IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

Countermeasuring Aggressors via Intelligent Adaptation of Contention Window in CSMA/CA Systems

Amir-Hossein Yazdani-Abyaneh[1], Mohammed Hirzallah[1], and Marwan Krunz[1]

[1]Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721, USA. An abridged version of this paper was presented at the IEEE DySPAN '19 Conference, Nov. 11 - Nov. 14, 2019, NJ, USA.

Corresponding author: Amir-Hossein Yazdani-Abyaneh (email:yazdaniabyaneh@arizona.edu).

This research was supported in part by NSF (grants # 2229386 and 1822071) and by the Broadband Wireless Access & Applications Center (BWAC). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of NSF. .

ABSTRACT To coordinate channel access and reduce collisions over unlicensed bands, wireless technologies implement a listen-before-talk (LBT) strategy, a variant of Carrier Sense Multiple Access (CSMA) with Collision Avoidance (CA). In LBT, a node backs off for a randomly selected amount of time, upper-bounded by the *minimum contention window (CW_{min})* which is specified by standard settings. However, an aggressive node can choose a lower $\mathrm{CW}_{\mathrm{min}}$ value, deviating from standards settings, to gain an unfair throughput advantage at the cost of compliant nodes performance. To address this problem, we propose a framework called *Intelligent Contention Window (ICW)* that allows compliant nodes to adapt their $\mathrm{CW}_{\mathrm{min}}$ values to counter aggressive nodes and achieve their fair share of the channel's airtime. The adaptation process is based on a *random forest*, a machine learning model that includes a large number of decision trees. We train the random forest in a supervised manner to recommend the possible best $\mathrm{CW}_{\mathrm{min}}$ over a large number of spectrum sharing scenarios. Our results show high generalization performance of the random forest for diverse aggressive spectrum sharing settings. We validate our design using over-the-air hardware experiments as well as simulations. Our results suggest that under ICW, nodes receive their fair shares of the channel airtime and achieve multi-fold boosting in throughput and reduction in latency in both static and dynamic aggression settings. Our SDR experiments show $5.62\times$ throughput improvement when ICW is used relative to the Wi-Fi protocol.

INDEX TERMS Aggressive behavior, CSMA/CA, $\mathrm{CW}_{\mathrm{min}}$, distributed MAC, fairness, machine learning, random forest.

I. INTRODUCTION

The number of Wi-Fi access points (APs) has quadrupled between 2018 and 2023 [1]. Besides Wi-Fi systems, new wireless technologies, such as 5G New Radio Unlicensed (5G NR-U) and LTE Licensed Assisted Access (LTE-LAA), also operate over the unlicensed 5 and 6 GHz bands, supplementing licensed cellular services [2]. Unlicensed-band technologies use channel access protocols that are fundamentally based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and exponential backoff. Devices that utilize these technologies have specific System-on-Chip (SoC) modules or Network Interface Cards (NICs), in which time-critical MAC and PHY functionalities are executed. SoC's and NIC's run specific tasks based on their installed firmware. Research has shown that it is possible to tamper with such firmware and modify the operating behavior of the modules [3], [4]. In this work, we focus on the challenges related to aggressive channel access behavior. In particular, we consider a scenario where one or more coexisting devices manipulate their channel access parameters, specifically their $\mathrm{CW}_{\mathrm{min}}$ values, to gain unfair advantage in the channel's airtime.

Under CSMA/CA, a node that wants to transmit a packet must first sense the channel for a fixed Inter-Frame Space (IFS) interval. During this interval, if the channel remains idle, the node will proceed with a packet transmission; if not, the node defers its transmission and waits for a random backoff duration. The backoff duration consists of k randomly selected slots, uniformly sampled from the set $\{0, 1, ..., W-1\}$, where W is called the contention window. Starting with an initial value of $W = \mathrm{CW}_{\min}$, W is doubled after each collision until it reaches a maximum value of $\mathrm{CW}_{\max}$. Thus, after j successive collisions, the backoff timer k is selected randomly from $\{0, 1, ..., \min(2^j\mathrm{CW}_{\min}, \mathrm{CW}_{\max}) - 1\}$. After a successful transmission, W is set back to $\mathrm{CW}_{\min}$.

If all nodes that share a channel comply with their respective standards, CSMA/CA is proven to be fair. However, in the presence of aggressive nodes that select lower $\mathrm{CW}_{\min}$ values than the standard setting, the throughput of compliant nodes degrades and CSMA/CA can no longer guarantee fairness. Due to the distributed nature of the channel access, it is difficult to identify nodes with non-compliant $\mathrm{CW}_{\min}$ values. Compliant nodes need a way to detect whether or not they are getting their fair share of the airtime, and accordingly adapt their $\mathrm{CW}_{\min}$ to not fall behind in their throughput performance.

To cast more light on this issue, we conduct a simple simulation in which three Wi-Fi nodes that operate according to the IEEE 802.11ac standard with an access category AC3 share the same unlicensed channel. All three nodes are backlogged (saturated traffic) and have a fixed transmission rate of 12.79 Mbps. All three nodes are in each other's sensing ranges, and their relative locations with respect to the AP ensure all packet transmissions are successful. Two nodes, say N_2 and N_3, are configured to act aggressively by setting their initial $\mathrm{CW}_{\min}$ values to 4. We consider two cases for the setting of N_1's $\mathrm{CW}_{\min}$. In Case (a), N_1 chooses $\mathrm{CW}_{\min} = 16$ (the standardized setting [5]), while in Case (b), N_1 randomly chooses its $\mathrm{CW}_{\min}$ between 2 and 16. The throughput for the three individual nodes is shown in Figure 1. For Case (a), N_2's and N_3's throughputs account for 91.26% of the total network throughput. In contrast, for Case (b) the random assignment of $\mathrm{CW}_{\min}$ value partially alleviates the unfairness issue and improves the relative throughput of N_1 to 22.69%. Rather than randomizing the selection of $\mathrm{CW}_{\min}$, in this paper, we exploit Machine Learning (ML) techniques to bring N_1's relative throughput close to its fair share of 33%. Achieving this fair allocation is challenging because the $\mathrm{CW}_{\min}$ values of N_2 and N_3 are not known to N_1.

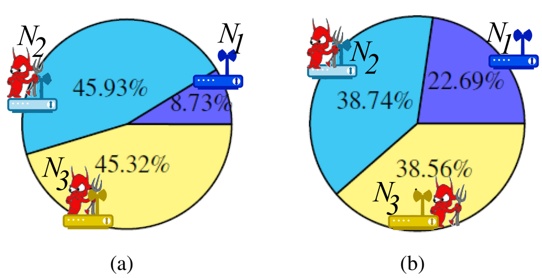


FIGURE 1: Relative per-node throughput for a network of three nodes. Nodes N_2 and N_3 are aggressive with $\mathrm{CW}_{\min}$ set to 4: (a) N_1's $\mathrm{CW}_{\min}$ is set to 16, (b) N_1 randomly selects its $\mathrm{CW}_{\min}$ between 2 and 16.

In Figure 2, we show the throughput ratio of an aggressive node over a compliant node's throughput, where there is only one compliant node and the number of aggressive nodes are varied from 0 to 100. Results are gathered through numerical analysis of the Markov Chain of the DCF protocol under aggressive settings (see Section III). It can be seen that the unfairness issue is present even for high number of nodes in the network. It can also be concluded that a lower $\mathrm{CW}_{\min}$ will cause more unfairness i.e., higher throughput ratio.

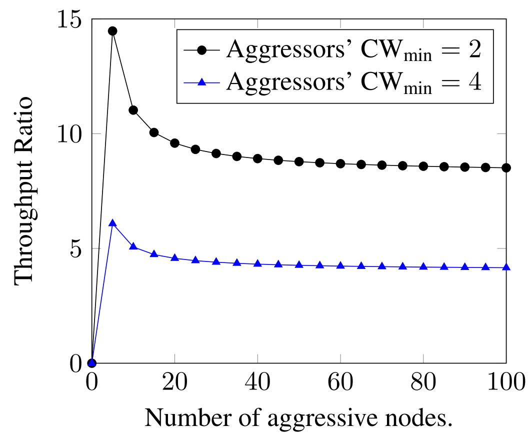


FIGURE 2: Average throughput ratio of an aggressive node over a compliant node ($\mathrm{CW}_{\min} = 16$) vs. number of aggressive nodes in the network.

Modifying the CSMA/CA protocol and adapting some of its parameters, such as $\mathrm{CW}_{\min}$, $\mathrm{CW}_{\max}$, etc., using classical techniques have been extensively investigated in the literature. However, employing ML techniques is still in its infancy. We introduce a framework called *Intelligent Contention Window (ICW)*, which allows a node to adapt its $\mathrm{CW}_{\min}$ value to alleviate the effect of low $\mathrm{CW}_{\min}$ settings by aggressive nodes. We refer to such an adapting node as *intelligent node*. This work presents two options that an intelligent node can take to adapt its $\mathrm{CW}_{\min}$. The first option requires the intelligent node to know the $\mathrm{CW}_{\min}$ values of all its neighbors – a $\mathrm{CW}_{\min}$ estimation technique is presented in [6] – and accordingly calculate its expected airtime using Markovian analysis. The intelligent node can then choose the $\mathrm{CW}_{\min}$ that results in the fairest expected airtime (see Section III). The second option, the former knowledge constraint on $\mathrm{CW}_{\min}$ settings of neighboring nodes is relaxed, instead the intelligent node can monitor the channel and obtain the statistics of neighboring nodes' channel access behavior. Using these statistics along with a trained ML algorithm, e.g., a random forest, the intelligent node can then determine an appropriate $\mathrm{CW}_{\min}$ value (see Section IV). The second option avoids computationally intensive numerical calculations associated with Markovian analysis. The choice of selecting random forest stems from the fact that it is trained faster and run with lower computational complexity compared to other deep learning models (e.g., deep feed-forward neural networks and convolution neural networks [7], [8]).

The CW_{min} adaptation is formulated as an optimization problem in which intelligent nodes maximize their fair share of the airtime. Our fairness criterion captures the channel idle duration, where idle time is equally divided between active nodes. We analyze the Markov chain of the CSMA/CA mechanism in the presence of aggressive nodes, derive the channel access and collision probabilities for well-behaving and aggressive nodes, and formulate the objective function of fair sharing of the airtime. We discuss the challenges associated with solving the problem using classical optimization techniques and show how the problem can be modeled using empirical estimates and solved using ML techniques. Input features used by the ML module for CW_{min} adaptation are selected to indirectly characterize the state of the wireless channel. Intelligent nodes build features by observing the channel for a monitoring period and gathering empirical estimates of channel occupancy, busy, and idle times.

The main contributions of this paper are summarized as follows:

- We model the problem of optimizing the fair share of airtime in CSMA/CA channel access under aggressive settings and discuss related challenges for solving the problem using classical approach. We introduce empirical modeling of the problem and show how ML can be used to learn solutions of the problem under different aggression settings. Our ML solution, based on random forests, lets intelligent nodes optimize their CW_{min} such that they receive their fair share of channel airtime in the presence of aggressors. Intelligent nodes are self-enforcing, meaning they only adapt their windows when aggressors are present and fallback to regular CW_{min} settings when aggressors retreat. Our solution works in a distributed fashion and requires no communication overhead between intelligent nodes, making it appealing for real-world deployments.
- We develop over-the-air (OTA) testbed using USRP-based software defined radios (SDRs) to validate the generalization of our ML model. We also develop a discrete-event simulation framework for generating training data and deriving optimum CW_{min} settings under diverse network conditions. The simulator helps with considering a large set of aggressive scenarios and obtain excellent generalization performance. We conduct extensive optimization and ablation studies to optimize the hyper parameters of the random forest. We also evaluate feature importance and determine which features have the highest impact on generalization performance.
- We define different aggressive behaviors, including static as well as slow and fast dynamics, and show the effect of these dynamics on the intelligent nodes' CW_{min} adaptation process. Intelligent nodes can always track an aggressor's dynamics and maintain their fair share of the airtime. Simulation results show that under static aggressive scenarios, where an aggressor uses a fixed CW_{min} value below the default one, fairness is improved by 36.4% compared to the DCF protocol adopted by 802.11 systems. This is thanks to the adaptation of CW_{min} at the intelligent nodes. Consequently, these nodes improve their throughput by $5.96\times$ and decrease their frame delivery latency by 87.11%. Furthermore, under dynamic aggression scenarios, where aggressors change their CW_{min}'s, intelligent nodes improve their throughput by 58.6% compared to standard techniques. Our OTA USRP experimental results suggest that by adopting ICW, intelligent nodes achieve $5.62\times$ improvement in throughput compared to the standard DCF protocol.

The rest of the paper is organized as follows. Section II surveys related work. The ICW framework is introduced in Section III. We present our ML solution and its optimization in Sections IV and V, respectively. Evaluation results are provided in Section VI. Finally, we conclude the paper in Section VII.

II. RELATED WORKS

Standard MAC settings could be tampered to gain advantage over or degrade the performance of compliant stations [3], [4]. By reverse engineering chips and their firmwares, aggressive users can modify the firmware and change lower-MAC functionalities. Shulz et al. [9] developed *Nexmon*, a firmware patching framework for Broadcom/Cypress chips [10], where modifications to the CSMA/CA mechanism can be done, e.g., by changing the CW_{min} value. New functionalities, such as CSI measurements from commodity devices can also be enabled by firmware modification [11], [12]. In [13], authors used the Nexmon framework to define new functionalities in the programmable state machine of the D11 core, which drives all low-level and time-critical MAC operations, and developed a reactive jammer on a Nexus 5 smartphone. In [14], the authors modified the firmware of Qualcomm Atheros AR7010 AR9271 SoC's used in off-the-shelf Wi-Fi dongles to perform jamming attacks. The authors located special-purpose registers that define the channel access settings, such as CW_{min} and CW_{max}. Furthermore, they found that the original driver divided CW_{min} by two, which caused unfairness to other devices with a standard CW_{min} value. In [15], the authors modified the microcode of the Network Interface Card (NIC) of off-the-shelf IEEE 802.11 access points to perform reactive jamming functionalities, facilitated by the *OpenFWWF* project [16]. A high-level reactive programming language was presented in [17] to program Wi-Fi chips on mobile-consumer devices and extend or mend PHY, MAC, or IP layer mechanisms. These tools can help researchers evaluate different alterations of MAC and PHY functionalities, but pose potential adversarial threats.

Heusse et al. [18] showed the negative impact of the exponential backoff mechanism of the CSMA/CA scheme on WLAN performance. To overcome this negative impact and enhance both throughput and fairness, the authors in [19] introduced a scheme where each node is provided with a set of backoff windows. Instead of following the CSMA/CA protocol, nodes select one of the backoff windows based on observed downlink throughput. Ksentini et al. [20] proposed a deterministic algorithm to decrease the number of collisions and retransmissions. Rather than doubling the W value after each collision, their algorithm modifies both lower and upper ends of the randomly selected backoff interval based on the current and previous traffic loads. Their results show improved fairness and throughput. To maximize channel utilization, Xia et al. [21] equipped nodes with a *proportional-derivative* (PD) controller, which adjusts W based on the average number of consecutive idle slots between two transmissions. The authors in [22] proposed a measurement-based scheme to adapt $CW_{\min}$ and $CW_{\max}$ so as to meet some QoS requirements. Chen et al. [23] introduced a game theory-based model to control contention. The equilibrium point was achieved by applying a distributed update algorithm, resulting in short-term fairness, low collision, and high throughput. To enhance both throughput and delay, the authors in [24] proposed an estimation-based algorithm that produces a new W value after each successful transmission or collision. This value is obtained by using the estimates of network load variations and the number of active nodes. The authors in [25] designed a *proportional-integral* (PI) controller for adapting $CW_{\min}$ based on the rates of retransmissions and successful transmissions. To increase network throughput and improve short-term fairness in saturated scenarios, Chun et al. [26] developed an algorithm that forecasts the number of active nodes. Using this information, they derived the optimal W settings for all nodes in the network. To guarantee low latency for real-time applications, Wang et al. [27] introduced an adaptation scheme based on deep neural networks. This scheme sets the Arbitrary Inter-Frame Space (AIFS) and $CW_{\min}$ values for all nodes based on the number of active nodes and changes in channel conditions. Theoretical analysis of the DCF protocol under both saturated and unsaturated settings where all nodes adopt the same $CW_{\min}$ have been studied in [28]. The former theoretical model has been adopted by Gao et al. to maximize network throughput in DCF [29] and EDCA [30] networks. Sun et al. presented an extension of the network throughput maximization problem by adopting a distributed parametric update of the contention window via estimation of channel occupancy times [31]. A trade-off analysis of throughput and channel access delay is presented in [32]. However, in the former works, the authors did not consider the presence of aggressive nodes in the network and the unfairness caused by such deviant players. To achieve high fairness and maximize the total network throughput, Syed et al. [33] introduced an adaptive backoff algorithm that computes a new W value based on channel state probabilities and the estimated number of active nodes. A modified DCF backoff process was proposed by Karaca et al. [34], in which the backoff counter is decreased based on the number of active nodes and channel idle periods. In [35], the collision rate between duty-cycled LTE-unlicensed (LTE-U) and WLANs transmission was reduced by using AI-based techniques. Their technique allows nodes to adapt the communication direction and transmission rates depending on LTE-U interference. To provide harmonious coexistence between Wi-Fi and LTE-LAA networks, authors in [36] Han et al. proposed a multi armed bandit solution to jointly select the contention window size in Wi-Fi AP and LTE BS networks, where they studied both cooperative and non cooperative variants. Compared to others, our work has no communication overhead between nodes due to its distributed nature, is technology agnostic, and generalizes well to diverse spectrum sharing scenarios.

ICW concept has also been applied by Kumar et al. [37], where a Reinforcement Learning (RL) module based on deep q-learning [38] is chosen for $CW_{\min}$ adaptation. They show throughput improvement for the intelligent node when all the other nodes adopt the same aggressive $CW_{\min}$ setting. Their assumption may not hold in most practical settings where each node is operating independently, hence may adopt to different $CW_{\min}$ settings. We show comparison results with this RL scheme in Section VI.

III. INTELLIGENT CONTENTION WINDOW (ICW) FRAMEWORK

Our goal is to adapt $CW_{\min}$ to achieve fair sharing of an unlicensed channel in the presence of aggressive nodes that manipulate their $CW_{\min}$ values. To reach this goal, nodes that decide to adapt their $CW_{\min}$'s (intelligent nodes) may choose two approaches for their adaptation mechanism. For the first approach, they could estimate the $CW_{\min}$ of all their neighboring nodes (as done in [6]) and calculate the channel access and collision probabilities for all nodes by solving a set of nonlinear equations derived from Markovian analysis of the CSMA/CA protocol. The former probabilities are then used to calculate intelligent nodes' expected airtime based on their selected $CW_{\min}$. Next, the $CW_{\min}$'s that bring intelligent nodes their fair expected channel airtime – this would have been generically achieved if all nodes were standard compliant – are selected for CSMA/CA. Alternatively, as for the second approach, the intelligent nodes could monitor the wireless channel and obtain statistics of the occupied channel airtime of all nodes. Using these statistics, intelligent nodes can determine whether they are getting their fair shares of the channel airtime based on a defined fairness criterion and adapt their $CW_{\min}$'s accordingly. In the second approach, an intelligent node does not need to estimate the $CW_{\min}$ of any other node and is not required to solve any nonlinear equations, rather, this node uses a heuristic machine learning approach, namely a random forest, to adapt its $CW_{\min}$ value.

A. SYSTEM MODEL

Our system model incorporates arbitrary numbers of intelligent, aggressive, and standard-compliant nodes. We characterize the state of the wireless channel by the $CW_{\min}$ values of all nodes sharing the unlicensed spectrum. We consider a spectrum sharing scenario of L nodes that share a wireless channel. Let $\mathcal{N} = \{N_1, \cdots, N_L\}$ denote the set of nodes. Nodes access the channel using CSMA/CA with exponential backoff. Let w_j be the $CW_{\min}$ used by node $N_j \in \mathcal{N}$. Depending on w_j, $\mathcal{N}$ includes three types of nodes: Well-behaving, aggressive, and intelligent. The well-behaving nodes select a default $\text{CW}_{\min}$ value (i.e., the standard value), while aggressive nodes select a small $\text{CW}_{\min}$ value to increase their airtime. We study how intelligent nodes should optimize their $\text{CW}_{\min}$ value to secure their fair share of airtime without causing significant impact on well-behaving nodes. To optimize the minimum size of contention window for intelligent nodes, we first need to characterize and formulate the utility achieved by each node in $\mathcal{N}$. The contention behavior in CSMA/CA makes it hard to characterize such utility using deterministic formulation. Therefore, we rely on the stochastic formulation and Markovian analysis, as discussed in [39]–[41]. Such analysis can be used to derive the occupied airtime as well as busy and idle time observed by each node in $\mathcal{N}$. Node N_j in $\mathcal{N}$ backs off for a random time that is capped by a value that is relevant to w_j. After each collision, the node enters a new *backoff stage* in which the upper cap of contention window is doubled, whereby the node has higher likelihood to backoff for a longer duration. Once the maximum retransmission attempt, say M, is reached, the upper cap of contention window is reset to w_j.

The backoff behavior can be modeled by a two-dimensional Markov process. The first dimension indicates the retransmission attempt, while the second dimension indicates the remaining backoff time, i.e., countdown process. Let $B_j(t) = \{s_j(t), b_j(t)\}$ be the two-dimensional Markov process that models the backoff behavior of node $N_j \in \mathcal{N}$, where $s_j(t) \in \{0, 1, ..., M\}$ denotes the backoff stage at time t, i.e., retransmission attempt, and $b_j(t)$ denotes the remaining time before accessing the channel; and it can take a value from $K_j(\iota) = \{0, 1, ..., W_j^{(\iota)} - 1\}$, where ι is the retransmission attempt. When $b_j(t)$ becomes zero, the node can access the channel. Let τ_j be the probability that $b_j(t)$ becomes zero, i.e., the probability of a channel access attempt. The channel becomes busy when one or more nodes attempt to access the channel. Let p_j be the probability of observing a busy channel by node N_j, which can be expressed as follows:

$$p_j = 1 - \prod_{\{N_k \in \mathcal{N} \setminus N_j\}} (1 - \tau_k). \tag{1}$$

To determine τ_j, we consider the Markov Chain (MC) that corresponds to the process $B_j(t)$, as shown in Figure 3. Let $\Pr_j[\iota', k' | \iota, k]$ be the transition probability of this MC from state (ι, k) to state (ι', k'). In line with [41], we can formulate the transition probabilities of $B_j(t)$ as follows:

$$\begin{cases} \Pr_j[0, k|\iota, 0] = \frac{1-p_j}{w_j}, & \iota \in \{0, \cdots, M-1\}, k \in K_j(0) \\ \Pr_j[0, k|\iota, 0] = \frac{1}{w_j}, & \iota = M, k \in K_j(0) \\ \Pr_j[\iota, k|\iota, k] = p_j, & \iota \in \{0, \cdots, M\}, k \in K_j(\iota) \setminus \{0\} \\ \Pr_j[\iota, k|\iota, k+1] = 1 - p_j, & \iota \in \{0, \cdots, M\}, k \in K_j(\iota) \setminus \{2^\iota w_j - 1\} \\ \Pr_j[\iota, k|\iota - 1, 0] = \frac{p_j}{2^\iota w_j}, & \iota \in \{1, \cdots, M\}, k \in K_j(\iota). \end{cases} \tag{2}$$

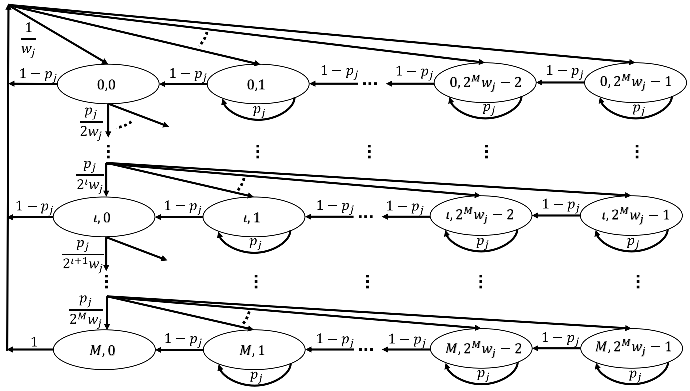


FIGURE 3: Markov chain transition state diagram for node N_j.

Let $\pi_j(\iota, k)$ be the steady-state probability of state (ι, k). By chain regularity, we can trace the steady-state probabilities of states in $\lim_{t\to\infty} B_j(t)$ back to the steady state probability of state $(0, 0)$, i.e., $\pi_j(0, 0)$, as follows:

$$\begin{cases} \pi_j(\iota, 0) = p_j^\iota \pi_j(0, 0) & \iota \in \{0, \cdots, M\}, \\ \pi_j(\iota, k) = \frac{2^\iota w_j - k}{2^\iota w_j (1-p_j)} \pi_j(\iota, 0) & \iota \in \{0, \cdots, M\}, k \in K_j(\iota). \end{cases} \tag{3}$$

By substituting (3) in $\sum_{\iota=0}^{M} \sum_{k \in K_j(\iota)} \pi_j(\iota, k) = 1$, we obtain an expression for $\pi_j(0, 0)$:

$$\pi_j(0, 0) = \left[\sum_{\iota=0}^{M} p_j^\iota \big(1 + \frac{1}{1-p_j} \sum_{\{k \in K_j(\iota) \setminus 0\}} \frac{2^\iota w_j - k}{2^\iota w_j}\big) \right]^{-1}. \tag{4}$$

Finally, we can formulate the channel access attempt probability, τ_j, by adding the steady-state probabilities of states that have zero backoff value:

$$\tau_j = \sum_{\iota=0}^{M} \pi_j(\iota, 0) = \frac{1 - p_j^{M+1}}{1 - p_j} \pi_j(0, 0). \tag{5}$$

B. OPTIMIZATION OF CW_{min}

To formulate the utility of fair airtime for intelligent nodes over a fairly long time window T, we need to consider three quantities of interest for a contending node during T: *Expected channel occupancy time* $\bar{T}^{(o)}$, *expected channel busy time* $\bar{T}^{(b)}$, and *expected channel idle time* $\bar{T}^{(i)}$. The first quantity can be expressed as follows:

$$\bar{T}_j^{(o)} = \tau_j T. \tag{6}$$

Let $p_j^{(b)}$ be the probability that node N_j freezes its backoff counter. Then, $p_j^{(b)} = (1 - \tau_j) p_j$. We can find the expected time that the channel is sensed to be occupied by nodes, other than N_j, $\bar{T}_j^{(b)}$ as:

$$\bar{T}_j^{(b)} = p_j^{(b)} T = (1 - \tau_j) \Big[1 - \prod_{\{N_k \in \mathcal{N} \setminus N_j\}} (1 - \tau_k) \Big] T. \tag{7}$$

The channel remains idle when all nodes in $\mathcal{N}$ have a non-zero backoff counter or have an empty transmission buffer, and this happens with probability $p^{(i)} = \prod_{\{k \in \mathcal{N}\}}(1 - \tau_k)$. Therefore, the expected time the channel is sensed to be idle $\bar{T}^{(i)}$ by any node in $\mathcal{N}$ can be expressed as:

$$\bar{T}^{(i)} = p^{(i)}T = T \prod_{\{N_k \in \mathcal{N}\}} (1 - \tau_k). \tag{8}$$

We define the *utility* $\bar{U}_j$ for node N_j to be its *expected normalized channel occupancy*, which can be expressed as:

$$\bar{U}_j = \frac{\bar{T}_j^{(o)}}{\bar{T}_j^{(o)} + \bar{T}_j^{(b)} + \bar{T}^{(i)}} = \tau_j. \tag{9}$$

Our goal is to let each intelligent node achieve its fair share of channel airtime. Hence, we define the *fair-optimal utility* $\bar{U}^*$ for a node in $\mathcal{N}$ as:

$$\bar{U}^* = \frac{1}{L} + \frac{\bar{T}^{(i)}}{L(\bar{T}_j^{(o)} + \bar{T}_j^{(b)} + \bar{T}^{(i)})} = \frac{1}{L}\left(1 + \prod_{\{N_k \in \mathcal{N}\}} (1 - \tau_k)\right). \tag{10}$$

The first term in (10) represents the fair portion of normalized channel airtime that a node should achieve when nodes have saturated traffic loads, whereas the second term corresponds to additional occupancy time that can be allocated to a node under unsaturated traffic scenarios. This improves network utilization by allocating idle channel time to intelligent nodes without harming the performance of other nodes. Thus, we can formulate the objective function for node N_j as the absolute difference between its utility and the fair-optimal utility:

$$F_j = \left|\bar{U}_j - \bar{U}^*\right| = \left|\tau_j - \frac{1}{L} \prod_{\{N_k \in \mathcal{N}\}} (1 - \tau_k) - \frac{1}{L}\right|. \tag{11}$$

Our goal is to find the optimal setting of $\mathrm{CW_{min}}$ that minimizes the objective functions of intelligent nodes. This can be obtained by minimizing the sum of their objective functions, i.e., $\sum_{\{N_j \in \mathcal{N}_e\}} F_j$, as follows:

$$\begin{aligned} P_1 :& \arg\min_{\mathcal{W}_e} \sum_{\{N_j \in \mathcal{N}_e\}} \left|\tau_j - \frac{1}{L} \prod_{\{N_k \in \mathcal{N}\}} (1 - \tau_k) - \frac{1}{L}\right|, \\ s.t. & \quad w_j \in \Omega \end{aligned} \tag{12}$$

where $\mathcal{N}_e \subset \mathcal{N}$ is the set of L_e intelligent nodes and $\mathcal{W}_e = \{w_1, \cdots, w_{L_e}\}$ is the set of decision variables, i.e., $\mathrm{CW_{min}}$'s, that can take values in $\Omega = \{w^{(\min)}, w^{(\min)}+1, \cdots, w^{(\max)}\}$. The above optimization problem is an integer nonlinear program. In principle, it can be solved by relaxing decision variables and applying nonlinear programming techniques. However, such an approach is challenging due to the following reasons:

1. Obtaining the global solution requires strict coordination and synchronization among intelligent nodes. Even if such coordination were to be achieved through protocol design, it still incurs communication overhead among intelligent nodes.
2. To compute the objective function, one must have global knowledge of aggressive and well-behaving nodes, including their $\mathrm{CW_{min}}$ values and traffic loads. Obtaining such global knowledge is hard to achieve in practice.
3. The optimization problem needs to be resolved repeatedly whenever changes in the wireless environment or the $\mathrm{CW_{min}}$ settings of aggressive and well-behaving nodes occur.

Analytical solution to such a problem is difficult. Instead, we employ machine learning to address this problem for a large number of settings and scenarios. An ML module can be trained to map existing solutions of this problem and ensure they generalize to obtain solutions under new settings. This allows for solving the problem in a distributed fashion where intelligent nodes can act independently.

IV. MACHINE LEARNING SOLUTION

To obtain an ML solution to (12), each intelligent node needs to empirically estimate its objective function. To do so, intelligent nodes independently gather empirical observations of channel occupancy and idle times.

A. EMPIRICAL MODELING OF THE OBJECTIVE FUNCTION

To estimate the quantities in (9), each intelligent node can monitor the channel and build sufficient statistics to estimate the activities of neighboring nodes. Let $\widetilde{T}_j^{(o)}$ be the empirical estimation of normalized channel occupancy for node $N_j \in \mathcal{N}_e$, and let $t_q^{(o)}$ be the duration of its channel occupancy time during the qth channel access attempt, $q = 1, 2, \cdots$. For $n^{(o)}$ channel access attempts during T, we can express $\widetilde{T}_j^{(o)}$ as follows:

$$\widetilde{T}_j^{(o)} = 1/T \sum_{q=1}^{n^{(o)}} t_q^{(o)}. \tag{13}$$

Similarly, let $t_q^{(b)}$ be the qth busy channel duration sensed by N_j. For $n^{(b)}$ channel busy events during T, we can write the empirical estimation of the normalized channel busy time $\widetilde{T}_j^{(b)}$:

$$\widetilde{T}_j^{(b)} = 1/T \sum_{q=1}^{n^{(b)}} t_q^{(b)}. \tag{14}$$

The empirically estimated normalized channel idle time $\widetilde{T}_j^{(i)}$, $N_j \in \mathcal{N}$ can be expressed as:

$$\widetilde{T}^{(i)} = 1 - \left(\widetilde{T}_j^{(b)} + \widetilde{T}_j^{(o)}\right). \tag{15}$$

Similarly, we can express the empirical version of the utility in (9), $\widetilde{U}_j$, as:

$$\widetilde{U}_j = \widetilde{T}_j^{(o)}, \tag{16}$$

and the *fair-optimal utility* of (10) can be empirically expressed as:

$$\widetilde{U}^* = \frac{1}{L} + \frac{\widetilde{T}^{(i)}}{L}. \tag{17}$$

From node a fairness perspective, $\text{CW}_{\min}$ should be set such that N_j receives $\frac{1}{L}$ fraction of T plus $\frac{1}{L}$ portion of the time it senses the channel to be idle, i.e., $\widetilde{T}^{(i)}/L$. This way, an intelligent node is motivated to exploit the idle time and access the channel more frequently when the network is unsaturated. Accordingly, the *empirical objective function* $\widetilde{F}_j$ of node N_j can be expressed as:

$$\widetilde{F}_j = \left|\widetilde{U}_j - \widetilde{U}^*\right|. \tag{18}$$

For node N_j, we can exhaustively test all possible w_j values in Ω and select the one that minimizes (18). Next, we explain how we can take advantage of previous empirical estimations to construct features and use them to learn solutions of different instances of (12) and (18).

B. CHANNEL STATE MODELING AND FEATURE DESIGN

Depending on $\text{CW}_{\min}$ values used by neighboring nodes, we have different instances of problem (12). We characterize each instance by a *channel state*, S, where the space of channel states includes all possible $\text{CW}_{\min}$ combinations used by neighboring nodes:

$$S = \{w_k | N_k \in \mathcal{N} \backslash N_j\}. \tag{19}$$

It should be noted that the state of channel is not fully observable by the intelligent node N_j, but it can still acquire partial knowledge about it; thanks to empirical estimations in (13), (14), and (15). Thus, the intelligent node can construct a feature that indirectly characterizes the unique state of the channel. Let v_j be the feature vector constructed by node N_j over time window T, where:

$$v_j = \langle \widetilde{T}_j^{(o)}, \widetilde{T}_j^{(b)}, \widetilde{T}^{(i)}, L, w_j \rangle. \tag{20}$$

We can assign to each feature the recommended $\text{CW}_{\min}$ value w_j^* that minimizes the value in (18), and consider w_j^* to be the assigned label. Then, we train an ML module, e.g., a random forest, to learn the mapping of the feature vector v_j and label w_j^*. However, after training, the ML module will not necessarily output exact optimal labels w_j^*'s, we denote the recommended $\text{CW}_{\min}$ by node N_j's ML module after training by $\hat{w}_j$. It should be noted that v_j works as a proxy to characterize the instance of problem (18) to be solved, while $\hat{w}_j$ works as the recommended solution to this instance. We next explain how we can construct features and labels for learning solutions of problem (18). The set of best $\text{CW}_{\min}$ values (ω^*) are the labels that are used for supervised training. A key concept to keep in mind is that there is no need to train the random forest for all possible channel states (S); instead, training offline over a small well-representing subset of the whole possible channel states is sufficient for obtaining generalized solution to (18) over new untrained/unobserved channel states. We next discuss how such well-representing training dataset can be constructed.

C. TRAINING DATA CONSTRUCTION

Going forward, without loss of generality, we drop the j subscript used to denote parameters associated to an intelligent node $N_j \in \mathcal{N}_e$, and bring it back whenever it is needed. We develop a discrete-event simulator to model the CSMA/CA channel access and generate data for training the solution of problem (18) using a C++ library called CSIM [42]. The dataset corresponds to a large set of feature vectors and their optimal $\text{CW}_{\min}$ values, i.e., labels (w^*'s). After training, we expect the ML module to output $\hat{w}$ that is as close as possible to w^*. The feature vector includes the set of observations that N_j monitors, during an observational window of length T, along with the channel access parameters it uses during the monitoring period. To obtain the label for a particular state S, we gradually increase the $\text{CW}_{\min}$ value that node N_j uses, i.e., w, in Ω, and monitor the observations needed to compute the empirical utility in (16). We then select w^* value that minimizes $\widetilde{F}$ as a label. For each $w \in \Omega$, we also keep track of features in (20) and save them in feature set $\mathcal{V}$.

Algorithm 1 explains how a feature set $\mathcal{V}$ and optimal labels w^* can be constructed for a given wireless channel state S, Ω, and a monitoring window T. For each setting of S and Ω, a set $\mathcal{V}$ and a label w^* will be created by Algorithm 1. We run the algorithm for different settings of S and gather different $\mathcal{V}$'s and w^*'s, and include them all in a training set denoted by $\mathcal{R}$.

Algorithm 1 Dataset construction for S, Ω, and T

Input S, Ω, and T;
Variables: $w \leftarrow w^{(\min)}$, $w^* \leftarrow w^{(\min)}$, $\widetilde{F}^* \leftarrow 1$, $\mathcal{V} = \{\}$;
Outputs: $\mathcal{V}$ and w^*;
while $w \leq w^{(\max)}$ **do**
  Run the CSMA/CA simulator for T seconds, and gather statistics as observed by node N_j:
  $v \leftarrow \langle \widetilde{T}^{(o)}, \widetilde{T}^{(b)}, \widetilde{T}^{(i)}, L, w \rangle$;
  $\widetilde{U} \leftarrow \widetilde{T}^{(o)}$;
  $\widetilde{U}^* \leftarrow \frac{1}{L} + \frac{\widetilde{T}^{(i)}}{L}$;
  $\widetilde{F} \leftarrow \left|\widetilde{U} - \widetilde{U}^*\right|$;
  if $\widetilde{F} \leq \widetilde{F}^*$ **then**
    $w^* \leftarrow w$;
    $\widetilde{F}^* \leftarrow \widetilde{F}$;
  end if
  $w \leftarrow w + 1$;
  Add v to $\mathcal{V}$;
end while
Return $\mathcal{V}$ and w^*;

D. EXAMPLE OF TRAINING DATA GENERATION

We provide an example that shows the observations made by N_1 when it shares the channel with two other nodes, N_2 and N_3, as shown in Figure 4. During T, N_1 finds $L = 3$, $n_1^{(o)} = 2$, and $n_1^{(b)} = 3$. It calculates the first three elements of v_1 based on (13), (14), and (15).

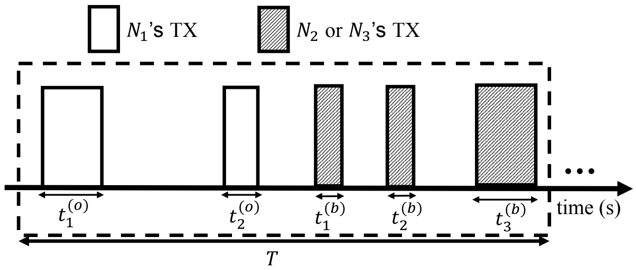


FIGURE 4: Example of monitoring the channel over a period of T seconds (network of three nodes, where N_1 is an intelligent node).

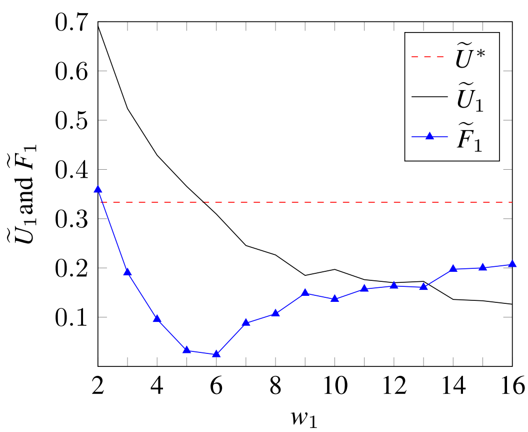


FIGURE 5: $\widetilde{U}_1$ and $\widetilde{F}_1$ as observed by node N_1 vs. $\mathrm{CW}_{\min}$ for N_1 ($L = 3$, $S = \{9, 4\}$, and saturated traffic).

Figure 5 depicts the process of obtaining best $\mathrm{CW}_{\min}$ value to be used by N_1 (i.e., the label) when N_2 has $w_2 = 9$ and N_3 has $w_3 = 4$. We set the traffic intensity of all nodes to 300 frames/sec (fps), which results in heavy loaded buffers at all three nodes, i.e., $\widetilde{T}^{(i)} \approx 0$, hence the optimal empirical utility $\widetilde{U}^*$ is $\frac{1}{3}$ (shown by the dashed red line in Figure 5). We vary w_1 from 2 to 16 and monitor $\widetilde{U}_1$. As it can be observed in Figure 5, $\widetilde{F}_1$ has its lowest value at $w_1 = 6$, hence $w_1^* = 6$. Accordingly, $w_1^* = 6$ is the label for the state $S = \{9, 4\}$. Different states can have different w_1^* values, as shown in [43].

V. ML MODULE DESIGN

A. CONSTRUCTING A DECISION TREE AND A RANDOM FOREST

As discussed in Section IV-C, $\mathcal{R}$ is the set of training samples. A decision tree of depth d_T divides the feature space into 2^{d_T} distinct and non-overlapping regions, $O_1, O_2, ..., O_{2^{d_T}}$, where each region corresponds to a particular class. Each class represents the set of all feature vectors, as in (20), that are associated with the same label w^*. Samples of one class could be part of multiple regions, since the total number of regions, 2^{d_T}, could be larger than the total number of classes. These regions are the leaves of the decision tree. In order to have a fast training phase, we use *recursive binary splitting (RBS)* algorithm to build the decision trees [44]. Before explaining RBS, first we introduce the *Gini index*. Consider an arbitrary set r of samples that potentially belong to different classes, i.e., $r \subset \mathcal{R}$. The Gini index $G(r)$ of the set r can be expressed as:

$$G(r) = \sum_{k=1}^{C} \rho_{k,r}(1 - \rho_{k,r}), \tag{21}$$

where C is the number of classes to which samples in the set r belong, and $\rho_{k,r}$ is the proportion of training samples that belong to class k and are also in the r set. Gini index measures the dispersion (impurity) of the samples in the set r. A low $G(r)$ value indicates that the samples in r are more likely to belong to the same class.

To build a decision tree, we start with the root of the tree and look for a feature $v[j]$, where $v[j]$ is the jth feature in the feature vector v, and a cut-point value ϕ_j that splits $\mathcal{R}$ into two subsets $r_1 = \{v : v[j] \leq \phi_j, v \in \mathcal{R}\}$ and $r_2 = \{v : v[j] > \phi_j, v \in \mathcal{R}\}$ such that the value $G(r_1) + G(r_2)$ is minimized. Each of these subsets is represented by an internal node below the root. The splitting process is recursively repeated for each subset, i.e., splitting them into two new subsets, such that the sum of the Gini indices over them is minimized. This way, internal nodes become parents to new nodes beneath them. For instance, we can split r_1 into two new subsets $r_{11} = \{v : v[i] \leq \phi_i, v \in r_1\}$ and $r_{12} = \{v : v[i] > \phi_i, v \in r_1\}$, where the feature $v[i]$ and cut-point value ϕ_i are selected to minimize the sum $G(r_{11}) + G(r_{12})$. The splitting process is continued until the depth of the decision tree is d_T.

A random forest of depth d_F consists of d_F decision trees. These trees are constructed as explained before, but for each split we only consider $\lfloor \sqrt{N_f} \rfloor$ random features, where N_f is the number of features in v. In our case, $\lfloor \sqrt{N_f} \rfloor = \lfloor \sqrt{5} \rfloor = 2$. After training, feature classification is done by feeding the features in (20) to each tree and obtaining d classification results. The final classification result of the random forest is the mode of the d individual classifications.

To train our ML module, we construct multiple datasets for large number of states and different number of nodes. We gather simulation observations for the former settings using our developed discrete-event-based simulator that uses classes and functions for synchronizing and generating process-oriented events. Table 1 shows the configuration for each of our simulated datasets. Ω is set to $\{2, \cdots, 16\}$ for all the datasets. In some of our training tasks, we use a combination of datasets, e.g., we denote the composite dataset from D_1, D_3 and D_5 by $D_{1|3|5}$.

B. HYPERPARAMETER SETTINGS

The depth of the random forest d_F, depth of each tree d_T, and the observation window T duration are hyperparameter settings that influence the random forest's prediction accuracy of the optimal labels (w^*). In this section, we discuss the appropriate settings of these values. Moreover, our results indicate that the throughput performance is not much impacted

TABLE 1: Generated datasets using Algorithm 1.

Dataset	L	T (sec)	# Unique S's	# Possible S's
D_1	3	5	300	15^2
D_2	3	10	300	15^2
D_3	6	5	300	15^5
D_4	9	5	300	15^8
D_5	10	5	1216	15^9

TABLE 2: Feature importance measures.

Feature	DC AD = 0 (%)	MID (%)
$\widetilde{T}^{(o)}$	0.02	22.03
$\widetilde{T}^{(b)}$	0.34	24
$\widetilde{T}^{(i)}$	0.7	22.35
L	2.88	4.78
w	22.28	26.84

by small deviations from w^*. A misclassification of optimal w^* by one or two drifts, i.e., $|\hat{w} - w^*| \leq 2$, results in near-optimal performance. We take advantage of this and relax the prediction accuracy requirement by considering CW_{min} values that are 1 or 2 apart from w^* to be sufficiently accurate. Formally, we define the *Acceptable Drift (AD)* as the largest deviation that a recommended $\hat{w}$ can have from its true label w^* to be considered as an accurate prediction. To find the classification accuracy of a model trained on a specific dataset, D_j, we consider 67% and 33% of the states for training and testing, respectively. This makes sure that we only test the performance of the model on observations collected from states that the model was not trained on, which gives a more accurate estimate of the generalization performance for unobserved states.

1) Random Forest Design

To design the architecture of the ML module, we need to find the values of d_F and d_T for the random forest that are as small as possible. This ensures that our model has low computational complexity, and high desirable prediction accuracy. We present the accuracy of random forest vs. d_F and d_T, when trained and tested on $D_{1|3|5}$ for AD values of 0, 1, and 2, as shown in Figure 6. It can be seen that by setting $d_F = 20$ and $d_T = 20$, random forest's classification accuracy is 69.24%, 96.8%, and 99.61%, when AD is 0, 1, and 2, respectively.

2) Selection of Observation Window T

The ML module uses current window statistics (channel observations) to produce a CW_{min} value for the next window. This process is valid if the state of the wireless channel (S) stays fixed during consecutive monitoring windows; however, when the network is dynamic (dynamic S), it is important to have a small monitoring window T to track the dynamics appropriately (see evaluation results for dynamic aggression scenarios in Section VI-C2). Nonetheless, the value of T also needs to be set large enough to gather sufficient statistics to produce appropriate CW_{min}'s. We consider a random forest with $d_F = d_T = 20$, and train it on D_2. In Figure 7, we plot prediction accuracy of the ML module for T values ranging from 0.5 to 10 sec[1] while having AD of 0, 1, and 2. From this figure, we see that when $T = 5$ sec the prediction accuracy for ADs of 1 and 2 are 91.12% and 98.71%, respectively. Thus, we choose $T = 5$ sec throughout our evaluations, meaning that all intelligent nodes will predict their CW_{min} values based on the observed statistics that they have monitored during the prior 5 sec. Figure 8 depicts the histogram of the deviation of CW_{min} predictions from their w^* labels, when $T = 5$ sec. This figure shows that by setting AD to 2, 98.71% of CW_{min} predictions are accurate.

C. FEATURE IMPORTANCE

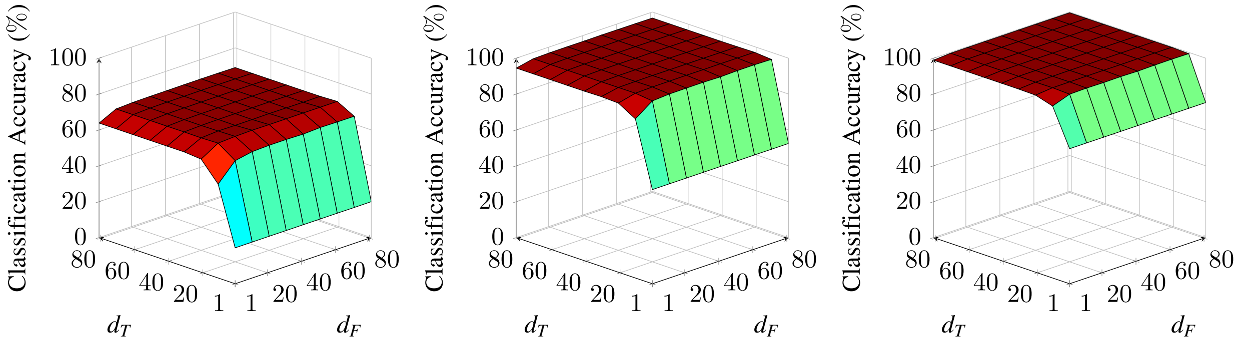
To bring some insight into our ML module's prediction logic, we introduce two measures to evaluate the importance of each feature for optimal CW_{min} estimation. These findings help relax the number of statistics that a node needs to observe during T. The first importance measure is called the *Drop Column (DC)* measure, which requires retraining the model from scratch for importance evaluation of each feature. To evaluate the importance of a feature in v, we drop that feature from the feature vectors and retrain the model and calculate its classification accuracy. The features are ranked based on the performance drop that they cause when they are dropped from v. The second measure is called the *Mean Impurity Decrease (MID)*, which does not require the model to be retrained for each feature evaluation, rather, MID is calculated during one training phase, for all features in v. This measure is the decrease in average impurity (e.g., Gini Index, see Section V-A) of a feature over all the internal node splits during the construction of the random forest. The most important feature gives the most mean decrease in impurity. We evaluate MID and DC importance of all features used by the random forest trained on $D_{1|3|5}$, as shown in Table 2.

MID favors features that take continuous real values, i.e., $\widetilde{T}^{(o)}$, $\widetilde{T}^{(b)}$, and $\widetilde{T}^{(i)}$. DC is a more representative importance measure than MID, but requires retraining for each importance evaluation. Based on DC measures it can be seen that w is the most important feature followed by L. We can drop either of $\widetilde{T}^{(o)}$, $\widetilde{T}^{(b)}$, or $\widetilde{T}^{(i)}$ from our feature vector and expect the random forest to perform as well as before, and reduce processing overhead and ML module complexity. In section VI, the intelligent nodes do not use $\widetilde{T}^{(i)}$ as one of their features in their ML module, which is also trained on the set of features that do not include $\widetilde{T}^{(i)}$.

VI. PERFORMANCE EVALUATION

In this section, we conduct extensive evaluations under both simulated and real-world scenarios. We use OTA experiments to validate our simulation findings and rely on simulations to conduct extensive evaluations. Our OTA experiments are based on Software Defined Radios (SDRs). We use National Instruments (NI) SDRs NI-USRP 2944r, 2942r, and Flexrio

[1] D_2 corresponds to observations gathered through 10 sec, but for lower observational windows, we gather observations and construct data-subsets from D_2 depending on the value of T.



(a) AD = 0 (b) AD = 1 (c) AD = 2

FIGURE 6: Random forest classification accuracy vs. d_F and d_T, when (a) AD = 0, (b) AD = 1, and (c) AD = 2.

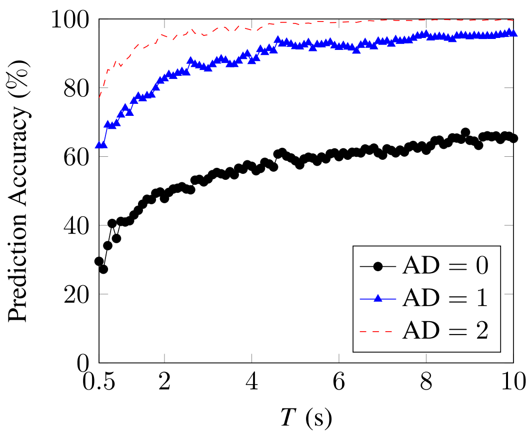


FIGURE 7: Prediction accuracy of the ML module vs. the monitoring period, T.

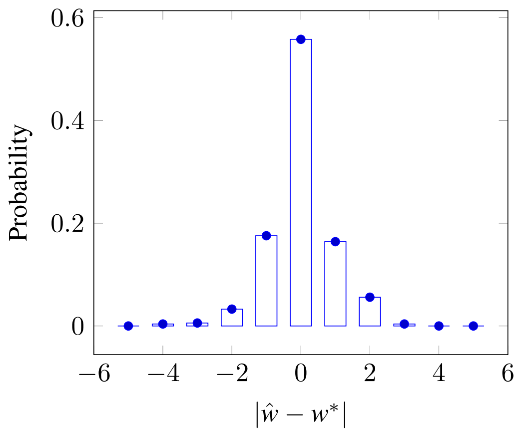


FIGURE 8: Histogram of the deviation of CW_{min} predictions from their w^* labels.

5791, along with NI LabVIEW 802.11 application framework [45]. For our simulation setups, we run simulations using the same simulator used in Section IV-C. We first present our evaluations for a single intelligent node and later present multiple intelligent node scenarios operating under both static and dynamic aggressive settings. We also show multiple aggressor scenarios. In all OTA and simulation experiments, the intelligent nodes utilize a random forest that is trained on 67% of states in $D_{1|2|5}$, which is a simulated dataset. We compare ICW with two other mechanisms for controlling CW_{min}. One is the standard *DCF*, where a node fixes its CW_{min} value to 16. In the second mechanism, intelligent nodes will use the RL module presented in [37] to adapt their CW_{min}. The RL module is also trained on the same dataset as our random forest module, i.e., 67% of states in $D_{1|2|5}$.

A. SINGLE INTELLIGENT NODE (OTA USRP EXPERIMENTS)

To evaluate the performance of ICW in practice, we conduct a set of experiments using NI-USRP 2944r, 2942r, and FlexRio 5791. To modify the CW_{min} value of a radio, we change the MAC layer FPGA code of the LabVIEW 802.11 Application Framework. Due to our hardware limitations, we consider three stations sending traffic to a common AP. Figure 9 shows our experimental setup. N_2 chooses its CW_{min} from $\{4, 8, 16\}$, and can act aggressively. N_3 is a well-behaving node and fixes its CW_{min} to 16. All three stations are approximately 2 meters away from the AP. In Figure 10, we show per-node uplink throughput vs. CW_{min} of N_1 for all three possible CW_{min} settings of N_2. Over all scenarios, it can be seen that low CW_{min} improves aggressive node's throughput but harms the performance of other nodes. Thus, it is important to choose a CW_{min} that obtains a fair throughput share and is considerate of the number of nodes sharing the wireless channel.

To compare ICW with DCF and RL CW_{min} selection mechanisms, we select N_1 as an intelligent node, while keeping the former configurations for N_2 and N_3. Figure 11 shows per-node uplink throughput for ICW, DCF, and RL CW_{min} selection mechanisms under different CW_{min} settings of N_2. It can be observed that ICW helps N_1 always get its fair share of throughput ($\sim 35.59\%$), increasing its throughput by $5.62\times$ compared to the DCF mechanism, when $w_2 = 4$ or $w_2 = 8$. On the other hand, the RL mechanism provides

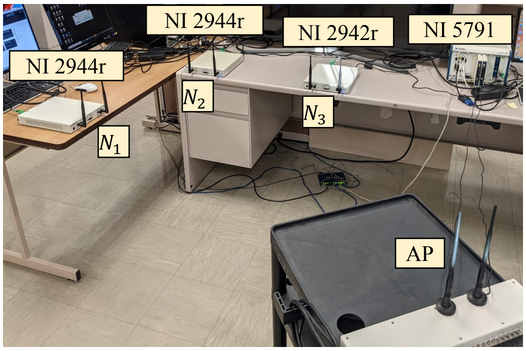


FIGURE 9: USRP experimental setup.

an unfair boost in throughput for N_1 ($\sim 87.68\%$ of available throughput). We can also conclude that when both N_2 and N_3 abide by the standard settings, choosing a $\text{CW}_{\min}$ of 16, N_1 under ICW behaves as a standard node and provides similar performance as the DCF mechanism, whereas under the RL scheme, N_1 behaves aggressively and degrades fairness. These experiments used ML models that were trained on simulated datasets and they support the feasibility claim of deploying ICW modules in real-world applications that are trained on simulated datasets, which are prepared offline.

To evaluate multiple aggressors and multiple intelligent nodes, and due to hardware constraints, we continue our evaluations based on datasets generated using our CSMA/CA discrete event simulator.

B. MULTIPLE AGGRESSORS

1) $L = 3$ with two aggressive nodes

We select N_1 as our single intelligent node. Figures 12 and 13 depict the per-node uplink throughput and per-frame latency, respectively, for $S = \{2,2\}$, $\{4,2\}$, and $\{16,2\}$ under the three $\text{CW}_{\min}$ selection mechanisms, i.e., ICW, DCF, and RL. On average, under ICW, node N_1 achieves 594% throughput gain over what it gets under DCF and RL mechanisms. In Table 3, we present the corresponding Jain's index [46], calculated over throughput, under the three mechanisms. ICW improves fairness by 43.32% compared to DCF and RL. Moreover, under ICW, node N_1's latency is 87.11% less than its latency under DCF and RL[2]. It can be noted that the RL results are identical to the DCF's, which is inconsistent with the results in Section VI-A. Whereas, ICW maintains fairness in booth simulation and real-world scenarios when trained on simulated data.

2) $L = 10$ with varying number of aggressive nodes

We evaluate the effect of multiple aggressive nodes in dense scenarios. Consider $L = 10$ with one intelligent node N_1. We consider up to three aggressive nodes, setting their $\text{CW}_{\min}$ to 2, while the remaining nodes select $\text{CW}_{\min} = 16$. Throughout

[2]Going forward, we will show only the throughput performance results, since the per-frame latency is inversely proportional to the uplink throughput when nodes have saturated transmission buffers.

TABLE 3: Jain's index for different S values over different $\text{CW}_{\min}$ selection mechanisms.

S	$\{2,2\}$	$\{4,2\}$	$\{16,2\}$
ICW's Jain's Index	0.99	0.82	0.66
DCF's Jain's Index	0.72	0.59	0.43
RL's Jain's Index	0.72	0.59	0.43

our evaluations, we observe that nodes with same $\text{CW}_{\min}$ value have similar throughput performance, therefore, we choose to show the average throughput value for these nodes. We denote the average performance of aggressive and standard nodes by N_A and N_S, respectively. Figure 14 shows the average uplink throughput results when node N_1 selects ICW, DCF, and RL as its $\text{CW}_{\min}$ adaptation mechanism. Under ICW, on average, when aggressive nodes exist, the intelligent node's throughput is increased $6.69\times$ compared to DCF. On average, under ICW and RL N_1 achieves 11.57% and 37.75% of total throughput, which indicates ICW as a fairer mechanism than RL.

C. MULTIPLE INTELLIGENT NODES

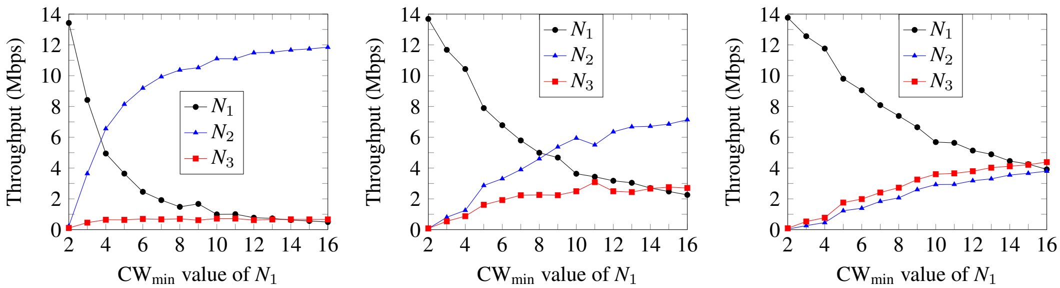
Our model accounts for multiple intelligent nodes. We study the performance of two intelligent nodes for two types of aggressive behavior: *Static Aggression (SA)* and *Dynamic Aggression (DA)*. For SA, the aggressor sets its $\text{CW}_{\min}$ to a fixed value from the set $\{2,8,12\}$ during the simulation experiment. For DA, the aggressor randomly selects its $\text{CW}_{\min}$ from $\{2,8,12\}$, while it can change its $\text{CW}_{\min}$ slowly or fast relative to the monitoring period T. Due to inconsistent and unfair results that were provided by the RL $\text{CW}_{\min}$ selection mechanism in the previous sections, we continue the performance comparison of ICW with the benchmark DCF scheme, only.

1) $L = 3$ – Static Aggression

We consider two intelligent nodes (N_1 and N_2) and one aggressive node (N_3). Initially, the $\text{CW}_{\min}$ value of intelligent nodes is set to 16. N_1 and N_2 sequentially update their $\text{CW}_{\min}$ values each 10 seconds based on their latest 5 seconds of monitored observations. In Figures 15(a), 15(b), and 15(c), we plot $\text{CW}_{\min}$ for nodes N_1 and N_2 vs. time, when N_3 has a $\text{CW}_{\min}$ value of 2, 8, and 12, respectively. In Figures 15(d), 15(e), and 15(f), we plot the respective per-node uplink throughput vs. time. From these figures, we observe that each node achieves its equal share of the total network throughput of about 4.2 Mbps. Table 4, presents the averaged Jain's index for the three $\text{CW}_{\min}$ selections of node N_3 under the two mechanisms. When N_3 chooses a $\text{CW}_{\min}$ of 2, DCF mechanism falls short in providing good fairness; on the other hand, ICW maintains a Jain's index of 0.99 for all cases and improves the throughput of intelligent nodes by $4.9\times$.

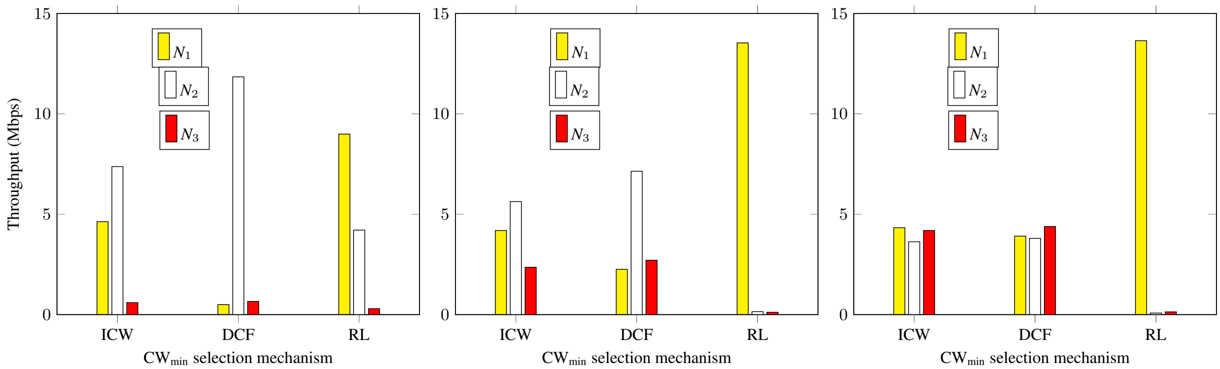
2) $L = 6$ – dynamic aggression

We set nodes N_1 and N_2 as intelligent nodes and N_3 as an aggressive node, while nodes N_4, N_5, and N_6 fix their $\text{CW}_{\min}$



(a) N_2's $CW_{min} = 4$ (b) N_2's $CW_{min} = 8$ (c) N_2's $CW_{min} = 16$

FIGURE 10: Per-node uplink throughput vs. N_1's CW_{min} when N_2 has a CW_{min} value of (a) 4, (b) 8, and (c) 16.



(a) N_2's $CW_{min} = 4$ (b) N_2's $CW_{min} = 8$ (c) N_2's $CW_{min} = 16$

FIGURE 11: Per-node uplink throughput under three mechanisms for setting CW_{min} at N_1 , when N_2 sets its CW_{min} to (a) 4, (b) 8, and (c) 16.

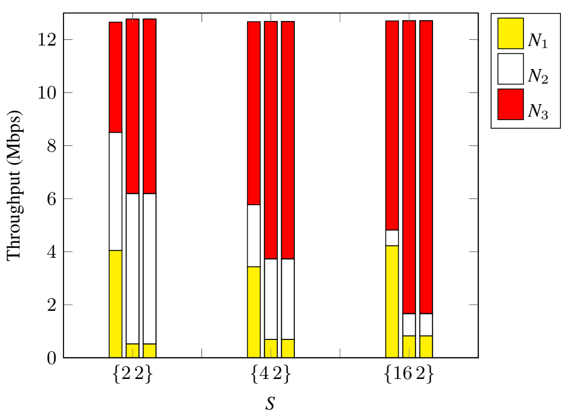


FIGURE 12: Upnlink throughput under three mechanisms for setting CW_{min} at node N_1 and three different values for S. “Color only”, “angled lines”, and “dots” patterns represent ICW, DCF, and RL, respectively.

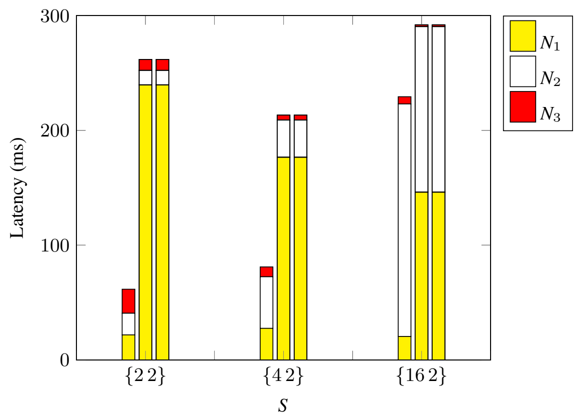


FIGURE 13: Per-frame latency under three mechanisms for setting CW_{min} at node N_1 and three different values for S. “Color only”, “angled lines”, and “dots” patterns represent ICW, DCF, and RL, respectively.

values to 16, behaving as standard nodes. In Figure 16, we are showing throughput results for two the intelligent nodes (N_1 and N_2), the aggressive node (N_3) and the average performance of well-behaving nodes, i.e., N_s. It can be observed that under ICW, the intelligent nodes always get their fair share of throughput, i.e., 2.1 Mbps.

VII. CONCLUSIONS

In this work, we considered the problem of fair channel access in CSMA/CA systems that include aggressive devices. Such

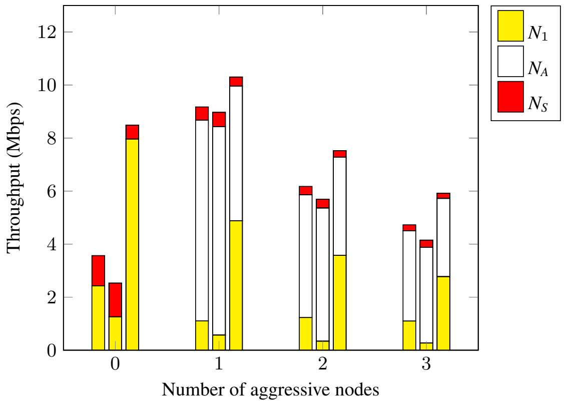


FIGURE 14: Upnlink throughput under three mechanisms for setting $\mathrm{CW}_{\mathrm{min}}$ at node N_1 vs. number of aggressive nodes. "Color only", "angled lines", and "dots" patterns represent ICW, DCF, and RL, respectively.

TABLE 4: Jain's index for different $\mathrm{CW}_{\mathrm{min}}$ values of N_3 over different $\mathrm{CW}_{\mathrm{min}}$ selection mechanisms ($L = 3$ and saturated traffic).

N_3's $\mathrm{CW}_{\mathrm{min}}$	2	8	12
ICW's Jain's Index	0.99	0.99	0.99
DCF's Jain's Index	0.43	0.89	0.98

devices intentionally set their $\mathrm{CW}_{\mathrm{min}}$ to small values thus harming the performance of nodes with standard-compliant $\mathrm{CW}_{\mathrm{min}}$ settings. We presented the ICW framework in which nodes adapt their $\mathrm{CW}_{\mathrm{min}}$ values using an ML framework. Intelligent nodes gather observations of the channel, and feed them to an ML module to produce recommended $\mathrm{CW}_{\mathrm{min}}$ values. We also discussed the importance of choosing a proper monitoring window for gathering observation, which is able to both track network dynamics and gather sufficient statistics from the channel. We showed that ICW-enabled nodes will always get their fair share of throughput, regardless of the type of aggressive behavior that a greedy node can follow, e.g., static or dynamic aggression. An ML module was trained in a supervised manner and its labels were obtained based on a proportional factor that takes into account the idle channel time. We argued that by training the ML module only on a subset of the whole possible channel states, we would be able to achieve high generalization results for unseen channel states. OTA results showed that our ICW ML models, which were trained on simulated datasets, could provide the same performance benefits for real-world deployments as for simulated testbeds. It was also shown that we can further improve the network performance by equipping more nodes with ICW capabilities. ICW's ML module has low computational complexity and it gets trained fast relative to other deep learning models, e.g., DNN and CNN. The method proposed for designing its structure ensures that the ML module does not overfit and rather generalizes well. ICW is used in a distributed fashion to provide fairness among nodes operating under different technologies. Our simulations indicate that in highly aggressive scenarios, compared to the DCF mechanism, ICW increases increases fairness by 36.4%. Consequently, the throughput of an ICW-enabled node is improved by $5.96\times$ and its per-frame latency reduced by 87.11%.

REFERENCES

[1] Cisco, "White paper: Cisco visual networking index: Forecast and methodology," https://bit.ly/2wmdZJb, available online, accessed on 9/4/2023.

[2] M. Hirzallah, M. Krunz, B. Kecicioglu, and B. Hamzeh, "5G new radio unlicensed: Challenges and evaluation," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 3, pp. 689–701, 2021.

[3] "Quarkslab: Reverse-engineering broadcom wireless chipsets," https://blog.quarkslab.com/reverse-engineering-broadcom-wireless-chipsets.html, published: 2019-04-16.

[4] "Google Project Zero: Exploiting Broadcom's Wi-Fi Stack," https://https://googleprojectzero.blogspot.com/2017/04/over-air-exploiting-broadcoms-wi-fi_4.html, published: 2017-04-04.

[5] "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," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534.

[6] A.-H. Yazdani-Abyaneh and M. Krunz, "CWmin estimation and collision identification in Wi-Fi systems," in *Proc. of the IEEE MASS*, 2021, pp. 490–498.

[7] A. Yazdani Abyaneh, V. Pourahmadi, and A. Hosein Gharari Foumani, "CSI-based authentication: Extracting stable features using deep neural networks," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, p. e3795, 2020.

[8] A.-H. Yazdani-Abyaneh and M. Krunz, "Automatic machine learning for multi-receiver CNN technology classifiers," in *Proc. of the ACM WiseML*, 2022, p. 39–44.

[9] M. Schulz, "Teaching your wireless card new tricks: Smartphone performance and security enhancements through Wi-Fi firmware modifications," Ph.D. dissertation, Technische Universität, 2018. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/7243/

[10] M. Schulz, D. Wegemer, and M. Hollick, "Nexmon: Build your own Wi-Fi testbeds with low-level MAC and PHY-access using firmware patches on off-the-shelf mobile devices," in *in Proc. of the ACM WiNTECH*, 2017, p. 59–66.

[11] F. Gringoli, M. Schulz, J. Link, and M. Hollick, "Free your CSI: A channel state information extraction platform for modern Wi-Fi chipsets," in *in Proc. of the ACM WiNTECH*, 2019, p. 21–28.

[12] M. Schulz, J. Link, F. Gringoli, and M. Hollick, "Shadow Wi-Fi: Teaching smartphones to transmit raw signals and to extract channel state information to implement practical covert channels over Wi-Fi," in *in Proc. of the ACM MobiSys*, 2018, p. 256–268.

[13] M. Schulz, F. Gringoli, D. Steinmetzer, M. Koch, and M. Hollick, "Massive reactive smartphone-based jamming using arbitrary waveforms and adaptive power control," in *in Proc. of the ACM WiSec*, 2017, p. 111–121.

[14] M. Vanhoef and F. Piessens, "Advanced Wi-Fi attacks using commodity hardware," in *in Proc. of the ACM ACSAC*, 2014, p. 256–265.

[15] D. S. Berger, F. Gringoli, N. Facchi, I. Martinovic, and J. B. Schmitt, "Friendly jamming on access points: Analysis and real-world measurements," *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 6189–6202, 2016.

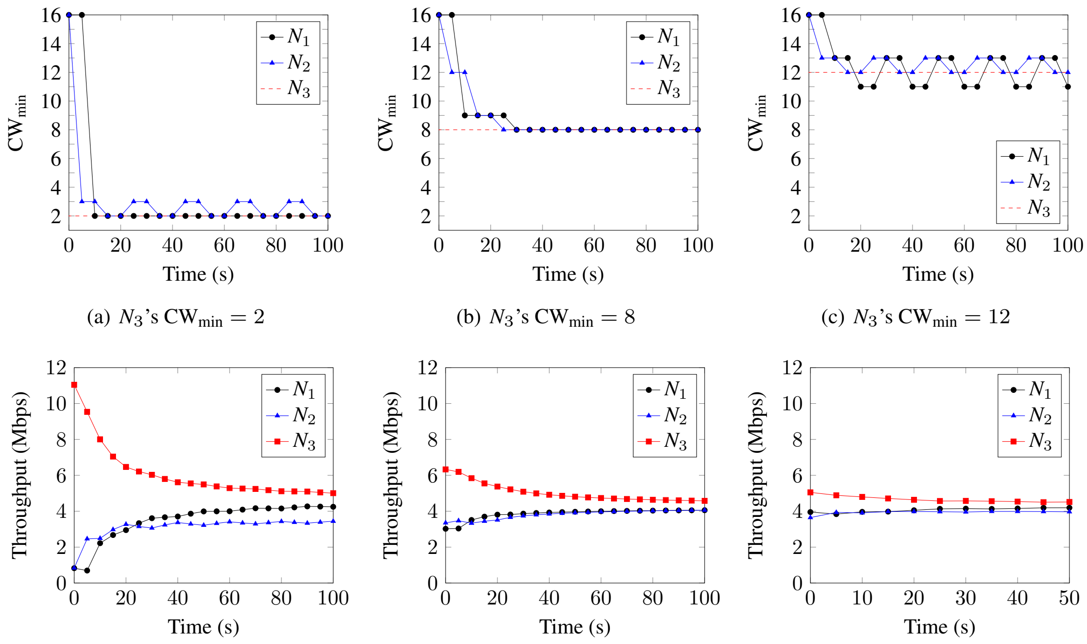
[16] L. N. Francesco Gringoli, "OpenFWWF: Open firmware for WiFi networks," http://netweb.ing.unibs.it/ openfwwf/, 2009.

[17] A. Sterz, M. Eichholz, R. Mogk, L. Baumgärtner, P. Graubner, M. Hollick, M. Mezini, and B. Freisleben, "ReactiFi: Reactive programming of Wi-Fi firmware on mobile devices." *Art Sci. Eng. Program.*, vol. 5, no. 2, p. 4, 2021.

[18] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *Proc. of the IEEE INFOCOM*, vol. 2, 2003, pp. 836–843.

[19] Y. Chetoui and N. Bouabdallah, "Adjustment mechanism for the IEEE 802.11 contention window: An efficient bandwidth sharing scheme," *Computer Communications*, vol. 30, no. 13, pp. 2686–2695, 2007.

[20] A. Ksentini, A. Nafaa, A. Gueroui, and M. Naimi, "Deterministic contention window algorithm for IEEE 802.11," in *Proc. of the IEEE PIMRC*, vol. 4, 2005, pp. 2712–2716.



(a) N_3's CW$_{\min} = 2$ (b) N_3's CW$_{\min} = 8$ (c) N_3's CW$_{\min} = 12$

(d) N_3's CW$_{\min} = 2$ (e) N_3's CW$_{\min} = 8$ (f) N_3's CW$_{\min} = 12$

FIGURE 15: CW$_{\min}$ value of all nodes vs. time, when N_3 is having CW$_{\min}$ values of (a) 2, (b) 8, and (c) 12, and their respective uplink throughput in (d), (e), and (f), respectively ($L = 3$).

[21] Q. Xia and M. Hamdi, "Contention window adjustment for IEEE 802.11 WLANs: a control-theoretic approach," in *Proc. of the IEEE ICC*, vol. 9, 2006, pp. 3923–3928.

[22] R. Pries, S. Menth, D. Staehle, M. Menth, and P. Tran-Gia, "Dynamic contention window adaptation (DCWA) in IEEE 802.11e wireless local area networks," in *Proc. of the IEEE ICCES*, 2008, pp. 92–97.

[23] L. Chen, S. H. Low, and J. C. Doyle, "Random access game and medium access control design," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 4, pp. 1303–1316, 2010.

[24] S.-W. Kang, J.-R. Cha, and J.-H. Kim, "A novel estimation-based backoff algorithm in the IEEE 802.11 based wireless network," in *Proc. of the IEEE CCNC*, 2010, pp. 1–5.

[25] P. Patras, A. Banchs, P. Serrano, and A. Azcorra, "A control-theoretic approach to distributed optimal configuration of 802.11 WLANs," *IEEE Transactions on Mobile Computing*, vol. 10, no. 6, pp. 897–910, 2011.

[26] S. Chun, D. Xianhua, L. Pingyuan, and Z. Han, "Adaptive access mechanism with optimal contention window based on node number estimation using multiple thresholds," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 2046–2055, 2012.

[27] C. Wang and W.-H. Kuo, "A utility-based resource allocation scheme for IEEE 802.11 WLANs via a machine-learning approach," *Wireless networks*, vol. 20, no. 7, pp. 1743–1758, 2014.

[28] L. Dai and X. Sun, "A unified analysis of IEEE 802.11 DCF networks: Stability, throughput, and delay," *IEEE Transactions on Mobile Computing*, vol. 12, no. 8, pp. 1558–1572, 2013.

[29] Y. Gao, X. Sun, and L. Dai, "Throughput optimization of heterogeneous IEEE 802.11 DCF networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 1, pp. 398–411, 2013.

[30] ——, "IEEE 802.11e edca networks: Modeling, differentiation and optimization," *IEEE Transactions on Wireless Communications*, vol. 13, no. 7, pp. 3863–3879, 2014.

[31] X. Sun and Y. Gao, "Distributed throughput optimization for heterogeneous IEEE 802.11 dcf networks," *Wireless Networks*, vol. 24, pp. 1205–1215, 2018.

[32] X. Sun and L. Dai, "Backoff design for IEEE 802.11 dcf networks: Fundamental tradeoff and design criterion," *IEEE/ACM Transactions on Networking*, vol. 23, no. 1, pp. 300–316, 2015.

[33] I. Syed and B.-h. Roh, "Adaptive backoff algorithm for contention window for dense IEEE 802.11 WLANs," *Mobile Information Systems*, 2016.

[34] M. Karaca, S. Bastani, and B. Landfeldt, "Modifying backoff freezing mechanism to optimize dense IEEE 802.11 networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 9470–9482, 2017.

[35] M. Hirzallah, W. Afifi, and M. Krunz, "Full-duplex-based rate/mode adaptation strategies for Wi-Fi/LTE-U coexistence: A POMDP approach," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 1, pp. 20–29, Jan 2017.

[36] M. Han, S. Khairy, L. X. Cai, Y. Cheng, and R. Zhang, "Reinforcement learning for efficient and fair coexistence between lte-laa and wi-fi," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8764–8776, 2020.

[37] A. Kumar, G. Verma, C. Rao, A. Swami, and S. Segarra, "Adaptive contention window design using deep q-learning," in *Proc. of IEEE ICASSP*, 2021, pp. 4950–4954.

[38] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[39] M. Hirzallah, M. Krunz, and Y. Xiao, "Harmonious cross-technology coexistence with heterogeneous traffic in unlicensed bands: Analysis and approximations," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 3, pp. 690–701, Sep. 2019.

[40] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[41] Y. Xiao, "Performance analysis of priority schemes for IEEE 802.11 and IEEE 802.11e wireless LANs," *IEEE Transactions on Wireless Communications*, vol. 4, no. 4, pp. 1506–1515, 2005.

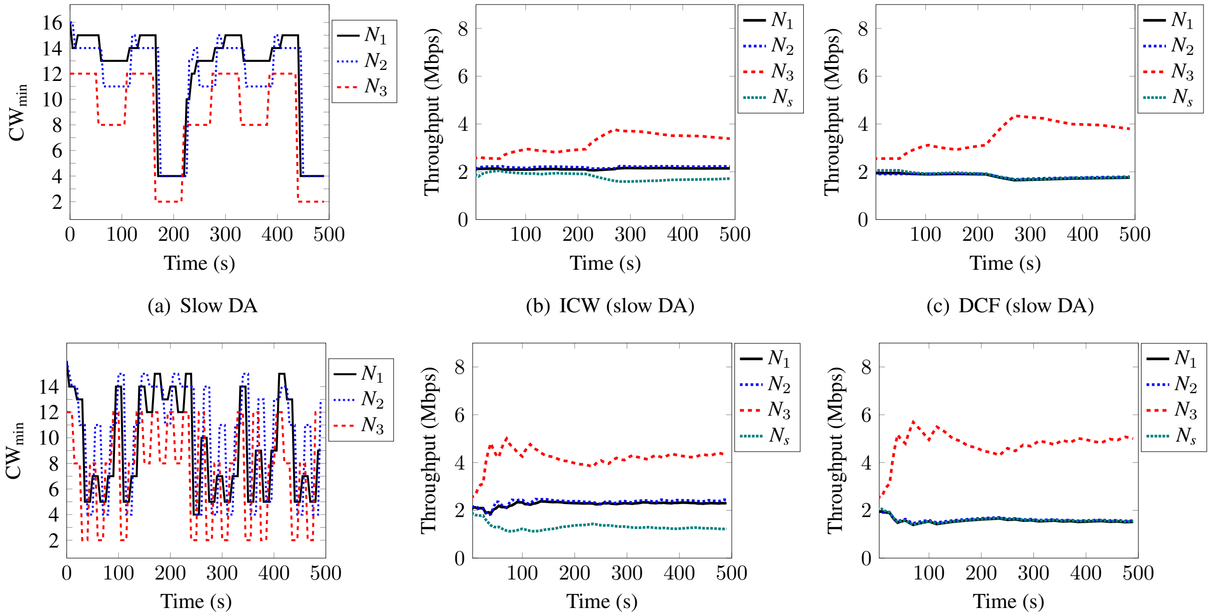
[42] "Csim20," [http://www.mesquite.com], accessed: 4/18/2022.

[43] A. H. Y. Abyaneh, M. Hirzallah, and M. Krunz, "Intelligent-CW: AI-based Framework for Controlling Contention Window in WLANs," in *in Proc. of the IEEE DySPAN*, 2019, pp. 1–10.

[44] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[45] "LabVIEW communications 802.11 application framework v2.1," https://www.ni.com/en-us/support/downloads/software-products/download.labview-communications-802-11-application-framework.html, accessed: 2021-12-08.

[46] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure



(a) Slow DA (b) ICW (slow DA) (c) DCF (slow DA)
(d) Fast DA (e) ICW (fast DA) (f) DCF (fast DA)

FIGURE 16: Per-node CW$_{\text{min}}$ selection and uplink throughput vs. time ($L = 6$).

of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984.

AMIR-HOSSEIN YAZDANI-ABYANEH is currently a Software Engineer at CelPlan Technologies, Inc. He received the MSc degree in electrical and computer engineering from the University of Arizona. He has worked with ED2 corporation on 5G mmWave systems and protocols as a system engineer intern. He also joined DOCOMO Innovations Inc. in the Summer of 2022 as a research engineer intern, where he worked on reinforcement learning solutions for joint communication and sensing in mobile mmWave systems. His research interests are mainly focused on machine learning and deep learning algorithms for spectrum sharing, wireless communication, and sensing.

MOHAMMED HIRZALLAH is currently a Senior Engineer at Qualcomm, Inc. He received the Ph.D. degree in electrical and computer engineering from the University of Arizona. In 2018, he joined CableLabs as a Wireless Research Intern, where he conducted research on NR-U and application of AI to wired/wireless communications. His research interests are mainly focused on wireless communications and protocols, machine learning for wireless communications, spectrum sharing, radar, remote sensing, and localization.

MARWAN KRUNZ is a Regents Professor at the University of Arizona. He holds the Kenneth VonBehren Endowed Professorship in ECE and is also a professor of computer science. He directs the Broadband Wireless Access and Applications Center (BWAC). Dr. Krunz's research is on resource management, network protocols, and security for wireless systems. He is an IEEE Fellow, an Arizona Engineering Faculty Fellow, and an IEEE Communications. He served as the Editor-in-Chief for the IEEE Transactions on Mobile Computing. Dr. Krunz served as chief scientist for two startup companies that focus on 5G and beyond systems and machine learning for wireless communications.