Enhancing the Energy-Efficiency and Timeliness of IoT Communication in WiFi Networks

Jaykumar Sheth and Behnam Dezfouli
Internet of Things Research Lab, Department of Computer Science and Engineering, Santa Clara University, USA
jsheth@scu.edu, bdezfouli@scu.edu

***Abstract*—Increasing the number of IoT stations or regular stations escalates downlink channel access contention and queuing delay, which in turn result in higher energy consumption and longer communication delays with IoT stations. To remedy this problem, this paper presents *Wiotap*, an enhanced WiFi access point that implements a downlink packet scheduling mechanism. In addition to assigning higher priority to IoT traffic compared to regular traffic, the scheduling algorithm computes per-packet priorities to arbitrate the contention between the transmission of IoT packets. This algorithm employs a least-laxity first (LLF) scheme that assigns priorities based on the remaining wake-up time of the destination stations. We used simulation to show the scalability of the proposed system. Our results show that Wiotap achieves 37% improvement regarding the duty cycle of IoT stations compared to a regular access point. In addition, we developed a testbed to confirm the implementation correctness and the performance benefits of Wiotap in a network with four IoT stations and regular traffic. For the edge and cloud scenarios, our empirical evaluations show up to 44% and 38% improvement in energy and 52% and 41% improvement in delay, respectively.**

***Index Terms*—802.11, Scheduling, Prioritization, Scalability, Low-Power, DoS attacks.**

I. INTRODUCTION

The Internet of Things (IoT) is the enabler of applications such as remote medical monitoring, building automation, industrial automation, and smart cities [1]–[3]. Gartner predicts that there will be more than 20 Billion connected devices by 2020 [4]. To offer ease of deployment and support mobility, most of these applications rely on wireless communication.

Various wireless technologies, such as 802.15.4, NB-IoT, LoRa, LTE, and 802.11, are currently being used to connect these devices [5], and a wide range of mechanisms from antenna design [6]–[8] to medium access control [9], [10] to application layer protocols [11] have been proposed to enhance the energy efficiency of these applications. Meanwhile, 802.11 is particularly important due to several reasons: First, 802.11 networks are widely deployed in home, enterprise, and commercial environments. Therefore, for example, 802.11-based smart home systems do not require the installation of additional wireless infrastructure. Second, the power consumption of 802.11 transceivers has been significantly reduced during the recent years. This has been achieved by both new chip manufacturing technologies as well as the various power saving methods proposed for this standard [12]–[18]. Third, compared with cellular networks, 802.11 operates in unlicensed bands, thereby its usage is free of charge. Finally, compared to 802.15 technologies, 802.11 offers considerably higher data rates, which justifies the adoption of this standard in applications such as medical monitoring and industrial control [3], [19]. The higher data rate also reduces the duration of transmissions and increases sleep intervals.

The traffic prioritization and power saving mechanisms of 802.11 fall short in IoT applications. From the timeliness point of view, 802.11 differentiates between real-time (e.g., voice and video) and elastic (e.g., email, file transfer) flows by defining voice, video, best-effort, and background access categories. However, these access categories only accommodate the four traffic categories of *regular* (non-IoT) user devices and cannot differentiate the existence of IoT traffic. Therefore, IoT stations suffer from longer delays and also waste their energy because of idle listening before their downlink packets are delivered. From the energy efficiency point of view, 802.11 introduces power save mechanisms, which require each station to wake up periodically and fetch the buffered packets from the *access point* (AP). In addition, since the traditional *power save mode* (PSM) significantly increases end-to-end delay [20], the *Adaptive PSM* (A-PSM) has been proposed to enable the clients to stay in awake mode and exchange multiple packets with the AP. By utilizing a short *tail time* after each packet exchange, this mechanism prevents the station from having to frequently switch between awake and sleep mode to receive downlink packets when inter-arrival delays are less than the beacon interval. IoT applications can also benefit from this mechanism to reduce the duration of their transactions. For example, assume a medical IoT device detects an anomaly, reports the data to a server, and performs some actions based on the commands received. In this case, the tail time can be exploited to reduce the delay of this transaction. Consequently, we observe that the state-of-the-art low-power 802.11 transceivers support this mode [15], [21], [22]. However, the tail time duration might also result in a higher energy consumption if the AP's downlink traffic is high and packet transmissions are not properly scheduled based on energy efficiency concerns. In other words, when the number of regular or IoT stations increases, the amount of time spent in tail time increases as well without positively affecting timeliness or energy efficiency.

In this paper, we propose *Wiotap* (WiFi IoT AP) to address the problem of joint channel access and power saving in 802.11-based networks. We assume that the network includes *regular user stations* (such as smartphones and laptops) as well as *IoT stations* (such as battery powered medical monitoring devices). When a large number of IoT stations exist in the network, Wiotap significantly enhances energy efficiency by applying per-packet scheduling. Also, Wiotap reduces the negative impact of regular traffic on the energy efficiency of IoT stations. Specifically, the contributions of this paper are as follows:

– We propose a queue allocation algorithm, which allocates the number, priority, and service rate of a set of queues dedicated to IoT traffic. The proposed algorithm allocates deadline guarantees to each queue based on the distribution of tolerable delay values collected during a time window. To enforce the delivery delay guarantee of queues, we employ a time-division-based traffic shaping approach to control the service rate of the queues.
– We propose a least-laxity first (LLF) packet scheduling mechanism based on the tolerable delay of each packet before the expiry of the destination station's tail time. Once a packet arrives on the AP's wired interface, if the destination station is awake, the AP assigns a priority to the packet and pushes it into the selected IoT queue. The assigned priority depends on the deadline of this packet relative to the deadline of all the awake IoT stations. This mechanism reduces the waiting time of IoT stations and increases the number of packets exchanged during a wake-up period. If the AP determines that a packet cannot be delivered to its destination station before shifting into sleep mode, a priority that is higher than that of regular stations' packets is assigned to the IoT packet. This mechanism expedites the delivery of packets to IoT stations after beacon reception during the next wake-up period.
– Since Wiotap's per-packet scheduling is especially applicable to large-scale deployments, we implement a simulation tool using the OMNet++ simulation framework. Compared to a regular AP, our solution provides an average performance improvement of 37%. Additionally, increasing the number of queues dedicated to IoT traffic from 2 to 4 can further reduce the duty cycle of IoT stations by 28%.
– We implement Wiotap by adding kernel space and user-space components to a Linux AP. One of the salient features of Wiotap is its independence from the MAC layer. This feature simplifies the adoption of Wiotap irrespective of the 802.11 NIC used. In addition to the simple loading of user-space modules, at the network layer we utilize a modified version of the default PRIO qdisc kernel module. This module can also be loaded dynamically during runtime.
– To confirm the implementation correctness and performance benefits of Wiotap even when the number of IoT stations is not high, we implement a testbed using four IoT stations and different types of regular traffic generators.
IoT stations use the Message Queuing Telemetry Transport (MQTT) protocol to report events to the broker and receive a response back. To show the impact of server location on energy efficiency, we change the location of MQTT server to represent edge and cloud scenarios. The empirical results confirm that Wiotap reduces delay and energy consumption by up to 52% and 44% in the edge scenario and 41% and 38% in the cloud scenario.

The remainder of this paper is organized as follows: Section II introduces power-save modes, traffic prioritization in 802.11 networks, and the employed system model. Section III presents Wiotap. The implementation details are explained in Section IV. Sections V and VI present simulation and empirical performance evaluations. In Section VII, we review the related work. Finally, Section VIII concludes the paper and presents future directions.

II. Background and System Overview

In this section, we first present an overview of 802.11 in terms of energy saving and traffic scheduling. We then present our system overview.

A. 802.11 Power Saving Mechanisms

802.11's PSM is one of the most integral aspects of energy efficiency. PSM and its variants [20], [23]–[27] reduce idle listening during inactivity times by switching into a low-power mode. The two main PSM techniques are: *Legacy PSM* (a.k.a., Static PSM), and *Adaptive PSM* (A-PSM) (a.k.a., Dynamic PSM). In the former, each station periodically wakes up to receive the beacons sent by the AP. By inspecting the *traffic indication map* (TIM) embedded in beacon packets, the station checks if the AP has buffered packets. In case the TIM for the station was set, the station sends a PS-Poll frame to the AP to fetch the buffered packets. The station enters the sleep mode when it receives a data frame with a cleared *more* data flag, which indicates no more packets are buffered in the AP.

When using A-PSM, the station does not immediately switch to the sleep mode after receiving the last buffered packet. Instead, it waits for a *tail time* duration (denoted by Γ), expecting that another packet will soon follow [28], [29]. At the end of the tail time and before transitioning into sleep mode, the station sends a QoS NULL frame to the AP with the power management bit set. In addition to enhancing the timeliness of interactions with IoT stations, this mechanism also reduces the overhead of radio switching. For example, assume an industrial monitoring and control device detects an anomaly, reports the event, and waits for the reception of a command to perform a proper action. In this case, using the tail time enables the devices to connect, report, and receive the commands promptly. Due to these benefits, the state-of-the-art 802.11 transceivers, such as CYW43907 [16] and BCM4343W [15], support A-PSM.

B. Traffic Prioritization

Since 802.11 networks are used for the exchange of both elastic and real-time data, the 802.11e standard provides various *access categories* (AC) for *voice* (AC_VO), *video* (AC_VD), *best effort* (AC_BE), and *background* (AC_BK) traffic flows [30]–[32]. The AC of MAC frames is determined based on the *differentiated services code point* (DSCP) field[1] of IP header. This field comprises of different flagged bits, which when set, conveys to the lower layers the flow type and enforces IP precedence for per-hop QoS and priority. This layer-3 field is then mapped to the *class of service* (CoS) field in the MAC header [33]. By using CoS mapping,

[1]This field is also known as the *type of service* (ToS).

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Meaning
s_i^{iot}	An IoT station
s_i^{reg}	A regular station
$\mathbf{Q}_{net}^{iot}$	Set of IoT queues (in the qdisc layer)
$\mathbf{Q}_{net}^{reg}$	Set of regular queues (in the qdisc layer)
Q_i	An IoT queue
η	Number of IoT queues
Γ	Tail time duration
$\tilde{t}$	Last activity time of a station
$\Delta(p_i)$	Deadline of packet p_i
$\boldsymbol{\Delta}$	A circular queue holding $D(p_i)$ values
$\mathcal{M}(Q_i)$	Maximum tolerable deadline of queue Q_i
$\mathcal{D}(Q_i)$	Duration of transmitting packets currently in Q_i
$\mathcal{S}(Q_i)$	Packets serviced by queue Q_i during service period
$\bar{\mathcal{S}}(Q_i)$	Service size of queue Q_i
$\mathbf{Tx}$	A circular queue holding packet transmission delays
$\mu_{\mathbf{Tx}}$	Average packet transmission duration

the kernel sets the priority socket buffer and enqueues the packet to the corresponding transmit queue. In the 802.11e protocol, each *enhanced distributed channel access* (EDCA) queue behaves as a virtual station and contends for the channel independently according to the contention parameters. It is not a regular practice to implement new ACs or modify the EDCA contention parameters because the 802.11e standard specifies them. To this end, to accommodate the QoS of various scenarios, almost all Linux systems implement queuing disciplines, which are known as *qdisc*. qdisc provides several types of traffic scheduling (what packet to send) and traffic shaping (how many packets to be sent per time unit) mechanisms, which are applied before forwarding packets to layer-2.

C. System Model

The system is composed of an AP and *stations*. There are two types of stations: (i) *regular stations*, denoted by s_i^{reg}, such as smartphones and laptops, and (ii) *IoT stations*, denoted by s_i^{iot}, such as medical IoT devices or industrial robot arms, which perform machine-to-machine communication. Although regular stations might exchange voice and video traffic with the AP, we assume that IoT stations are resource-constrained and therefore, preserving their energy resources has a higher priority compared to regular stations. Besides, even if the network only includes IoT stations, we are interested in reducing the energy consumption of all the stations. Once a station exchanges a packet with the AP, it stays awake for a tail time duration Γ. Table I summarizes the notations used in this paper.

III. Scheduling Mechanism

Figure 1 shows the queue allocation scheme. On top of the MAC layer, we introduce *IoT queues* in addition to the *regular* queues, in the qdisc layer. Once a packet for a regular (non-IoT station) arrives on the wired interface of AP, it is pushed into one of the regular queues, i.e., $\mathbf{Q}_{net}^{reg} = \{Q_{vo}, Q_{vi}, Q_{be}, Q_{bk}\}$, depending on the packet's ToS field. Packets destined to IoT stations are pushed into one of the IoT queues, i.e.,

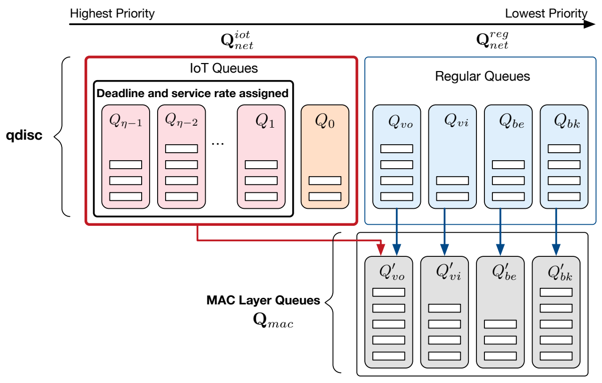

Fig. 1. The proposed scheduling algorithm introduces IoT queues in addition to the regular queues in qdisc layer. The IoT queues are configured based on the delay distribution of IoT packets.

$\mathbf{Q}_{net}^{iot} = \{Q_0, Q_1, ..., Q_{\eta-1}\}$, to accelerate the transmission of these packets. We refer to Q_0 as the *base queue* and Q_1 to $Q_{\eta-1}$ as the *prioritized queues*. The queue selection process is based on a scheduling algorithm that we will present later on. The qdisc packets are assigned to MAC layer queues, i.e., $\mathbf{Q}_{mac} = \{Q'_{vo}, Q'_{vi}, Q'_{be}, Q'_{bk}\}$, based on their priority, as indicated by the arrows in Figure 1.

The basic idea of transmission scheduling is to prioritize the packets of IoT stations in order to reduce idle listening time and number of sleep/wake transitions of these stations. When a packet belonging to an IoT station arrives, if the destination is awake, we evaluate the time left before the station enters sleep mode. The packet is accelerated for delivery if the remaining duration is longer than a threshold. The acceleration algorithm tries to maximize the chance of packet delivery to all IoT stations by taking into account the relative priority of all the packets' deadlines. The rest of this section explains these operations in detail.

A. Acceleration Eligibility

A packet is eligible for acceleration if its destination IoT station is still awake and acceleration results in a packet delivery before the end of tail time. To this end, it is essential to keep track of the operational status of all IoT stations and determine the remaining tail time of awake stations. To determine the remaining duration, however, it is not possible to rely on the sleep/wake schedule of stations because the tail time is renewed every time a packet is exchanged with the AP. Therefore, it is necessary to record the time stamp of the last packet exchanged with the AP. Section IV will present the implementation details.

Assume a packet p_j belonging to an IoT station s_i^{iot} arrives at time t. This packet is eligible for acceleration if:

$$\Delta(p_j) = \Gamma(s_i^{iot}) - (t - \tilde{t}(s_i^{iot})) > Th(s_i^{iot}) \quad (1)$$

where $\Delta(p_j)$ reflects the *delivery laxity* of packet p_j upon its arrival, $\Gamma(s_i^{iot})$ is the tail time of the station, t is the current time, and $\tilde{t}(s_i^{iot})$ is the last time the station has exchanged a packet with the AP. In addition to decision making about the acceleration of incoming packets, the computed Δ values are also used for queue configuration. We will present *Queue Configuration* algorithm in Section III-B. If Inequality 1 holds for an incoming packet, then the *Enqueue* algorithm tries to accelerate the transmission of this packet by determining a suitable prioritized queue within the set $\{Q_1, Q_2, ..., Q_{\eta-1}\}$. We will present this algorithm in Section III-C. If Inequality 1 does not hold, then the packet is pushed into *base queue* Q_0 to increase its priority compared to regular traffic. The base queue expedites the delivery of this packet compared to regular traffic during the next wake up time.

The threshold value $Th(s_i^{iot})$ depends on channel access contention, physical layer rate, link reliability between the AP and the station, and the rate of regular traffic. For example, if multiple transmissions are required to reach the station, then the threshold value should be long enough to account for the retransmissions. Since all the qdisc IoT queues are mapped to MAC layer's queue Q'_{vo}, to measure these delays, we record the time interval between the instance an IoT packet arrives in Q'_{vo} until its successful delivery to the destination node. These values are stored in a circular array denoted as $\mathbf{Tx}$.

B. Queue Configuration

In this section we present the operation of Queue Configuration algorithm, which assigns priorities to the IoT queues to enable fine-grained prioritization of IoT packets.

$\mathbf{Q}_{net}^{iot} = \{Q_0, Q_1, ..., Q_{\eta-1}\}$ is the set of queues devoted to IoT packets. Queue Q_0 is used to prioritize IoT packets that are not acceleration eligible, and the rest of the queues are configured by Queue Configuration algorithm to offer deadline-aware packet acceleration. Associated with each prioritized queue $Q_i \in \{Q_1, ..., Q_{\eta-1}\}$ is a *maximum tolerable delay* (MTD) value, denoted as $\mathcal{M}(Q_i)$. The MTD of a queue reflects the maximum potential delay experienced by the packets of that queue until transmission. MTD assignment to queues ensures that the deadline of the packets buffered in each queue satisfy their delivery deadline requirement.

The purpose of queue configuration is to assign an MTD to each IoT queue based on the distribution of packet laxities computed by Equation 1. Whenever a new Δ value is computed for an incoming IoT packet, the value is inserted into a circular queue denoted as $\boldsymbol{\Delta}$. The main idea is to divide the range of deadlines into two equal intervals and configure the MTD of queues based on the number of deadline entries that fall in each interval. Each interval is then broken into two more intervals, and the same process is repeated until each queue is assigned an MTD value.

The Queue Configuration algorithm is presented in Algorithm 1. The high level function `queue_conf()` computes the minimum and maximum of the laxity values stored in the circular queue $\boldsymbol{\Delta}$. These values, in addition to the minimum and maximum index of the IoT queues, are passed to function `qc()`, which is recursively called to assign MTDs to queues. The number of queues assigned to the left and right side of the queue corresponds to the distribution of laxity values around the mid value. If the number of queues assigned to the left side is equal to one, then the MTD of that queue is equal to the Δ_{mid} of that iteration. If the number of assigned queues is more than one, then the function is recursively called to configure left side queues. Queue allocation to the right side is performed similarly. If the number of queues assigned to the right side is one, then the MTD of that queue is equal to the Δ_{max} of that iteration. The time complexity of Queue Configuration algorithm is $O(n)$ because the division process continues until all the queues are configured individually.

C. Enqueue Algorithm

A packet p_j satisfying Inequality 1 could be assigned to a prioritized IoT queue if the following condition is met,

$$\exists i \in [1, \eta - 1] \ \mid \ \Delta(p_j) \leq \mathcal{M}(Q_i). \tag{2}$$

If none of the IoT queues can satisfy the above condition, then the packet is inserted into the base queue Q_0. If Condition 2 is satisfied, then a *least-laxity first* (LLF) scheduling strategy is employed to assign packets to the prioritized queues. To this end, the packet is added to the queue with the highest MTD value that can satisfy the delivery deadline. In other words, the index of the eligible queue, denoted as i, is found as follows:

$$\underset{i \in [1, \eta-1]}{\operatorname{argmin}} \big(\Delta(p_j) \leq \mathcal{M}(Q_i)\big). \tag{3}$$

However, it should be noted that the deadline of a queue does not only reflect the transmission duration of the packets in that queue. Instead, for each Q_i, its deadline is the deadline of its next higher priority queue (i.e., Q_{i+1}) plus the duration required to transmit the packets in Q_i. Therefore, to guarantee a maximum transmission delay for the packets of each queue, the following inequality must be true,

$$\mathcal{M}(Q_{i+1}) + \mathcal{D}(Q_i) < \mathcal{M}(Q_i), \quad 1 < i < \eta - 2 \tag{4}$$

where $\mathcal{D}(Q_j)$ is the delay of transmitting the packets in queue Q_j. However, to ensure the validity of the above inequality, it is essential to limit the number of packets serviced by each queue per time unit. We employ a time-division-based mechanism to satisfy this requirement. For each queue $i \in [1, \eta - 1]$ we define its *service capacity* as

$$\bar{\mathcal{S}}(Q_i) = \lfloor (\mathcal{M}(Q_i) - \mathcal{M}(Q_{i+1}))/\mu_{\mathbf{Tx}} \rfloor \tag{5}$$

where $\mu_{\mathbf{Tx}}$ is the average of the values stored in the circular array $\mathbf{Tx}$. In other words, service capacity represents the number of serviceable packets per *service period.* Since for the highest priority queue

$$\bar{\mathcal{S}}(Q_{\eta-1}) = \lfloor \mathcal{M}(Q_{\eta-1})/\mu_{\mathbf{Tx}} \rfloor , \tag{6}$$

the service period is defined as the MTD of the lowest priority queue, i.e., $\mathcal{M}(Q_1)$. To enforce service capacities, in addition to the $\bar{\mathcal{S}}(Q_i)$ values assigned to each queue, the number of packets that have been added to each queue during the current service period is maintained by the Enqueue algorithm. For each queue Q_i this value is denoted by $\mathcal{S}(Q_i)$. No more packets are inserted into a queue Q_i during a service period if $\mathcal{S}(Q_i) \geq \bar{\mathcal{S}}(Q_i)$. The $\mathcal{S}(Q_i)$ counters are reset at the beginning of each $\mathcal{M}(Q_1)$ interval.

Enforcing service capacity imposes another limitation on finding an appropriate queue for an acceleration-eligible

Algorithm 1: Queue Configuration Algorithm

```
Inputs:
    $\mathbf{\Delta}$: circular queue holding $D(p_i)$ values;
    $\mathbf{Q}^{iot}_{net}$: set of IoT queues;
Output: allocates a maximum tolerable deadline to each IoT queue
(i.e., assigns $\mathcal{M}(Q_i) \in \mathbf{Q}^{iot}_{net}$)

function queue_conf()
    $\eta$: number of IoT queues $\mathbf{\Delta} = \{\Delta_i\}_{i=1}^{N}$ ;
    $\Delta_{min} = \min_i \Delta_i$;
    $\Delta_{max} = \max_i \Delta_i$;
    $I_{max} = \eta - 1$;
    $I_{min} = 0$;
    qc($\Delta_{min}$, $\Delta_{max}$, $I_{min}$, $I_{max}$);

function qc($\Delta_{min}$, $\Delta_{max}$ , $I_{min}$, $I_{max}$)
    $\Delta_{mid} = \Delta_{min} + (\Delta_{max} - \Delta_{min})/2$;
    $q = I_{max} - I_{min} + 1$;
    $W_{left} = 0$;
    $W_{right} = 0$;
    for every entry $\Delta_i$ in $\mathbf{\Delta}$ do
        if $\Delta_i \geq \Delta_{min}$ and $\Delta_i \leq \Delta_{mid}$ then
            $W_{left} + +$;
        if $\Delta_i > \Delta_{mid}$ and $\Delta_i \leq \Delta_{max}$ then
            $W_{right} + +$;
    $W_{total} = W_{left} + W_{right}$;
    $W_{left} = W_{left}/W_{total}$;
    $W_{right} = W_{right}/W_{total}$;
    $W'_{left} = W_{left} \times q$;
    $W'_{right} = W_{right} \times q$;
    $W'_{left} = \lfloor W'_{left} + 0.5 \rfloor$;
    $W'_{right} = \lfloor W'_{right} + 0.5 \rfloor$;
    if $W'_{left} + W'_{right} == q + 1$ then
        randomly reduce the number of left or right queues by one;
    /*allocate queue to the values less than or
    equal to $\Delta_{mid}$/*
    if $W'_{left} == 1$ then
        $\mathcal{M}(Q_{\eta-1}) = \Delta_{mid}$;
    else if $W'_{left} > 1$ then
        $left\_end = \eta - 1$;
        $left\_start = \eta - W'_{left} + 1$;
        qc($\Delta_{min}$, $\Delta_{mid}$, $left\_start$, $left\_end$);
    /*allocate queue to the values greater than
    $\Delta_{mid}$/*
    if $W'_{right} == 1$ then
        $\mathcal{M}(Q_{I_{min}}) = \Delta_{max}$;
    else if $W'_{right} > 1$ then
        $left\_start = \eta - W'_{left}$;
        $right\_end = left\_start - 1$;
        $right\_start = left\_start - right\_queues$;
        qc($\Delta_{mid}$, $\Delta_{max}$, $right\_start$, $right\_end$);
```

packet. It is possible that the MTD of a packet satisfies the incoming packet's delivery deadline, but the queue has exceeded the maximum number of allowable insertions in the current service period. Formally speaking, the index i returned by formula 3 does not satisfy Equation 5. In this case, the search for an appropriate queue continues towards the higher priority queues. Therefore, to ensure a sustained guarantee of delivery deadlines, Condition 3 must be revised as follows,

$$\underset{i\in[1,\eta-1]}{\text{argmin}}\big(\Delta(p_j) \leq \mathcal{M}(Q_i) \ \text{ and } \ \mathcal{S}(Q_i) < \bar{\mathcal{S}}(Q_i)\big). \qquad (7)$$

Algorithm 2 presents the packet enqueue process. For each incoming packet (over the wired interface), this algorithm tries to find a prioritized queue if the packet deadline is higher than the threshold we discussed in Inequality 1 (cf. Line 2). The algorithm pushes the packet into the base queue if the packet deadline is longer than the MTD of the lowest priority queue (cf. Line 2). Please note that the service capacity of this queue is not taken into account since there is no deadline guarantee offered by this queue. If the packet has not been inserted into the base queue, the algorithm verifies if the deadline is lower than that of the highest priority queue. In this case, the packet is pushed into the queue if the queue is capable of serving more packets during the current service period (cf. Line 2). Otherwise, the packet is inserted into the base queue (cf. Line 2) since it is evident that the other queues cannot satisfy the deadline requirement of this packet. If none of the above two boundary cases hold, the algorithm tries to find a queue that satisfies the deadline requirement and is capable of serving more packets during the current service period (cf. Line 2). Please note that, since the packet service rate of each queue is limited, after the completion of this loop, the algorithm does not necessarily push the packet into the lowest priority queue that can satisfy the packet deadline. In the end, if no prioritized queue was found, the packet is inserted into the base queue.

The time complexity of this algorithm is $O(n)$ because, if the boundary values do not hold, the algorithm needs to evaluate the eligibility of all the queues.

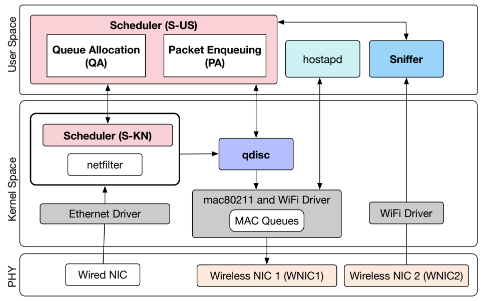

Fig. 2. Implementation architecture of Wiotap on a Linux AP.

IV. Implementation

Figure 2 presents the implementation architecture of Wiotap in Linux-based APs. This architecture is composed of four main modules: (i) *Scheduler module*, which includes a kernel-space sub-module (S-KN), and a user-space sub-module (S-US), (ii) *WiFi Logger* (WiLog) module, and (iii) *qdisc module*.

The S-US module runs the Queue Configuration and Enqueue algorithms. To ensure proper mapping to MAC layer queues, the S-KN module modifies the ToS field in the IP header of IoT packets according to the queue number specified by S-US. The WiLog module keeps track of the operational status of IoT stations. Finally, the ToS field set by the S-KN

Algorithm 2: Enqueue Algorithm

```
Inputs:
  $p_j$: an incoming packet;
  $\mathbf{Q}_{net}^{iot}$: set of IoT queues;
Output: the input packet has been inserted into a queue;

function enqueue($p_i$)
    $s_i^{iot}$ = destination of packet $p_j$;
    $\Delta(p_j) = \Gamma(s_i^{iot}) - (t - \tilde{t}(s_i^{iot}))$;
    if $\Delta(p_j) > Th(s_i^{iot})$ then
        if $\Delta(p_j) > \mathcal{M}(Q_1)$ then
            insert $p_j$ into $Q_0$;
            return;
        else if $\Delta(p_j) \leq \mathcal{M}(Q_{\eta-1})$ then
            if $\mathcal{S}(Q_{\eta-1}) < \bar{\mathcal{S}}(Q_{\eta-1})$ then
                $\mathcal{S}(Q_{\eta-1})++$;
                insert $p_j$ into $Q_{\eta-1}$;
                return;
            else
                insert $p_j$ into $Q_0$;
                return;
        else
            for $k = 1$ to $\eta - 1$ do
                if $\Delta(p_j) < \mathcal{M}(Q_k)$ then
                    if $\mathcal{S}(Q_k) < \bar{\mathcal{S}}(Q_k)$ then
                        $\mathcal{S}(Q_k)++$;
                        insert $p_j$ into $Q_k$;
                        return;
    insert $p_j$ into $Q_0$;
    return;
```

module is used by the qdisc kernel module to push the packet into the appropriate IoT queue.

The Wiotap system also includes the `hostapd` daemon [34], which is a user-space software to perform regular AP functionalities such as authentication, association, and beaconing. This daemon communicates with `cfg80211` through `nl80211`.

A. WiFi Logger Module (WiLog)

The proposed scheduling mechanism requires knowledge about the current status of IoT stations regarding their sleep/wake status and the remaining tail duration of awake stations. The WiLog module is responsible for providing the S-US module with this information. To this end, Wiotap is equipped with one additional wireless NIC to overhear all the packets (data and NULL) and collect the required information. Also, this module includes the circular array $\mathbf{Tx}$ explained in Section III-A. The capacity of this array is 100 in our implementation.

B. Scheduler Module

The scheduler is implemented in two parts: user space and kernel space, which are referred to as S-US and S-KN modules, respectively. To avoid floating-point calculations in the kernel space [35], the S-US module, which includes the Queue Configuration and Enqueue algorithms, is implemented in the user space. The S-KN module uses the netfilter [36] components located in the Linux kernel to grab the packets arriving on the Ethernet interface. Specifically, we use the `PREROUTING` mode, where all the packets are intercepted before the routing decision is made. If the destination of an arriving packet is an IoT station, the S-KN module requests the S-US module to determine the most appropriate queue for this packet. S-US collects $\tilde{t}(s_i^{iot})$ and $\mu_{\mathbf{Tx}}$ from the WiLog module. After determining a queue, S-US instructs S-KN to modify the IP header of the packet to reflect the new ToS value determined. Then, S-KN recalculates the checksum and passes the packet to the qdisc module to place it in the proper queue. The circular queue $\mathbf{\Delta}$ used to hold the packet laxity values is implemented in the S-US module. The size of this queue is 100 in our implementation.

C. qdisc Module

By default, every network interface is assigned a `pfifo_fast` as its transmit queuing mechanism [37]–[39]. `pfifo_fast` implements a simple three-band prioritization scheme based on the 8-bit ToS field in the IP header [40]. FIFO rules apply to the packets in each band. Also, as long as packets are waiting in band 0, band 1 cannot be processed. Similarly, band 1 has a higher priority compared to band 2. Hence, `pfifo_fast` always processes band-0's traffic regardless of the number and rates of contending flows [37]. `PRIO` qdisc is a classfull-configurable alternative of `pfifo_fast` and enables users to configure the number of queues/bands. In this work, we use a modified version of `PRIO` qdisc. To establish a mapping between the ToS field and queues, we have used `priomap` to associate the ToS values to the bands. The implemented `PRIO` queuing discipline can provide up to $n+4$ queues in the qdisc layer, where n queues are used for IoT stations and four queues for the regular stations. Also, the per-queue statistics (number of packets in each queue) is maintained by the `PRIO` qdisc kernel module and communicated to S-KN module through netlink sockets.

V. Simulation Results

Since the performance enhancement achieved by Wiotap is particularly revealed in scenarios including a large number of IoT stations, we present simulation results first and postpone empirical evaluations to Section VI. To this end, we have developed a simulation tool using the OMNet++ simulation framework [41]. The AP is placed in the center of a 50m$\times$50m area and regular and IoT stations are randomly placed in this area. The IoT stations are normally in sleep mode. Each IoT station wakes up every 4 seconds and reports an event to a server. The response of the server is a reply to the station. We refer to this process as a *transaction*. The tail time duration is 10ms, and beacon interval is 100ms. *Regular traffic* intensity refers to the percentage of AP bandwidth utilized by regular stations. When this percentage is between 95% to 99%, we refer to it as near-saturation (*NSat*).

We compare the performance of Wiotap versus two baselines: (i) *R-AP*: a regular AP that does not perform any

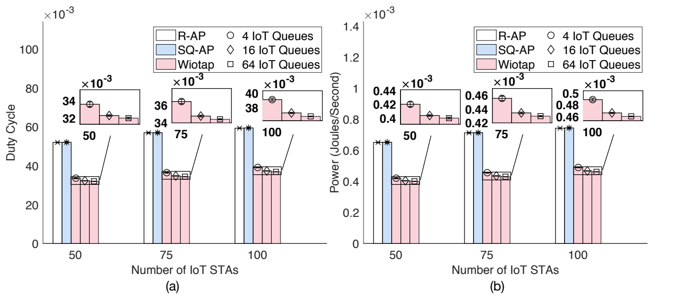
Fig. 3. Impact of the number of IoT stations on (a) average duty cycle and (b) average energy consumption of IoT stations. This experiment does not include any regular station.

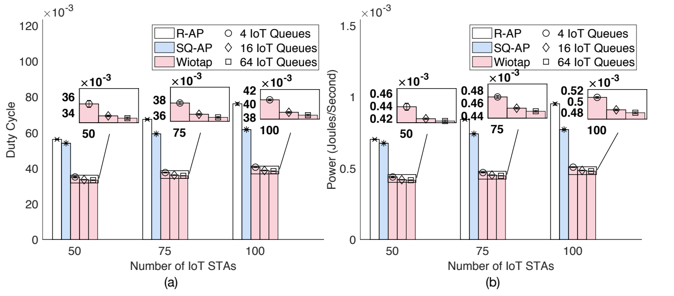
Fig. 4. Impact of the number of IoT stations on (a) average duty cycle and (b) average energy consumption of IoT stations. The type and intensity of regular traffic are AC_BK and 75%, respectively.

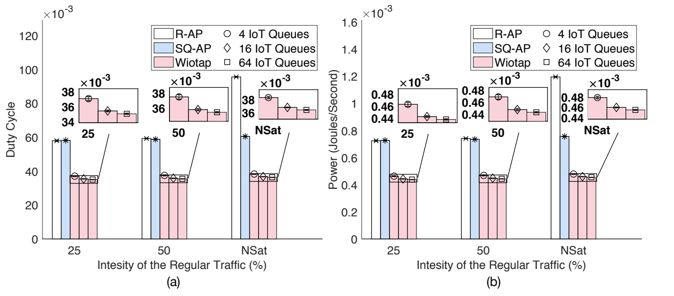
Fig. 5. Impact of regular traffic on (a) average duty cycle and (b) average energy consumption of IoT stations. The number of IoT stations is 75. The regular traffic type is AC_BK.

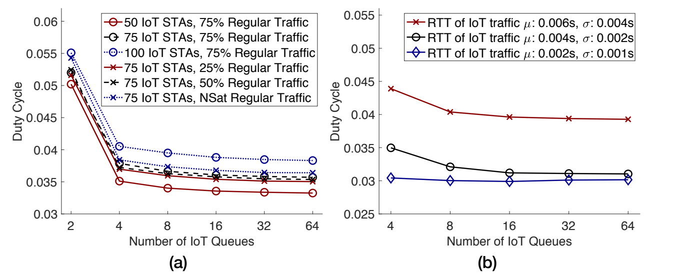
Fig. 6. Impact of the number of IoT queues on the duty cycle of IoT stations. (a): Duty cycle versus the number of IoT stations and regular traffic. (b): Duty cycle versus RTT delay in the presence of 100 IoT stations and no regular traffic.

IoT flow prioritization, and (ii) *SQ-AP*: a single-queue IoT prioritization mechanism that pushes IoT packets in the base queue Q_0. Please note that the latter baseline employs a simple FIFO scheduling of IoT traffic and does not offer deadline-based intra-IoT traffic prioritization. Each point shows the median of 50 experiments where each experiment includes 30 transactions. Error bars demonstrate upper and lower quartiles.

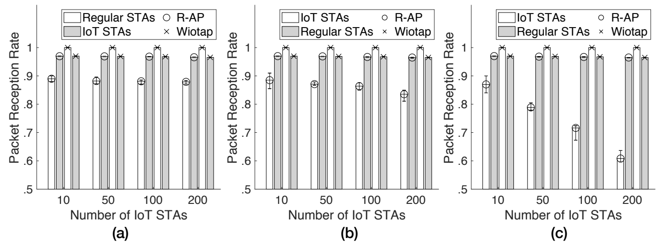
Fig. 7. Impact of packet scheduling on delivery rate. IoT stations generate uplink traffic every 4 seconds using a uniform distribution within the following intervals: (a) [0, 4s], (b) [0, 1s], (c) [0, 100ms], in a uniform manner.

Figure 3 shows the duty cycle and energy consumption rate per IoT station versus network density. Even when the number of stations is 50, the duty cycle achieved with Wiotap is 37% less than that of R-AP and SQ-AP. Please note that in this figure, the results achieved with R-AP and SQ-AP are similar because there is no difference between assigning IoT packets to Q_0 or a regular queue (i.e., Q^{reg}_{net}) when no regular traffic is present. Therefore, these results reveal the main advantage of Wiotap regarding deadline-aware packet scheduling in large-scale IoT networks by using multiple IoT queues and assigning per-packet priority levels among the IoT stations.

In the next experiment, we keep the regular traffic level at 75% and increase the number of IoT stations. Figure 4 presents the results. Compared to R-AP and SQ-AP, Wiotap shows 39% and 37% reduction in duty cycle when the number of IoT stations is 50, respectively. Also, when the number of IoT stations is increased from 50 to 100, R-AP and Wiotap show around 35% and 15% increase in duty cycle, respectively. Although SQ-AP shows lower duty cycle compared to R-AP by prioritizing IoT packets over regular traffic, its performance is significantly lower than that of Wiotap since it does not perform deadline-based scheduling.

Figure 5 depicts the performance improvement achieved by Wiotap versus the intensity of regular traffic when 75 IoT stations exist in the network. The average performance improvement in the presence of 25% regular traffic compared to R-AP and SQ-AP are 39% and 38%, respectively. By "average" we refer to the performance improvement of the proposed approach when using 4, 16, and 64 IoT queues compared to the baselines. Also, in the presence of NSat regular traffic, the average performance improvement compared to R-AP and SQ-AP are 61% and 38%, respectively. These results, in particular, show the impact of increasing regular traffic on the energy efficiency of IoT stations when R-AP or SQ-AP are used. For example, when using R-AP, increasing regular traffic from 25% to NSat increases the duty cycle of IoT stations by 65%. Even in high-capacity networks with a few regular stations, an attacker might generate a DoS attack to saturate the network and compromise the energy efficiency of IoT stations.

Figure 6(a) demonstrates how the number of queues affects the average duty cycle of IoT stations. We observe a decaying trend in duty cycle for all the test conditions because increasing the number of IoT queues enhances the granularity of IoT traffic prioritization. This figure shows that increasing the number of IoT queues from 2 to 4 results in 28% lower duty cycle on average for all the cases. After that, the duty cycle is reduced by around 5% when the number of queues is increased to 64. This behavior is because not all the queues available would be fully utilized as the queue filling rate depends on the amount of concurrent traffic. In general, certain conditions raise the importance of higher-granularity packet scheduling: increasing the number and traffic rate of IoT stations, increasing the standard deviation of RTT, various tail time values of IoT stations. As Figure 6(b) shows, increasing the mean and standard deviation of RTT enhances the benefits of using more number of queues.

Figure 7 shows the impact of packet scheduling on delivery rate. The delivery rate of a station is defined as the ratio of the packets delivered to a station by the AP over the number of packets received by the AP on its wired interface. When using R-AP, both regular and IoT traffic are assigned a similar priority and share the same set of regular qdisc queues. In this case, the impact of buffering on IoT traffic is similar to that of regular queues if the IoT transactions are uniformly distributed over time. In scenarios where multiple devices detect and report an event during a short duration, uplink and downlink communication are accumulated, and therefore, a higher packet loss rate occurs due to buffer overflow. In contrast, Wiotap schedules IoT packets to be delivered before regular packets. More importantly, since Wiotap tries to deliver IoT packets before these stations switch into sleep mode, the number of packets buffered before the next beacon instance reduces, and this enables the AP to serve a higher number of IoT stations without affecting the delivery rate. For example, as Figure 7 (c) shows, when 200 IoT stations perform their transactions during a 100ms interval every 4 seconds, using Wiotap increases delivery rate by 40% compared to R-AP.

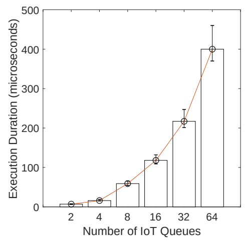

Fig. 8. Effect of the number of IoT queues on the execution duration of Queue Configuration algorithm. The execution time increases linearly versus the number of IoT queues.

As discussed in Section III-B, the complexity of Queue Configuration algorithm is $O(n)$. Figure 8 shows the processing time of this algorithm (implemented in C++) on a single core of a Core i3 processor versus exponential increase in the number of IoT queues. As the results confirm, execution time grows linearly versus the number of queues. Assuming 100 transactions per second, a 100-entry queue $\mathbf{\Delta}$ fills up every 1 second to trigger Queue Configuration algorithm. In this case, even if 64 queues are used, the processor's core is utilized less than 0.05% per second by this algorithm.

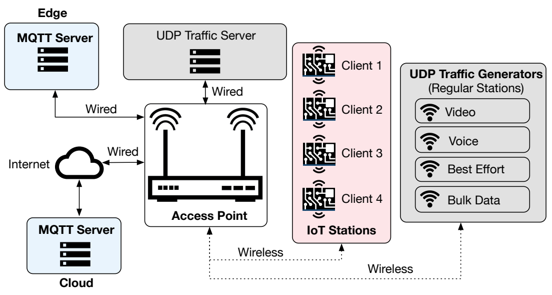

Fig. 9. Testbed architecture. Four IoT stations communicate with the AP in the presence of regular traffic. The IoT stations use MQTT to communicate with the broker.

VI. Empirical Evaluation

This section presents the testbed setup methodology and the empirical performance evaluation of the proposed system. Although Wiotap has been primarily designed to ensure deadline-aware scheduling of large-scale IoT networks, the results of this section show that the proposed solution enhances performance even in networks with a small number of IoT stations.

A. Testbed

Figure 9 shows the testbed architecture. The testbed has been set up as follows.

1) Hardware: We used Cypress CYW43907 [16], [42] as MQTT clients. CYW43907 offers a low-power design, is equipped with two ARM Cortex-R4 processors (for application and 802.11 sub-systems), and supports 802.11g/n. The AP has been implemented using an Intel NUC machine, which includes an Intel Wireless-AC 7265 card acting as the AP interface. We also added a Linksys AE1200 N300 to the AP for the WiLog module. The AP operates in 802.11e mode with 54Mbps capacity.

2) Publish/subscribe model: Due to its widespread use in IoT applications, we have adopted MQTT [43], which is based on the publisher/subscriber model. Each publisher publishes to a topic, and the broker forwards the published messages to all the subscribers interested in the topic. In this paper, we refer to the process of publishing a message and receiving a response as a *transaction*.

3) Background traffic: Many media-centric applications (such as video calling and gaming) use UDP as the transport layer protocol [44] [45]. Also, the application layer protocols over UDP (such as Google's Quick UDP Internet Connections (QUIC) protocol) represent over 7% of all the traffic in the Internet [46], [47]. Hence, UDP traffic can saturate a heterogeneous WiFi network by consuming most of the bandwidth [48], [49]. To mimic this behavior, we generate UDP traffic using a C program that runs on a client and sends data flows to the UDP traffic server. The program is capable of both setting the ToS field to associate an AC with each packet and controlling inter-packet transmission delays to adjust the amount of channel utilization. A similar program runs on the UPD server to continuously send back the received UDP packets to the traffic generator.

4) Energy measurement platform: EMPIOT [50] is used for energy measurement. This platform enables us to accurately measure the energy consumption of a code snippet running on the IoT devices. By annotating the code running on the IoT boards, the start and stop of energy measurement can be accurately controlled. The EMPIOT platform's basic sampling rate is 500Ksps, which are then averaged and streamed to the control software as 1Ksps. The maximum accuracy error of this platform is 4% compared to the existing high-end commercial products.

The IoT client boards (CYW43907) include components that increase the minimum achievable energy consumption compared to the SoCs only. Therefore, in order to merely take into account the energy consumption of 802.11 communication, we need to collect the base energy consumption of the board when the transceiver is in sleep mode. Our measurements show that the base current consumption is approximately 160mA, and wireless communication increases this value to about 250mA. By subtracting the base power consumption from the results collected during experiments, we report the average energy consumption of the board per MQTT transaction.

B. Methodology

The testbed has been used to run a series of experiments in the presence of various categories and background traffic rate. To represent a request-response scenario, IoT stations subscribe to their published topic to ensure that the client will receive the published messages from the broker. When referring to energy in the results, we show the energy consumed by publishing a message and receiving the reply. This process is called a *transaction*. We perform the experiments using MQTT's QoS 1 and QoS 2 modes. The former ensures that the message is delivered *at least once*, and the latter ensures the message is delivered *exactly once*. Although QoS 2 has a higher overhead in terms of latency and number of packets exchanged, it is the preferred QoS mechanism for mission-critical applications. We run 50 transactions for each AC and background traffic rate and depict the median and error bars to show the lower and higher quartiles. All the experiments were conducted after 12 AM to minimize the impact of nearby APs.

Based on the location of the MQTT broker, we have implemented edge and cloud computing scenarios. In the edge scenario, the broker is directly connected to the AP through an Ethernet cable. The mean and standard deviation of RTT are around 3ms and 2ms in this scenario, respectively. In the cloud scenario, we have placed the MQTT broker in a server located in Oregon, US, and the AP and IoT devices are located in Santa Clara, US. We observed that the mean and standard deviation of RTT is 30ms and 10ms, respectively. These scenarios, in particular, enable us to see the impact of round-trip-time (RTT) on energy consumption because the RTT of edge and cloud scenarios are less than and more than the tail time, respectively.

C. Results and discussions

Figures 10 and 11 demonstrate the result for the edge and cloud scenarios, respectively. The maximum and average performance improvement of Wiotap in the edge scenario are 52% and 36% in terms of delay and 44% and 18% in terms of energy. For the cloud scenario, the maximum and average performance improvement are 41% and 18% in terms of delay and 38% and 13% in terms of energy. When the broker is at the network edge, the response packets usually reach the AP during the tail-time. Depending on the deadline of this packet compared to other IoT stations, Wiotap prioritizes the packet to ensure its delivery before its deadline. In the cloud computing scenario, the network latency is usually larger than the tail-time, and stations will have to wake up during the next beacon instance to retrieve downlink packets from the AP. At the wake-up time, since our solution prioritizes the packets over background traffic, the station spends less time in idle listening mode. Therefore, compared to the edge scenario, more energy is spent in the cloud scenario on average per station per transaction.

As the results show, the energy consumption of IoT station is higher in the presence of AC_VO compared to AC_BK and AC_BE. Besides, the rate of background traffic has a higher impact on the energy consumption of IoT station for higher-priority background flows. We justify this behavior as follows. The regular traffic fills up the EDCA queues of corresponding ACs. Whenever a burst of IoT traffic occurs, the priority of IoT packets is promoted by the qdisc module. However, since we do not modify the 802.11e's EDCA at the MAC layer, although the IoT traffic has the highest priority, it would only have a higher chance of being transmitted while contending with the existing traffic in the EDCA queues. In a statistical sense, the transmission probability of IoT traffic is higher compared to lower priority ACs due to the lower values of CW_{min}, CW_{max}, and $AIFS$ for high-priority ACs (IoT traffic). However, the random backoff time chosen by lower-priority ACs adds a level of uncertainty, which results in scenarios where lower-priority flows gain channel access before higher-priority flows [51]. More specifically, to avoid the starvation of low-priority flows, the 802.11e MAC layer prioritization mechanism only offers a higher probability of transmission for higher priority ACs, and this mechanism does not guarantee that the higher priority packets will always be sent before the lower priority ones [31], [52].

VII. Related Work

Tauber and Bhatti [10] studied the impact of beacon and DTIM interval on energy efficiency versus inter-packet arrival time. They also highlighted the importance of per-client DTIM allocation, which has been addressed by [53]. Tozlu et al. [13] presented an extensive energy evaluation of 802.11 stations versus rate, security protocol, control packet overhead, and interference. He et al. [54] showed a higher impact of 802.11 power saving mechanism when the beacon and DTIM intervals are small. They also demonstrated the reduction in energy efficiency as the background traffic increases.

Packet scheduling mechanisms such as [55] and [56] propose solutions to group stations such that only a few stations

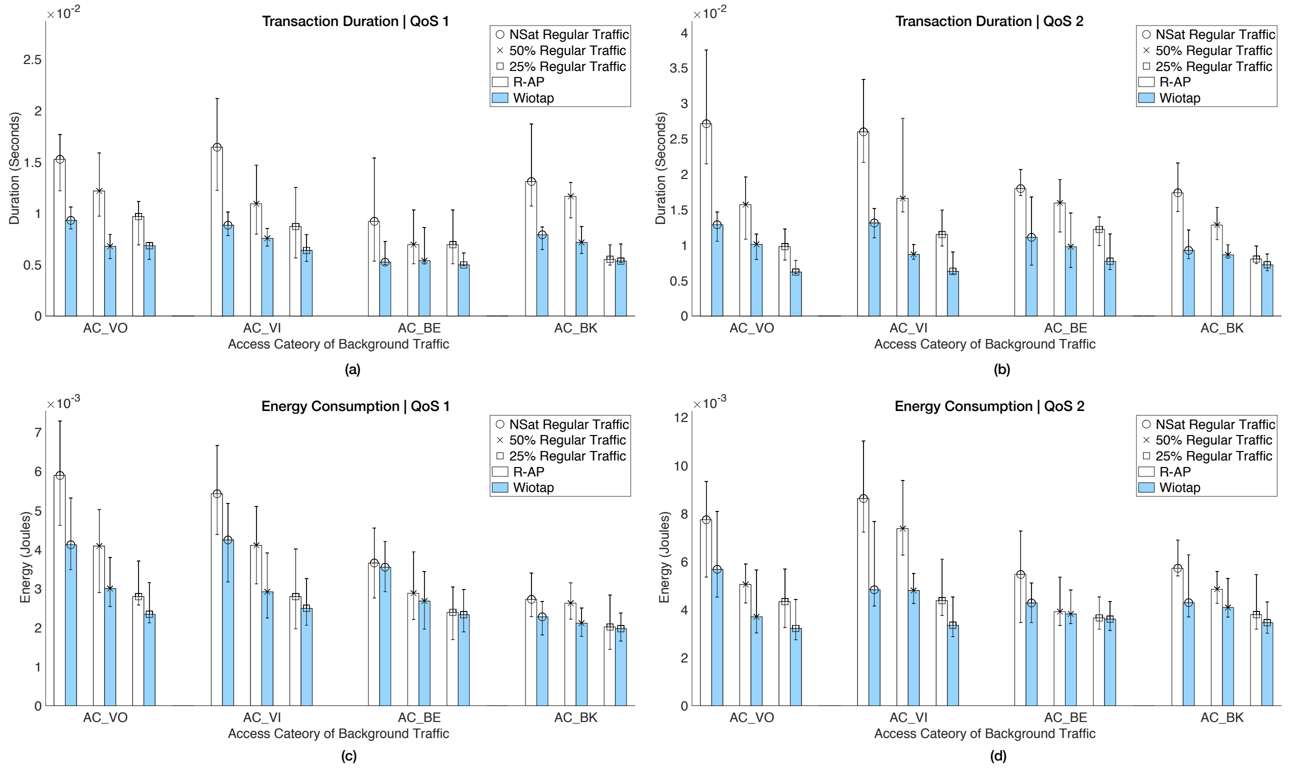

Fig. 10. Average delay and energy per MQTT transaction versus the different levels of background traffic in the edge computing scenario. (a) and (b) show transaction duration, and (c) and (d) show transaction energy consumption.

wake up to receive the packets after each beacon interval. Similarly, [57], [58], and [59] grouped the stations to control channel access contention. Kwon and Cho [60] assume that the AP is aware of the priorities of the registered stations. They prioritize packet reception according to the priorities of the stations based on their profile information such as the remaining power level of each station. Towards reducing the waiting time (and energy consumption) of PSM stations, Rozner et al. [61] proposed NAPman to transmit the packets of these stations before those stations that do not utilize PSM. The downlink packet scheduler proposed in [62] prioritizes downlink burst packet delivery after every beacon interval based on historical data and attention fairness. Clients with smaller attention requests are serviced before the others, thus allowing them to spend less energy to get one unit of attention. Also, the clients with larger attention requests sleep longer compared to other popular scheduling methods such as priority round robin and priority first come first serve. To enhance the sleep duration of clients, Liu et al. [63] postpone packet transmission from AP to clients. Considering a VoIP application, stations measure the tolerable delay of incoming packets and request for a sleep schedule from the AP accordingly. None of these approaches address the energy efficiency and timeliness concerns of using IoT devices in 802.11 networks. Besides, the aforementioned solutions did not present their effectiveness in large-scale deployments with tens of connected stations. To the best of our knowledge, our work is the first that addresses deadline-aware scheduling of IoT packets for large-scale deployments.

In contrast to the aforementioned approaches, which employ packet scheduling on top of the regular channel access mechanism of 802.11 (i.e., CSMA), some works completely utilize TDMA to enhance timeliness and offer bandwidth guarantee. For example, [64] has modified the 802.11 driver and utilized a simple TDMA scheduling to enhance the sampling rate of wireless control systems. Towards higher scalability, some of these works employ similar or enhanced variations of the scheduling algorithms that have been originally proposed for task scheduling [65], [66]. As an example, the LLF scheduling strategy has been used in [67] to develop a heuristic scheduling algorithm for WirelessHART networks. LLF has also been employed by [1] to enhance the number of admitted nodes in a real-time mobile wireless networks. Although the timeliness and reliability of these solutions are essential for application-specific, mission-critical scenarios such as factory automation, utilizing a TDMA approach in regular IoT applications requires an exact identification of each station's bandwidth requirements. However, this would not be possible in scenarios such as smart home where regular and IoT stations coexist and the traffic of these stations is highly unpredictable. In this regard, the novelties of this paper are (i) providing a new formulation of LLF for the delivery deadline of packets when Adaptive PSM is used, (ii) utilizing network layer to MAC layer priority mapping, and (iii) placing the proposed solution in the network layer to simplify its adoption and make it scalable and independent of the MAC layer implementation.

To enhance the performance of scheduling and channel access, various solutions were proposed to benefit from the

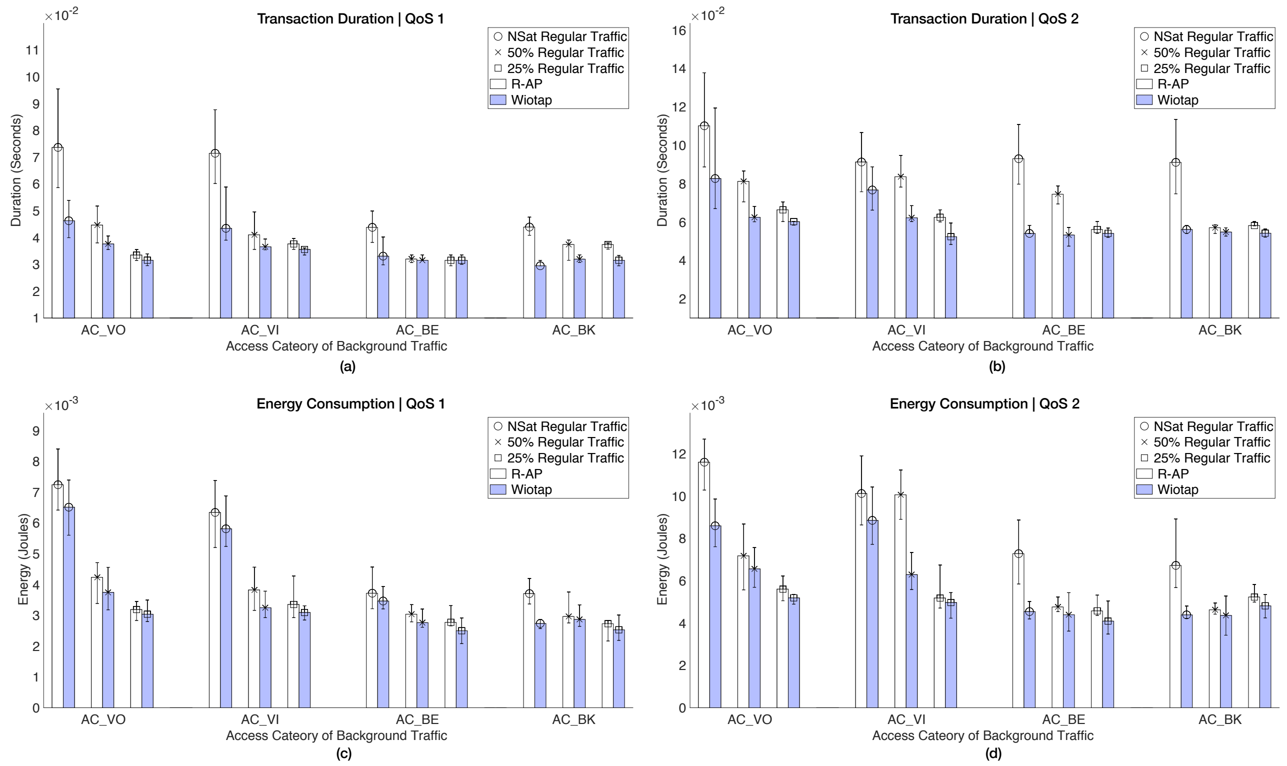

Fig. 11. Average delay and energy per MQTT transaction versus the different levels of background traffic in the cloud computing scenario. (a) and (b) show transaction duration, and (c) and (d) show transaction energy consumption.

information collected from the transport and application layer protocols. Birlik et al. [68] enhance the delivery of high-quality video streams to end-users by prioritizing critical video packets in a mesh network. By utilizing the H.264 encoding standard to classify the packets according to their importance, this approach modifies the ToS field in the IP header to promote the priority of important packets. Martin and Feamster [69] determine flow priorities based on user activity type. Each client uses a tool to find the currently active application window on the device and sends UDP control packets containing information such as the port numbers of the active and inactive processes. On receiving the control packets, the router prioritizes the flows which are associated with active processes by allocating a larger share of tickets under lottery based queue scheduling mechanism [70]. Pyles et al. [20] utilize an application classifier to increase the priority of interactive Android applications. When a high priority application is detected, the WiFi driver uses the A-PSM mechanism if the rate of packet exchange is beyond a certain threshold.

Wamser et al. [71] addressed resource allocation to multi-media traffic on home gateways. If the video or audio local playback buffer falls below 25s, its flow is moved to a higher priority queue, and once the buffer reaches 40s, the flow is downgraded to a lower priority queue. Flaithearta et al. [72] proposed an intelligent AP for VoIP traffic to address intra-AC prioritization among VoIP flows. This method finds the VoIP quality using the ITU-T E-Model [73] and the AP sets the DSCP value of packets based on network characteristics collected from the RTP Control Protocol (RTCP). By using the additional higher priority AC_VO transmission queue provided by the 802.11aa standard, they divert a few VoIP calls to prioritize them over others. Qazi et al. [74] propose fine-grained mobile application detection using a machine learning trainer based on a decision tree in the control plane. They use `netstat` logs from employee devices along with the flow features (first N packet sizes, port numbers, IP address range). In [75], the router dynamically adjusts bandwidth allocation of flows using Linux's `tc` utility. The aggregated bandwidths are computed for video, web browsing, file transfer, and voice classes using a linear utility function [76] on account of the contextual-priority reports sent by clients. The authors in [77] have demonstrated application-aware networking for video streaming. They identify characteristics of the flows through deep packet inspection and forward them via least congested links by dynamically changing the routing paths. They have compared the performance of bandwidth-based and DPI-based path selection mechanisms regarding *buffered playtime* in a software-defined network (SDN). Afzal et al. [78] proposed a context-aware resource allocation scheme in wireless multimedia sensor-based WLANS. This method formulates an optimization problem on the basis of the service requirements of each flow, and allocates appropriate bandwidth and TXOP to the stations.

Although Wiotap does not rely on information provided by transport and application layer, it can be enhanced further by incorporating this information. For example, the AP can automatically detect and classify IoT devices once MQTT or CoAP is used at the application layer. This presents the need to identify these devices manually on the AP. However, this may impose higher overhead to the AP to perform deep packet inspection. Also, if an SDN architecture is employed to perform the inspections remotely, the delay and overhead of AP-controller must be carefully taken into account. In particular, since IoT traffic is usually event-based and the number of packets exchanged with the AP is significantly lower than that of regular user traffic, it is essential to ensure the AP performs scheduling promptly for all the IoT packets.

VIII. Conclusion and Future Work

In this paper, we proposed Wiotap, which is an 802.11 AP serving IoT and regular stations. Wiotap enhances the energy efficiency and timeliness of IoT stations in large-scale networks by applying per-packet scheduling of downlink traffic based on the power state of stations. In addition, Wiotap ensures the high efficiency of IoT stations in the presence of regular traffic and protects these stations against DoS attacks.

The performance and applicability of the proposed approach can be enhanced by integrating context awareness. For example, application layer protocol detection eliminates the burden of manually identifying IoT stations. Also, analyzing per-device traffic pattern enables the AP to control the power status of stations by sending management packets. To extend the proposed mechanism to scenarios with multiple APs and mobile stations, a SDN architecture can be employed to collect the required data (e.g., laxity of packets) from multiple APs and run the proposed algorithms centrally.

Acknowledgment

This work has been partially supported by a research grant from Cypress Semiconductor Corporation (Grant No. CYP001).

References

[1] B. Dezfouli, M. Radi, and O. Chipara, "REWIMO: A real-time and reliable low-power wireless mobile network," *ACM Transactions on Sensor Networks (TOSN)*, vol. 13, no. 3, p. 17, 2017.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[3] F. Tramarin, S. Vitturi, M. Luvisotto, and A. Zanella, "On the use of IEEE 802.11n for industrial communications," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1877–1886, 2016.

[4] Gartner Inc. Leading the IoT. [Online]. Available: https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf

[5] A. Nikoukar, S. Raza, A. Poole, M. Güneş, and B. Dezfouli, "Low-power wireless for the Internet of Things: Standards and applications," *IEEE Access*, vol. 6, pp. 67 893–67 926, 2018.

[6] A. Di Serio, J. Buckley, J. Barton, R. Newberry, M. Rodencal, G. Dunlop, and B. O'Flynn, "Potential of sub-ghz wireless for future iot wearables and design of compact 915 mhz antenna," *Sensors*, vol. 18, no. 1, p. 22, 2018.

[7] B. Gerislioglu, A. Ahmadivand, M. Karabiyik, R. Sinha, and N. Pala, "VO_2-based reconfigurable antenna platform with addressable microheater matrix," *Advanced Electronic Materials*, vol. 3, no. 9, 2017.

[8] N. Khalid, T. Yilmaz, and O. B. Akan, "Energy-efficient modulation and physical layer design for low terahertz band communication channel in 5g femtocell Internet of Things," *Ad Hoc Networks*, vol. 79, pp. 63–71, 2018.

[9] Y.-C. Chang and J.-P. Sheu, "An energy conservation MAC protocol in wireless sensor networks," *Wireless Personal Communications*, vol. 48, no. 2, pp. 261–276, 2009.

[10] M. Tauber and S. N. Bhatti, "The effect of the 802.11 power save mechanism (PSM) on energy efficiency and performance during system activity," in *IEEE International Conference on Green Computing and Communications (GreenCom)*, 2012, pp. 573–580.

[11] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, no. 2, pp. 62–67, 2012.

[12] A. Bartoli, M. Dohler, J. Hernández-Serrano, A. Kountouris, and D. Barthel, "Low-power low-rate goes long-range: The case for secure and cooperative machine-to-machine communications," in *International Conference on Research in Networking*. Springer, 2011, pp. 219–230.

[13] S. Tozlu, M. Senel, W. Mao, and A. Keshavarzian, "Wi-Fi enabled sensors for Internet of Things: A practical approach," *IEEE Communications Magazine*, vol. 50, no. 6, 2012.

[14] D. Thomas, R. McPherson, G. Paul, and J. Irvine, "Optimizing power consumption of Wi-Fi inbuilt IoT device: an MSP430 processor and an ESP-03 chip provide a power-efficient solution," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 92–100, 2016.

[15] AVNET Inc. BCM4343W: 802.11b/g/n WLAN, Bluetooth and BLE SoC Module. [Online]. Available: https://products.avnet.com/opasdata/d120001/medias/docus/138/AES-BCM4343W-M1-G_data_sheet_v2_3.pdf

[16] Cypress Semiconductor. CYW43907: IEEE 802.11a/b/g/n SoC with an Embedded Applications Processor. [Online]. Available: http://www.cypress.com/file/298236/download

[17] Silicon Labs. Zentri AMW036/AMW136 Data Sheet. [Online]. Available: https://www.silabs.com/documents/login/data-sheets/ADS-MWx36-ZentriOS-101R.pdf

[18] Texas Instruments Incorporated. CCC3200MOD Data Sheet. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc3200mod.pdf

[19] B. Dezfouli, V. Esmaeelzadeh, J. Sheth, and M. Radi, "A review of software-defined WLANs: Architectures and central control mechanisms," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 431–463.

[20] A. J. Pyles, X. Qi, G. Zhou, M. Keally, and X. Liu, "SAPSM: Smart adaptive 802.11 PSM for smartphones," in *ACM conference on ubiquitous computing*, 2012, pp. 11–20.

[21] MediaTek Inc. MT7687: a low power 1T1R 802.11n single-band Wi-Fi subsystem and a power management unit (PMU). [Online]. Available: https://labs.mediatek.com/en/chipset/MT7687

[22] Qualcomm Technologies Inc. MT7687: Low-Energy Wi-Fi Single-Band 802.11a/b/g/n SoC. [Online]. Available: https://www.qualcomm.com/products/qca4002

[23] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta, "Wireless wakeups revisited: energy management for VoIP over Wi-Fi smartphones," in *International Conference on Mobile Systems, Applications and Services (MobiSys)*. ACM, 2007, pp. 179–191.

[24] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "802.11 power-saving mode for mobile computing in Wi-Fi hotspots: limitations, enhancements and open issues," *Wireless Networks*, vol. 14, no. 6, pp. 745–768, 2008.

[25] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl, "A case for adapting channel width in wireless networks," in *Computer Communication Review Volume 28*. ACM, 2008, pp. 135–146.

[26] J. Liu and L. Zhong, "Micro power management of active 802.11 interfaces," in *International Conference on Mobile Systems, Applications and Services (MobiSys)*. ACM, 2008, pp. 146–159.

[27] F. R. Dogar, P. Steenkiste, and K. Papagiannaki, "Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices," in *International Conference on Mobile Systems, Applications and Services (MobiSys)*. ACM, 2010, pp. 107–122.

[28] N. Ding, A. Pathak, D. Koutsonikolas, C. Shepard, Y. C. Hu, and L. Zhong, "Realizing the full potential of PSM using proxying," in *Proceedings of INFOCOM*. IEEE, 2012, pp. 2821–2825.

[29] S. K. Saha, P. Malik, S. Dharmeswaran, and D. Koutsonikolas, "Revisiting 802.11 power consumption modeling in smartphones," in *International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2016, pp. 1–10.

[30] B. Vijay and B. Malarkodi, "Improved QoS in WLAN using IEEE 802.11e," *Procedia Computer Science*, vol. 89, pp. 17–26, 2016.

[31] IEEE 802.11 Working Group, "IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications 2007."

[32] G. Cena, L. Seno, A. Valenzano, and C. Zunino, "On the performance of IEEE 802.11e wireless infrastructures for soft-real-time industrial

applications," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 425–437, 2010.

[33] M. Frikha, T. Najet, and F. Tabbana, "Mapping DiffServ to MAC differentiation for IEEE 802.11e," in *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW)*. IEEE, 2006, pp. 79–79.

[34] Malinen, Jouni. hostapd: IEEE 802.11. [Online]. Available: http://hostap.epitest.fi/hostapd

[35] R. Love, *Linux kernel development*. Pearson Education, 2010.

[36] R. Russell and H. Welte. Linux netfilter hacking howto. [Online]. Available: http://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO

[37] B. Hubert, G. Maxwell, R. van Mook, M. van Oosterhout, P. Schroeder, J. Spaans, and P. Larroy. Linux advanced routing & traffic control howto. [Online]. Available: https://www.tldp.org/HOWTO/Adv-Routing-HOWTO/

[38] Openwrt. Network Traffic Control (QoS). [Online]. Available: https://wiki.openwrt.org/doc/howto/packet.scheduler/packet.scheduler

[39] R. Rosen, *Linux kernel networking: Implementation and theory*. Apress, 2014.

[40] M. A. Brown, "Traffic control HOWTO," *Guide to IP Layer Network*, p. 49, 2006.

[41] A. Varga, "The OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference, June*, 2001, pp. 319–324.

[42] Cypress Semiconductor. CYW943907AEVAL1F Evaluation Kit. [Online]. Available: http://www.cypress.com/documentation/development-kitsboards/cyw943907aeval1f-evaluation-kit

[43] M. Version, "3.1. 1," *Edited by Andrew Banks and Rahul Gupta*, vol. 10, 2014.

[44] A. Bujari, G. Quadrio, C. Palazzi, D. Ronzani, D. Maggiorini, and L. Ripamonti, "Network traffic analysis of the steam game system," in *Consumer Communications & Networking Conference (CCNC)*. IEEE, 2017, pp. 716–719.

[45] C. Yu, Y. Xu, B. Liu, and Y. Liu, "Can you SEE me now? A measurement study of mobile video calls," in *Proceedings of INFOCOM*. IEEE, 2014, pp. 1456–1464.

[46] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The QUIC transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 183–196.

[47] P. Kumar and B. Dezfouli, "Implementation and analysis of QUIC for MQTT," in *Computer Networks*, vol. 150. Elsevier, 2019, pp. 28–45.

[48] N. Soetens, J. Famaey, M. Verstappen, and S. Latre, "SDN-based management of heterogeneous home networks," in *International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 402–405.

[49] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, K. Papagiannaki, and P. Steenkiste, "Identifying the root cause of video streaming issues on mobile devices," in *ACM Conference on Emerging Networking Experiments and Technologies*, 2015, p. 24.

[50] B. Dezfouli, I. Amirtharaj, and C.-C. Li, "Empiot: An energy measurement platform for wireless IoT devices," *Journal of Network and Computer Applications*, vol. 121, pp. 135–148, Elsevier, 2018.

[51] S. Mangold, S. Choi, G. R. Hiertz, O. Klein, and B. Walke, "Analysis of IEEE 802.11e for QoS support in wireless LANs," *IEEE Wireless Communications*, vol. 10, no. 6, pp. 40–50, 2003.

[52] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, and L. Stibor, "IEEE 802.11e Wireless LAN for Quality of Service," in *Proc. European Wireless*, vol. 2, 2002, pp. 32–39.

[53] J. Wang and T. Nagy, "Maintaining delivery traffic indication message (DTIM) periods on a per-wireless client device basis," Aug. 23 2011, US Patent 8,005,032.

[54] Y. He, R. Yuan, X. Ma, and J. Li, "The IEEE 802.11 power saving mechanism: An experimental study," in *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2008, pp. 1362–1367.

[55] H.-P. Lin, S.-C. Huang, and R.-H. Jan, "A power-saving scheduling for infrastructure-mode 802.11 wireless LANs," *Computer Communications*, vol. 29, no. 17, pp. 3483–3492, 2006.

[56] Y. Li, X. Zhang, and K. L. Yeung, "A novel delayed wakeup scheme for efficient power management in infrastructure-based IEEE 802.11 WLANs," in *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2015, pp. 1338–1343.

[57] M. Maity, B. Raman, and M. Vutukuru, "TCP download performance in dense wifi scenarios: analysis and solution," *IEEE Transactions on Mobile Computing*, vol. 16, no. 1, pp. 213–227, 2017.

[58] Z. Abichar and J. M. Chang, "Group-based medium access control for ieee 802.11n wireless LANs," *IEEE Transactions on Mobile Computing*, vol. 12, no. 2, pp. 304–317, 2013.

[59] Y. Yuan, W. A. Arbaugh, and S. Lu, "Towards scalable MAC design for high-speed wireless LANs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, no. 1, p. 012597, 2007.

[60] S.-W. Kwon and D.-H. Cho, " Efficient power management scheme considering inter-user QoS in wireless LAN," in *Vehicular Technology Conference*. IEEE, 2006, pp. 1–5.

[61] E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu, "Napman: network-assisted power management for wifi devices," in *International Conference on Mobile Systems, Applications and Services (MobiSys)*. ACM, 2010, pp. 91–106.

[62] Z. Zeng, Y. Gao, and P. Kumar, "SOFA: A sleep-optimal fair-attention scheduler for the power-saving mode of WLANs," in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2011, pp. 87–98.

[63] L. Liu, X. Cao, Y. Cheng, and Z. Niu, "Energy-Efficient Sleep Scheduling for Delay-Constrained Applications Over WLANs," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2048–2058, 2014.

[64] Y.-H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka, "RT-WiFi: Real-time high-speed communication protocol for wireless cyber-physical control applications," in *Real-Time Systems Symposium (RTSS)*. IEEE, 2013, pp. 140–149.

[65] W. Zhang, S. Teng, Z. Zhu, X. Fu, and H. Zhu, "An improved least-laxity-first scheduling algorithm of variable time slice for periodic tasks," in *International Conference on Cognitive Informatics*. IEEE, 2007, pp. 548–553.

[66] T. Kothmayr, J. Hirscheider, A. Kemper, A. Scholz, and J. Heuer, "Comparing heuristics and linear programming formulations for scheduling of in-tree tasksets," *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2014.

[67] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time scheduling for WirelessHART networks," in *Real-Time Systems Symposium (RTSS)*. IEEE, 2010, pp. 150–159.

[68] F. Birlik, O. Ercetin, and O. Gurbuz, "Prioritized video streaming in wireless mesh networks," in *International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2007, pp. 1–3.

[69] J. Martin and N. Feamster, "User-driven dynamic traffic prioritization for home networks," in *Proceedings of the SIGCOMM workshop on Measurements up the stack*, 2012, pp. 19–24.

[70] C. A. Waldspurger and W. E. Weihl, "Lottery scheduling: Flexible proportional-share resource management," in *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*, 1994, p. 1.

[71] F. Wamser, L. Iffländer, T. Zinner, and P. Tran-Gia, "Implementing application-aware resource allocation on a home gateway for the example of YouTube," in *International Conference on Mobile Networks and Management*. Springer, 2014, pp. 301–312.

[72] P. O Flaithearta, H. Melvin, and M. Schukat, "A QoS enabled multimedia WiFi access point," *International Journal of Network Management*, vol. 25, no. 4, pp. 205–222, 2015, Wiley Online Library.

[73] G107. The E-model: a computational model for use in transmission planning,. [Online]. Available: https://www.itu.int/rec/T-REC-G.107

[74] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in SDN," in *ACM SIGCOMM Computer Communication Review*, vol. 43, 2013, pp. 487–488.

[75] I. N. Bozkurt, Y. Zhou, T. Benson, B. Anwer, D. Levin, N. Feamster, A. Akella, B. Chandrasekaran, C. Huang, B. Maggs *et al.*, "Dynamic prioritization of traffic in home networks," in *CoNEXT Student Workshop, Heidelberg, Germany*, 2015.

[76] S. Shenker, "Fundamental design issues for the future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[77] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "SDN-based application-aware networking on the example of youtube video streaming," in *Second European Workshop on Software Defined Networks (EWSDN)*. IEEE, 2013, pp. 87–92.

[78] B. Afzal, S. A. Alvi, G. A. Shah, and W. Mahmood, "Energy efficient context aware traffic scheduling for IoT applications," *Ad Hoc Networks*, vol. 62, pp. 101–115, 2017.