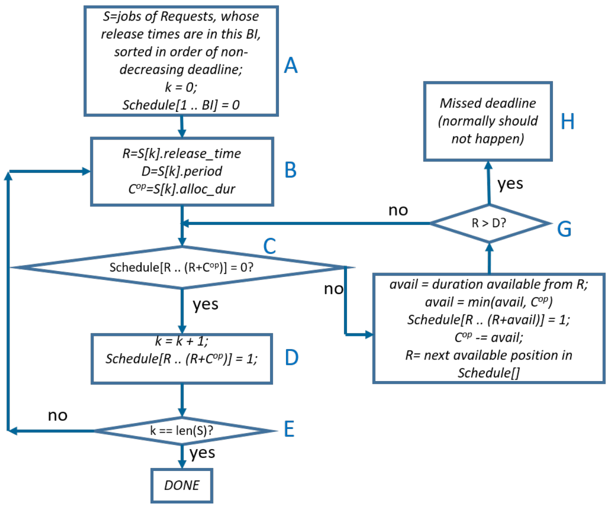

S=jobs of Requests, whose release times are in this BI, sorted in order of non-decreasing deadline; k = 0; Schedule[1 .. BI] = 0
A
R=S[k].release_time D=S[k].period C^op=S[k].alloc_dur
B
Schedule[R .. (R+C^op)] = 0?
C
yes
k = k + 1; Schedule[R .. (R+C^op)] = 1;
D
k == len(S)?
E
no
yes
DONE
no
avail = duration available from R; avail = min(avail, C^op) Schedule[R .. (R+avail)] = 1; C^op -= avail; R= next available position in Schedule[]
F
R > D?
G
no
yes
Missed deadline (normally should not happen)
H

Figure 1: Flowchart of the Proposed EDF Scheduler

in a non-decreasing order based on their deadlines and initializing some variables (Box A). It picks up the ordered jobs one at a time, extracts release time (R), deadline (D) (which is same as the period of the job), and the allocation duration (C^{op}) as shown in Box B. It then checks if the BI has unallocated contiguous duration C^{op} available starting from its release time (Decision Box C). If so, then that part of the BI is allocated to the job (Box D) and then the Algorithm loops back to schedule the next job, if there is one (Decision Box E). Otherwise, whatever duration (which is less than C^{op}) is available, is assigned to the request, C^{op} is decremented to determine the remaining duration to be allocated and the allocation point is advanced to the next unallocated (or empty) location in the BI (Box F). Note that if Box F is reached during scheduling of a job, then that job is fragmented, which is akin to preemption in CPU task scheduling parlance. If the next unallocated location is greater than the deadline (Decision Box G), then the request has missed its deadline (Box H). This is an error condition and should not happen in a correct implementation. Otherwise, the Algorithm repeats the process of finding unallocated contiguous duration for the reduced C^{op} (Decision Box C). Each request will search for free locations in a BI once (for all its jobs). Hence, the time complexity of our EDF based scheduler is $O(BI \cdot n)$, where n is the number of requests in the system.

Figure 2 illustrates our EDF based scheduling algorithm using an example. There are three requests, T_1, T_2 and T_3, whose C^{op} values are shown in filled rectangles. The release times of the requests are indicated by color-coded vertical arrows. Their periods, P_1, P_2 and P_3 are also shown. These parameters define jobs of the three requests in one BI duration. Request T_1 has eight jobs, T_2 has three jobs and T_3 has two jobs in one BI. The job number for each request is shown inside the rectangle representing the job. These jobs are first ordered in non-decreasing order of their respective deadlines. These allocation order numbers are shown as integer numbers on top of each job. When the C^{op} of a job cannot be allocated contiguously, then the allocation is *fragmented* and the fragment number is shown in parenthesis as a superscript

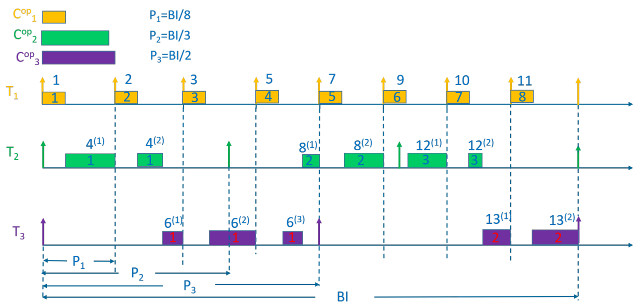


Figure 2: An Example Illustrating Our EDF Based Scheduler

to the order number. For example, the first three jobs of T_1 come first in the order, then the first job of T_2 (order number 4), since its deadline is before the fourth job of T_1 (order number 5). The first three jobs of T_1 are allocated contiguous durations each equal to C_1^{op}. Then the first job of T_2 is picked up for allocation. Since first job of T_1 has already been allocated C_1^{op} from the beginning of the BI, T_2 is allocated SP duration immediately following that allocation. However, the duration available is smaller than C_2^{op}. Hence, T_2's allocation is fragmented. The first fragment (shown with a superscript (1)) occupies the empty space in between the first and second allocation of T_1. The second fragment of first job of T_2 is then allocated after the allocation of second job of T_1. Following the flowchart of Figure 1, we end up with the schedule as shown in Figure 2. We want to reiterate that any new request arriving in the middle of a BI is considered for scheduling in the next BI. Hence, the schedule computed by our algorithm is not perturbed by a newly arriving request and therefore, remains static (or does not change) throughout the duration of the BI. In fact, this schedule will continue in the subsequent BIs until a new request arrives or an existing request leaves.

2.6 Choice of Admission Control Algorithms

The allocation duration in isochronous TSpec has two parameters, C_{min} and C_{max}. Hence, it provides flexibility to the ACAs in terms of choosing the operating allocation amount C^{op}. The strategy depends on the goal of the ACA. An ACA may aim to maximize the number of requests admitted into the system and hence would allocate minimum possible duration, i.e., it would set $C^{op} = C_{min}$ for all requests. Another ACA may have the goal to maximize allocation duration to all its requests, i.e., set $C^{op} = C_{max}$. But this ACA would take a performance hit in terms of number of requests admitted. But regardless of goal of an ACA, it should be fair to all its requests. For example, an ACA should not admit one request with C_{min} and another with C_{max}. Hence, we discuss fairness of an ACA in the next section and present three ACAs which are fair.

2.7 Fairness of Admission Control Algorithm

We use Jain's fairness index for each ACA. Let

$$x_i = \frac{C_i^{op} - C_i^{min}}{C_i^{max} - C_i^{min}}. \tag{3}$$

Then Jain's Fairness Index (JFI) for our ACA is defined as

$$\text{JFI} = \begin{cases} 1, & \text{if } x_i = 0, \forall i. \\ \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2}, & \text{otherwise.} \end{cases} \tag{4}$$

The ACA decides the C^{op} of the requests admitted into the system. We propose three admission control algorithms which are absolutely fair, i.e., JFI = 1.

- Minimum Allocation Admission Control (MnAAC): Every request is allocated its C^{min}, i.e., $C_i^{op} = C_i^{min}$ for $i = 1, \ldots, n$. While this allocation is compliant with the IEEE 802.11ad requirement, it is not efficient in terms of allocation especially when the system is lightly loaded. However, this algorithm will have the best performance in terms of number of requests admitted. For this ACA, C^{op} of a request does not change throughout its lifetime. Hence utilization of the system can be updated in constant time as a new request is admitted or when an existing request leaves the system. In addition, computation of C^{op} is also constant time. Hence, time complexity of this ACA is $O(1)$.
- Maximum Allocation Admission Control (MxAAC): Every request is allocated its C^{max}, i.e., $C_i^{op} = C_i^{max}$ for $i = 1, \ldots, n$. This allocation does not change throughout the life of the requests. This allocation is advantageous for the admitted requests, since they always get the maximum allocation. However, in general, it would admit lesser number of requests compared to MnAAC scheduler, especially at high load condition. Following similar argument as in the case of MnAAC, the time complexity of this ACA is also $O(1)$.
- Proportional Fair Allocation Admission Control (PFAAC): Requests are allocated operational duration (C^{op}) such that surplus allocation over C^{min} expressed as a fraction of the requested allocation range ($C^{max} - C^{min}$) is same for all the requests, i.e., $x_1 = x_2 = \ldots = x_n$, where x_i is given by Eq. (3). Note that allocation durations of admitted requests can change during their lifetimes when this algorithm is used. This algorithm tries to allocate as much duration as possible while maximizing the number of admitted requests.

2.8 Allocation in PFAAC

Allocation of C^{op} in MnAAC and MxAAC is very straightforward. However, when PFAAC is used, the operational allocation durations (C^{op}) of existing requests may change. Hence, the PFAAC algorithm computes new allocation duration (C^{op}) of existing requests as well as the new request when a new request is admitted and when an existing request leaves the system. Pseudocode for this algorithm is presented in Algorithm 1. The algorithm rejects the new request if the utilization of the system, assuming all the existing requests and the new request are allocated their respective minimum allocation (C^{min}), exceeds 1 (Line 4). Otherwise, it computes the *surplus* utilization based on this minimum allocation (Line 5). This surplus utilization ($U_{surplus}$) is then distributed to each individual request in proportion to its difference of utilization between maximum and minimum to the total (over all requests) difference of utilization. Hence, the operating utilization of request T_i is given by

Algorithm 1 **Admission_Control_PFAAC**

```
1: input: $C^{min}$, $C^{max}$ and $P$ of all existing $(n-1)$ requests and the new request.
2: output: Accept or Reject; $C^{op}$ of each request if the new request is accepted.
3: $U_n^{min} = \sum_{i=1}^{n} \frac{C_i^{min}}{P_i}$
4: if $U_n^{min} > 1$ then return Reject
5: $U_{surplus} = 1 - U_n^{min}$
6: $\Delta u_{tot} = 0$
7: for i=1 to n do
8:     $\Delta u_{tot} = \Delta u_{tot} + \frac{C_i^{max} - C_i^{min}}{P_i}$
9: for i=1 to n do
10:    $C_i^{op} = C_i^{min} + \min(1, \frac{U_{surplus}}{\Delta u_{tot}}) \cdot (C_i^{max} - C_i^{min})$
11: return Accept
```

$$u_i^{op} = u_i^{min} + \frac{\Delta u_i}{\Delta u_{tot}} \cdot U_{surplus}, \qquad (5)$$

where $u_i^{op} = \frac{C_i^{op}}{P_i}$, $\Delta u_i = \frac{C_i^{max}}{P_i} - \frac{C_i^{min}}{P_i}$ and $\Delta u_{tot} = \sum_{i=1}^{n} \Delta u_i$. Multiplying both sides of Eq. (5) by P_i and inserting the term $\min(1, \frac{U_{surplus}}{\Delta u_{tot}})$ to take care of the fact that $\frac{U_{surplus}}{\Delta u_{tot}}$ could be greater than 1, we get

$$C_i^{op} = C_i^{min} + \min(1, \frac{U_{surplus}}{\Delta u_{tot}}) \cdot (C_i^{max} - C_i^{min}), \qquad (6)$$

which is the expression in Line 10. Note that when $\frac{U_{surplus}}{\Delta u_{tot}}$ is greater than 1, it implies that there is enough surplus for every request, so that every request can be allocated its C^{max}. In this case, it can be verified from Eq. (6) that $C_i^{op} = C_i^{max}$. This also explains why the term $\min(1, \frac{U_{surplus}}{\Delta u_{tot}})$ should be used in Eq. (6) to prevent C_i^{op} going above C_i^{max}. Noting the for loops in Line 7 and Line 9 of Algorithm 1, each of which iterates n times, it is obvious that the time complexity of this ACA is $O(n)$.

3 PERFORMANCE RESULTS

3.1 Performance Metrics

In this section, we define the performance metrics used to evaluate our admission control algorithms.

- Acceptance Ratio (AR) : This is the fraction of total requests that are admitted by the ACA. Higher AR implies that an IEEE 802.11ad MAC can support a greater number of flows or applications.
- Allocation Efficiency (AE): This metric represents efficiency of an ACA in terms of allocation of SP duration. For a request T_i, it is defined as $\frac{C_i^{op} - C_i^{min}}{C_i^{max} - C_i^{min}}$. Note that AE is a value between 0 and 1. AE is 0 when the ACA allocates C_i^{min} to T_i and is 1 when it allocates C_i^{max}. An ACA with higher AE is preferable to an application since that translates to higher throughput for the application.
- BI Utilization (BU): This metric is the fraction of a BI duration that has been allocated to requests for SP channel access by the corresponding STAs.
- Degree of Fragmentation (DoF): When C_i^{op} of a request T_i cannot be allocated in one chunk, then the allocation is said to be *fragmented*. DoF_i of a request T_i in a given duration is given by $\frac{N_{chunk} - N_{job}}{N_{job}}$, where N_{chunk} is the number of chunks allocated to the request (during scheduling) and N_{job} is the number of jobs of the request in that duration. Thus, if there is no fragmentation of a request T_i, then $N_{chunk} = N_{job}$, and $DoF_i = 0$. In IEEE 802.11ad, a *guard time* is inserted between two consecutive allocations which is an overhead for the system. Hence, DoF is an indicator of guard time overhead of the system.
- Normalized Delay: The delay of a job of a request is the difference of time instance of end of allocation of the job and the release time of the job. Note that if the job is fragmented, the end of allocation of the job is the end of allocation of its last fragment. This metric essentially measures the time it would take to finish transmitting a job after it is available (released) at the MAC and is an indicator of packet level delay. Normalized delay is the delay normalized with respect to the period of the request, i.e., it is the ratio of delay of a job to the period of the request.
- Normalized Jitter: Jitter is the absolute difference of delay of two consecutive jobs of a request. In another words, it is the variation in two successive delays of two consecutive jobs of a request. This is normalized with respect to the period of the request to represent normalized jitter. Hence, normalized jitter is the ratio of jitter to the period of the request.

3.2 Simulation Experiment Design

To evaluate performance of our admission control and scheduling algorithms we design our simulation experiments as follows. AD-DTS requests arrive with a Poisson distribution having mean arrival rate λ, which is varied from 5 to 50 requests per BI in steps of 5. BI duration is set at 102 400 μs. The maximum allocation duration of a request (C^{max}) is uniformly distributed between 10 μs and 100 μs and is considered a per BI value. Thus, if the request period is an integer fraction of BI, then the randomly generated C^{max} value is scaled down by that fraction. If the request period, on the other hand, is an integer multiple of BI, then C^{max} value is scaled up by that integer multiple. The allocation interval ratio, C_{min}/C_{max}, is uniformly distributed between 0.5 and 1.0. The lifetime of each request follows normal distribution with the average duration of 100 BIs and standard deviation of 10 BIs. Thus, about 95 % of lifetimes are between 80 BIs to 120 BIs. Lifetime of a request is rounded down to its nearest period. The integer n that defines the integer fraction or integer multiple of period of a request is uniformly distributed between 1 to 5. The experiments are carried out in three *scenarios*: i) when all the requests have periods which are an integer multiple of BI (Scenario 1), ii) when all the requests have periods which are an integer fraction of BI (Scenario 2) and iii) when 30 % requests have periods which are integer multiple of BI and 70 % requests have periods which are integer fraction of BI (Scenario 3). The experiments are run for a duration equal to 1000 BIs. For a given random parameter, the same sequence of random numbers are used to represent the values of the parameter across the three scenarios, i.e., same seed is used across the three scenarios to generate the random parameter. This ensures that all the scenarios are fed with the same values of input parameters which makes the comparison across scenarios fair. Scenario 1 simulates applications requiring low bandwidth (e.g., IoT applications), whereas Scenario 2 represents applications requiring high bandwidth and low delay (e.g., streaming video). Scenario 3 is appropriate for a mix of these two types of applications.

3.3 Experiment Results

Before we present our results, we explain the *box and whisker plots* used in some of the performance metrics. The central mark of each box is the median, the edges of the box are the 25^{th} and 75^{th} percentiles. The upper (lower) *whisker* represents the largest (smallest) data point that is within 1.5 times the interquartile range (distance between the upper and lower quartiles) above (below) the upper (lower) box edge. Outliers beyond the whiskers are not shown.

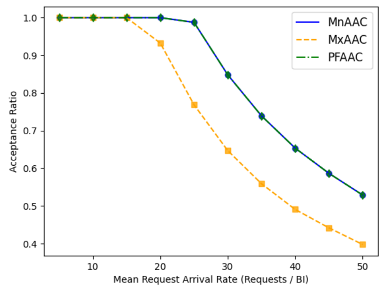
Figures 3(a), 3(b), and 3(c) show how acceptance ratio (AR) changes as the mean request arrival rate λ increases for the three scenarios. At low λ, the AR is 100 %, since the system has low utilization. But as λ increases, load on the system increases and more requests are rejected. Hence, AR starts to fall. AR for MxAAC is lower than MnAAC at high λ. Since MxAAC admits requests with their $C^{op} = C^{max}$, utilization of the system reaches 100 % with fewer requests and hence it admits less number of requests (see admission criterion Eq. (2)). AR of MnAAC and PFAAC are identical, since PFAAC admits connection based on C^{min} and distributes any surplus utilization to all the requests in a proportional manner (see Line 4 in Algorithm 1). Each ACA's performance is nearly identical across the three scenarios. This is because the utilization of the three scenarios at any given λ is almost the same, since the same sequence of random numbers represent the values of the parameters across the scenarios. In Scenario 1, for some requests, lifetime is rounded down to their respective periods. This contributes to the small discrepancies across the scenarios.

Performance graphs of average BI utilization versus mean request arrival rate are presented in Figures 4(a), 4(b), and 4(c). Once the system reaches steady state (after a few BIs), BI utilization is averaged over all the BIs until the end of the experiment duration to obtain average BI utilization. As expected, BI utilization increases as λ increases for all the scenarios and ACAs. However, beyond a certain λ value, the BI utilization remains almost at 100 %. At that λ value, the admitted requests use up all the BI duration. Any new incoming request is rejected until an existing request leaves the system. Thus, around the same λ value, AR starts to drop from 100 % (see Figures 3(a), 3(b), and 3(c)). MnAAC reaches 100 % BI utilization at a higher λ value than MxAAC and PFAAC since it allocates smaller duration (C^{min}) for a given request compared to MxAAC and PFAAC. Performance of MxAAC and PFAAC are almost equal since PFAAC tries to allocate $C^{op} = C^{max}$ whenever possible, but as BI utilization approaches 100 % (at around λ = 20 to 25 requests per BI), it tries to admit more requests at lower C^{op}. This leads to slight difference between MxAAC and PFAAC at that λ value. The performance of the three ACAs is almost identical across the three scenarios, because the requests in all the three ACAs are admitted based on utilization. Therefore, for a given ACA, the same requests are admitted across the scenarios which leads to almost the same BI utilization. As mentioned before, the slight discrepancy across the scenarios is the result of rounding down lifetime of some of the requests in Scenario 1.

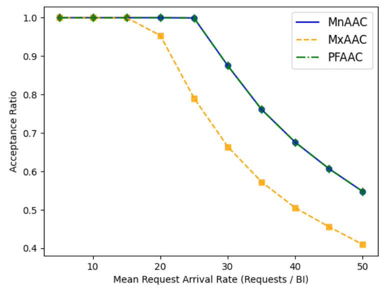
Figures 5(a), 5(b), and 5(c) show the performance of ACAs in terms of average AE of the requests as mean arrival rate of requests (λ) increases. To compute the average AE of a given request, AE is computed in every period and averaged over its lifetime. Then average AEs of all the requests are plotted as a box and whisker plot. As expected, average AE of MnAAC and MxAAC remain constant at 0 and 1 respectively. Initially AE of PFAAC remains at 1.0, but between λ = 20 requests per BI and λ = 25 requests per BI, it falls below 1.0 and becomes 0.0 beyond λ = 25 requests per BI for all the three scenarios. Recall that at around λ = 20 to 25 requests per BI, the BI utilization reaches 100 % and hence PFAAC sacrifices on allocation efficiency to admit more requests.

Performance in terms of average DoF of the system (ADoFS) for the three scenarios is captured in Figures 6(a), 6(b), and 6(c). For every request, DoF is computed in every period and then average DoF is computed over its lifetime. Another averaging is done over average DoF of all the requests to obtain ADoFS. For MnAAC and MxAAC, ADoFS is higher in Scenario 2 than in Scenario 1. In Scenario 2, requests have periods equal to an integer fraction of the BI, i.e., they have multiple jobs in a BI and have multiple deadlines to meet. Hence, allocations are more likely to be fragmented. For Scenario 1, since deadlines of requests are integer multiple of BI, there is no fragmentation until the BI utilization approaches 100 % (at λ = 20 requests per BI). At high λ ($\lambda \geq 40$ requests per BI), ADoFS becomes almost flat in all the scenarios. At those λ values, the system accepts very few new requests (see AR performance for these λ values). Hence, the scheduling of jobs of the requests changes very little leading to very little change in ADoFS. In Scenario 1, PFAAC performance is the worst. Since request periods are multiple of BI in this scenario, when a request leaves the system, surplus utilization is created which is added proportionally to the C^{op} of all the existing requests. So, in the next BI some of these requests may get additional fragment of allocation due to this increase in C^{op}. ADoFS performance in Scenario 3 is almost similar to that of Scenario 2 since the majority of requests in Scenario 3, as do requests in Scenario 2, have periods equal to integer fraction of a BI. But at λ value of 20 to 25 requests per BI, where ADoFS of PFAAC goes up slightly (note the log scale of these plots) compared to Scenario 2. Around those λ values, utilization of the system goes towards 100 % (see Figure 4) and the requests, having periods of integer multiple of BI, exhibit high fragmentation under PFAAC (similar to what was seen in Scenario 1).

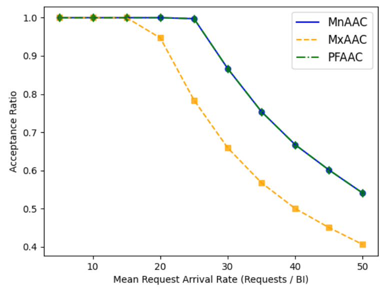
Figures 7(a), 7(b), and 7(c) show average normalized delay (AvND) of requests in the three scenarios as λ increases. To compute the AvND of a given request, normalized delay is computed in every period and averaged over its lifetime. Then AvND of all the requests are plotted as a box and whisker plot. At low λ ($\lambda \leq 15$ requests per BI), median AvND of MxAAC and PFAAC are almost equal to each other, but MnAAC has lower AvND in all the scenarios. At low λ, PFAAC is able to allocate C^{max} and hence, its AvND performance



(a) Scenario 1

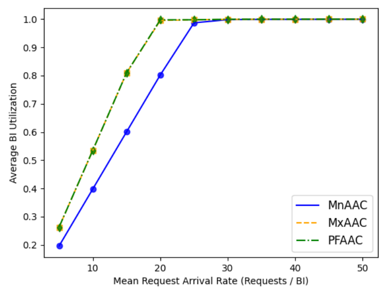


(b) Scenario 2

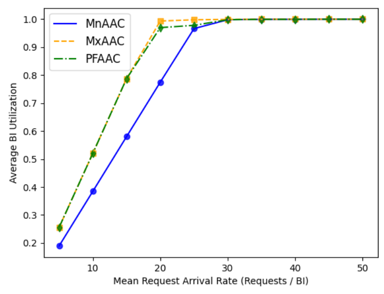


(c) Scenario 3

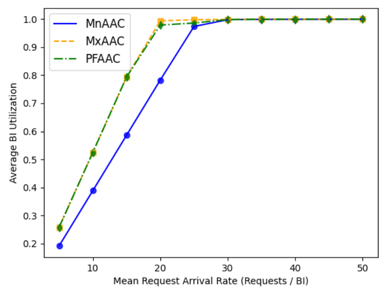
Figure 3: Acceptance Ratio vs. Mean Request Arrival Rate



(a) Scenario 1

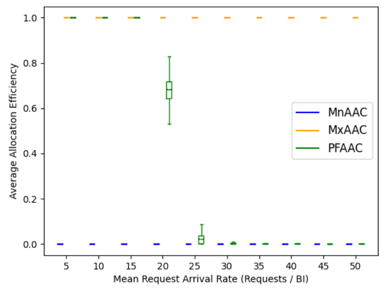


(b) Scenario 2

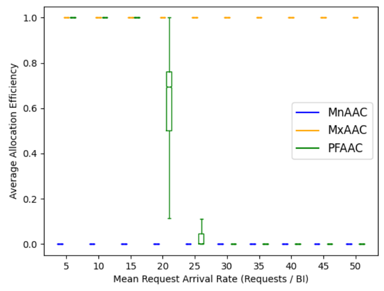


(c) Scenario 3

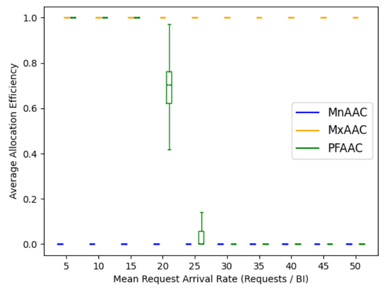
Figure 4: Average BI Utilization vs. Mean Request Arrival Rate



(a) Scenario 1



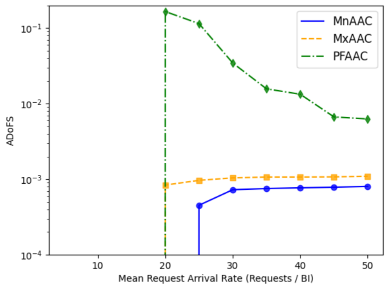
(b) Scenario 2



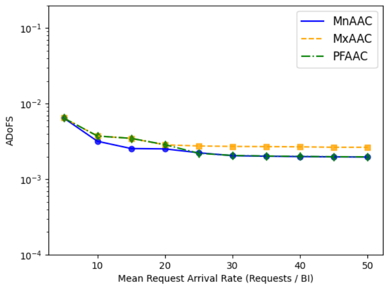
(c) Scenario 3

Figure 5: Average Allocation Efficiency vs. Mean Request Arrival Rate

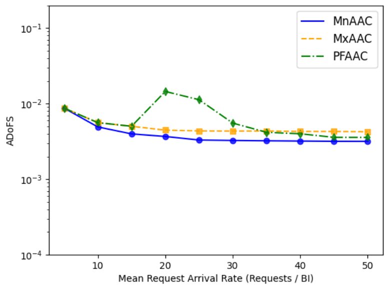
is similar to that of MxAAC. Since PFAAC and MxAAC get more allocation than MnAAC, the end of allocation of a job in those two ACAs is later than in MnAAC which leads to longer delay. At high λ ($\lambda \geq 35$ requests per BI), there is very little difference in the median AvND among the AACs in all the scenarios. At these λ values, the system runs at 100 % utilization and MnAAC and PFAAC have more requests running at C^{min} (each request has shorter delay), whereas MxAAC has less requests, but they are running at C^{max} (each request has longer delay). Hence, the median AvND for those ACAs become almost equal at high λ. At λ value of 20 to 25 requests per BI, PFAAC has higher median AvND than the other two ACAs in Scenario 1. Since requests have periods that are integer multiples of BI, when a request leaves, the C^{op} values of existing requests proportionally increases. Therefore, a request may get additional



(a) Scenario 1

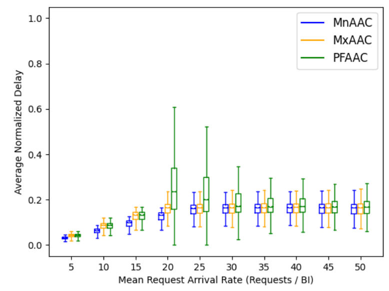


(b) Scenario 2

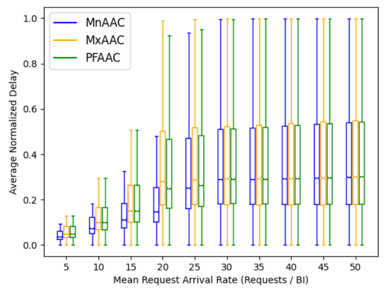


(c) Scenario 3

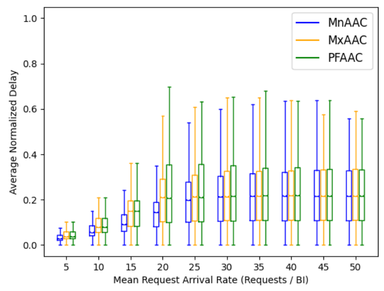
Figure 6: Average Degree of Fragmentation of the System (ADoFS) vs. Mean Request Arrival Rate



(a) Scenario 1



(b) Scenario 2

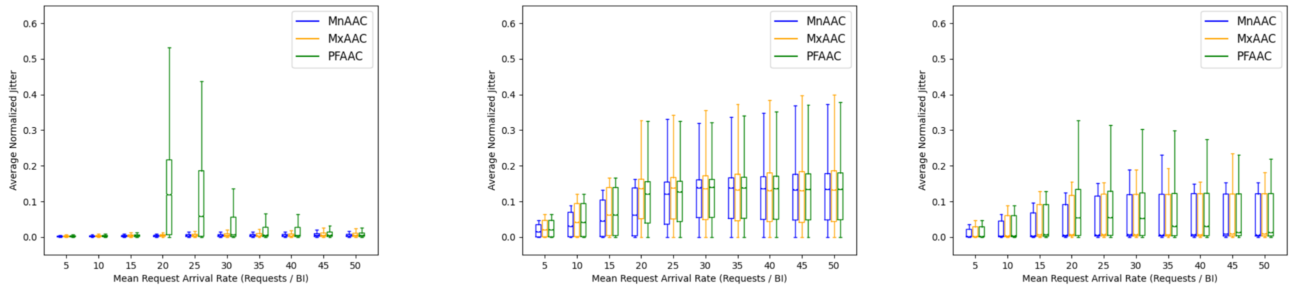


(c) Scenario 3

Figure 7: Average Normalized Delay (AvND) vs. Mean Request Arrival Rate

allocation across BIs and its end of allocation is delayed leading to higher delay. Comparing AvND performance across scenarios, generally Scenario 2 has higher median AvND compared to Scenario 1 for all three ACAs. Since the request periods are a fraction of BI in Scenario 2, they have tighter deadlines compared to Scenario 1. This leads to shuffling of relative allocation positions (in a BI) of jobs when new requests are admitted. In addition, there is more fragmentation in Scenario 2. These two factors lead to higher delay. Relative performance of the ACAs at different λ values in Scenario 3 is very similar to Scenario 2, but influence of requests having periods equal to multiple BI (same as requests in Scenario 1) is visible. In Scenario 1, the interquartile range (i.e., the height of the box) remains almost the same for all three ACAs at high λ and the median value is smaller than in Scenario 2. So, in Scenario 3, at high λ, the heights of the boxes (for all three ACAs) remains almost the same, but they are smaller than Scenario 2 and the median values are also smaller than those in Scenario 2.

Figures 8(a), 8(b), and 8(c) capture performance in terms of average normalized jitter (AvNJ). To compute the AvNJ of a given request, normalized jitter is computed in every period and averaged over its lifetime. Then AvNJ of all the requests are plotted as a box and whisker plot. In Scenario 1, MnAAC and MxAAC have very negligible AvNJ regardless of value of λ. At low λ, BI utilization is low and and hence, the schedule of existing requests do not change due to new requests. Therefore, AvNJ remains close to zero. At high λ, very few new requests are admitted and hence, schedule of existing requests does not change much which leads to low AvNJ. PFAAC also has very low AvNJ except for λ from 20 to 30 requests per BI (when BI utilization starts to rise to 100 %). At these λ values, PFAAC keeps adjusting C^{op} values of existing requests which leads to high jitter. In Scenario 2, as λ increases, median AvNJ increases for all the ACAs. This, as has been pointed out in AvND performance, is due to the shuffling of relative position of allocation and fragmentation of jobs. However, median AvNJ remains almost flat beyond $\lambda = 25$ requests per BI when BI utilization reaches 100 %, because the schedule of existing requests does not change much beyond this value. Comparing AvNJ performance between Scenario 1 and Scenario 2, median values in Scenario‘2 are generally higher than in Scenario 1 for the three ACAs. In Scenario 3, the median AvNJ of MnAAC and MxAAC are almost the same as Scenario 1, but there is more variance due to the mix of requests with periods equal to fraction of BI. For PFAAC in Scenario 3, median AvNJ initially increases, but unlike Scenario 2, after $\lambda = 25$ requests per BI it does not stay constant, but goes down. However, the interquartile range (i.e., box height) remains constant after $\lambda = 25$ requests per BI. This is the effect of existence of requests having periods equal to multiple BIs in Scenario 3. In Scenario 1, for PFAAC, box height decreases at high λ, but the median AvNJ remains close to zero. In



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

Figure 8: Average Normalized Jitter (AvNJ) vs. Mean Request Arrival Rate

Scenario 2, box height and median AvNJ remain constant at high λ. So, if these two effects are combined, then the box height remains constant, but the median decreases, as is seen in Scenario 3.

The experiments were run for a long duration equal to 1000 BIs due to which the run time of the experiments were very high. So, it was not possible to get the uncertainty measurements of AR for every λ value. We measured standard deviation of AR for $\lambda = 5$, 25 and 50 requests per BI for all the three scenarios to get some sample values. The minimum and maximum standard deviation were 0 % and 0.7 % of the mean respectively. Based on these sample values of standard deviation and the fact that our experiments were run for a long duration of 1000 BIs, we expect the standard deviation of AR and average BI utilization for all λ values to be very small.

4 RELATED WORK

There have been few analytical studies done on IEEE 802.11ad channel access. A 3D Markov chain based analytical model for performance analysis of SP and CBAP mode of channel access is presented in [4]. In [12], authors present a Markov chain based analytical model for CBAP allocation. The model accounts for presence of SPs and deafness and hidden node problems associated with directional antennas during CBAP. An analytical model for SP access is proposed in [7], in which the authors study the worst case delay of SP packets using the model. They also discuss a way to optimally allocate a channel between SP and CBAP access. [9] proposes a scheduling method based on analytical model for a multimedia flow using SP channel access in the presence of channel errors.

There has been very little work reported in the literature in experimental study of admission control and scheduling of SP and CBAP allocations. In [10], the authors present two algorithms for joint admission control and scheduling of periodic traffic streams using SP allocation. However, the authors only consider very simple application scenarios. In one scenario, all the applications are assumed to have the same traffic parameters. In the other case, there are only two sets of traffic parameters and an application chooses one from the two sets. Also, this study only considers periods which are integer fraction of the BI. This algorithm can become too expensive if it is to be implemented in a real IEEE 802.11ad system to be able to schedule applications with many different traffic parameters. Finally, in [2], the authors present a reinforcement learning (RL) based scheduling of SP allocation which finds the optimal duration of each SP. The RL based scheme uses Q-learning and interacts with the network deployment scenario to get the optimal SP duration. Queue size, in terms of number of packets, at the MAC layer represents *states* and *reward* is represented as a function of number of received packets and the *action* taken.

5 CONCLUSION AND FUTURE WORK

We presented three fair admission control algorithms and an EDF based scheduling algorithm for isochronous traffic in IEEE 802.11ad MAC. Two of the ACAs, MnAAC and MxAAC, are very simple and can be implemented in constant time. However, their performance is not consistent across different performance metrics discussed in this paper. For example, MnAAC has better performance (than MxAAC) in terms of AR, but is worse in terms of BI utilization and AE. PFAAC provides the best performance among the three ACAs with respect to some performance metrics such as AR and BI utilization. However, for some other performance metrics it may not be the best ACA (e.g., DoF). Thus, an IEEE 802.11ad MAC should choose an appropriate ACA based on its goal of which performance metric it wants to optimize. But, generally PFAAC offers the best tradeoff across different performance metrics. Thus, although its time complexity is higher than the the other two, PFAAC may be used to get overall good performance.

In this work, guard time overhead was not considered. We would like to take this into account in our future work. We are also looking at admission control and scheduling of IEEE 802.11ad MAC when both isochronous and asynchronous traffic are present. We would like to study non-preemptive versions of the EDF scheduler which will have reduced performance in terms of acceptance ratio but should have less overhead in terms of guard time.

REFERENCES

[1] 2016. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 802.11 Working Group of the LAN/MAN Standards Committee of the IEEE Computer Society.

[2] Tommy Azzino, Tanguy Ropitault, and Michele Zorzi. 2020. Scheduling the Data Transmission Interval in IEEE 802.11ad: A Reinforcement Learning Approach. In *2020 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 602–607.

[3] C. L. Liu and J. W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the Association for Computing*

Machinery 20, 1 (January 1973), 46–61.

[4] Qian Chen, Jiqiang Tang, David Tung Chong Wong, Xiaoming Peng, and Youguang Zhang. 2013. Directional Cooperative MAC Protocol Design and Performance Analysis for IEEE 802.11ad WLANs. *IEEE Transactions on Vehicular Technology* 62, 6 (2013), 2667–2677.

[5] Houssine Chetto and Maryline Chetto. 1989. Some Results of the Earliest Deadline Scheduling Algorithm. *IEEE Transactions on software engineering* 15, 10 (1989), 1261–1269.

[6] Yasaman Ghasempour, Claudio RCM da Silva, Carlos Cordeiro, and Edward W Knightly. 2017. IEEE 802.11ay: Next-generation 60 GHz Communication for 100 Gb/s Wi-Fi. *IEEE Communications Magazine* 55, 12 (2017), 186–192.

[7] C Hemanth and TG Venkatesh. 2015. Performance Analysis of Service Periods (SP) of the IEEE 802.11ad Hybrid MAC Protocol. *IEEE Transactions on Mobile Computing* 15, 5 (2015), 1224–1236.

[8] K. Jeffay, D. F. Stanat and C. U. Martel. 1991. On Non-Preemptive Scheduling of Periodic and Sporadic Tasks. In *IEEE Real-Time Systems Symposium (RTSS)*. 129–139.

[9] Evgeny Khorov, Alexander Ivanov, Andrey Lyakhov, and Vitaly Zankin. 2016. Mathematical Model for Scheduling in IEEE 802.11ad Networks. In *2016 9th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 153–160.

[10] Mattia Lecci, Matteo Drago, Andrea Zanella, and Michele Zorzi. 2020. Exploiting Scheduled Access Features of mmWave WLANs for Periodic Traffic Sources. *arXiv preprint arXiv:2011.05045* (2020).

[11] Thomas Nitsche, Carlos Cordeiro, Adriana B Flores, Edward W Knightly, Eldad Perahia, and Joerg C Widmer. 2014. IEEE 802.11ad: Directional 60 GHz Communication for Multi-Gigabit-per-second Wi-Fi. *IEEE Communications Magazine* 52, 12 (2014), 132–141.

[12] Chiara Pielli, Tanguy Ropitault, Nada Golmie, and Michele Zorzi. 2020. An Analytical Model for CBAP Allocations in IEEE 802.11ad. *IEEE Transactions on Communications* (2020).

[13] Sandra R Thuel and John P Lehoczky. 1994. Algorithms for Scheduling Hard Aperiodic Tasks in Fixed-Priority Systems Using Slack Stealing. In *RTSS*. 22–33.