Intelligent-CW: AI-based Framework for Controlling Contention Window in WLANs

Amir Hossein Yazdani Abyaneh, Mohammed Hirzallah, and Marwan Krunz
Department of Electrical and Computer Engineering, University of Arizona, AZ, USA
Email: {yazdaniabyaneh, hirzallah, krunz}@email.arizona.edu

***Abstract*—The heterogeneity of technologies that operate over the unlicensed 5 GHz spectrum, such as LTE-Licensed-Assisted-Access (LAA), 5G New Radio Unlicensed (NR-U), and Wi-Fi, calls for more intelligent and efficient techniques to coordinate channel access beyond what current standards offer. Wi-Fi standards require nodes to adopt a fixed value for the minimum contention window (CW_{min}), which prohibits a node from reacting to aggressive nodes that set their CW_{min} to small values. To address this problem, we propose a framework called *Intelligent-CW (ICW)* that allows nodes to adapt their CW_{min} values based on observed transmissions, ensuring they receive their fair share of the channel airtime. The CW_{min} value at a node is set 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 over a large number of WLAN scenarios, including different misbehaving and aggressive scenarios. Under aggressive scenarios, our simulation results reveal that ICW provides nodes with higher throughput (153.9% gain) and 64% lower frame latency than standard techniques. In order to measure the fairness contribution of individual nodes, we introduce a new fairness metric. Based on this metric, ICW is shown to provide $10.89\times$ improvement in fairness in aggressive scenarios compared to standard techniques.**

I. Introduction

The number of Wi-Fi access points (APs) that operate over unlicensed bands is expected to increase by fourfolds between 2017 and 2022 [1]. Unlicensed bands will also host new wireless technologies, including LTE-LAA and 5G NR-U [2]. Under NR-U, mobile network operators (MNOs) can utilize the unlicensed spectrum to supplement their services over licensed bands. The plethora of wireless technologies that will have to coexist over the unlicensed spectrum create many challenges related to fairness, transmission reliability, etc. In this paper, we focus on one of these challenges namely, the greediness of some devices that try to acquire most of the channel airtime at the expense of others.

Several unlicensed-band technologies use a variant of Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with exponential backoff for channel access. These include Wi-Fi, LTE-LAA, DSRC, etc. According to the CSMA/CA procedure, a station that wishes to transmit will first sense the channel for a fixed duration called the initial Inter-Frame Space (IFS). If the channel remains idle during the initial IFS period, then the station starts transmission; otherwise, the station defers its transmission and waits for a random backoff period. The backoff period consists of k idle slots, where k is randomly chosen from $[0, \text{CW} - 1]$. Initially, CW is set to CW_{min} value and doubled after every collision until it reaches the maximum size of contention window, i.e., CW_{max}. In other words, a station that has consecutively collided for j times chooses its k randomly from $[0, \min(2^j \text{CW}_{\text{min}}, \text{CW}_{\text{max}}) - 1]$. The exponential increase in contention window helps nodes avoid collisions. Nodes with successful transmissions reset their contention window to the CW_{min} value. Although CSMA/CA does an excellent job in ensuring fairness among devices and reducing collisions, it is still vulnerable to aggressive nodes that do not abide by the standard-defined CW_{min} values, hence harming the performance of compliant nodes. Enforcing specific CW_{min} values is hard because it is difficult to detect and identify non-compliant nodes. Non-aggressive nodes need a mechanism to detect aggressive behavior and adapt their CW_{min} values accordingly, ensuring they receive their fair share of the airtime. They also need to roll back to their standard CW_{min} setting once aggressive nodes retreat to a compliant behavior.

To cast more light on this issue, we conduct a simple experiment in which three Wi-Fi nodes share the same unlicensed channel. Two nodes, D_2 and D_3, are aggressive. They set their initial CW_{min} value to 4. We consider two cases for the setting of D_1's CW_{min} value. In Case 1, D_1 chooses the default $\text{CW}_{\text{min}} = 16$ value, while in Case 2, D_1 randomly chooses CW_{min} between 2 and 16. We plot the per-node throughput in Figure 1. For Case 1, nodes D_2's and D_3's traffic accounts for 91.26% of the network throughput, while D_1's traffic accounts for only 8.73% of the total throughput. In contrast, for Case 2 the random assignment of CW_{min} value alleviates the unfairness issue and improves the throughput of D_1 by almost $2.5\times$ ($\approx 22.78\%$ of the total throughput). Rather than randomizing the selection of CW_{min}, in this paper we exploit artificial intelligence techniques to achieve even higher throughput at D_1, close to its fair share of 33% of the total throughput. Achieving this fair allocation is challenging because the CW_{min} values of D_2 and D_3 are not known at D_1. In fact, D_1 is not aware of the greedy behavior of D_2 and D_3.

To mitigate the impact of the aggressive setting of the CW_{min} value, we introduce a framework called *Intelligent-CW (ICW)*, in which nodes adapt their CW_{min} values to coexist fairly with their neighbors. Nodes become aware of the aggressive activities of other nodes by observing the unlicensed channel and obtaining some statistics about their neighboring nodes, including the number of active neighbors and the duration of time each neighbor occupies the channel. To

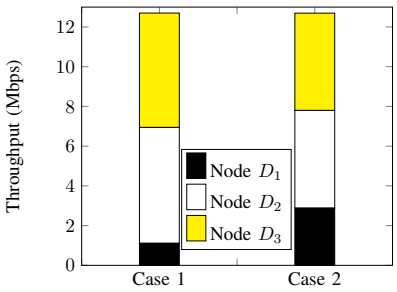


Fig. 1: Per-node throughput for a network of 3 nodes, where D_2 and D_3 are aggressive and set their CW_{min} values to 4 (Case 1: D_1 sets its CW_{min} to 16, i.e., standard value; Case 2: D_1 randomly selects its CW_{min} between 2 and 16).

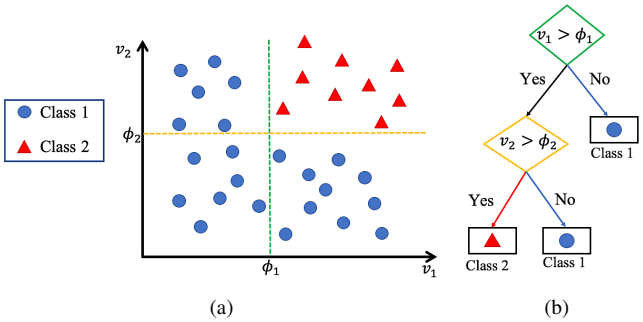


Fig. 2: (a) Example of a feature space of two classes (circles and triangles), and (b) the decision tree that is used to classify objects into these classes.

accelerate the CW_{min} adaptation process, we equip each node with a machine learning (ML) module, designed based on the random forest, a well-known technique with several attractive properties in terms of short training time and low computational complexity. Compared to other existing approaches, our approach is more efficient, as it avoids computationally intensive Markovian analysis approaches and the solving of nonlinear equations [3]. It is also robust to dynamical changes in the data that is fed to it, and is resistant to over-fitting. We design a rule to help nodes detect the aggressive behavior of their neighbors and select the best CW_{min} values that maximize fairness.

ICW is practical for two reasons. First, it is trained to generalize to large-size WLANs. Secondly, it is designed to work in a distributed fashion, with light computations and no communication overhead. Our simulation results reveal that on the presence of aggressive nodes, ICW increases the throughput by 153.9%, decreases the average per-frame latency by 64%, and most importantly, based on the new fairness metric that we define to measure the fairness contribution of individual nodes, ICW provides $10.89\times$ improvement in fairness in aggressive scenarios compared to standard techniques. Also, ICW rolls back to standard setting when aggressive nodes abandon their greedy behavior.

The paper is organized as follows. Section II provides some background about decision trees and random forest. In Section III, we introduce the main components of the ICW framework. The ML module is presented in Section IV, followed by evaluation results in Section V. Finally, we survey related works and conclude the paper in Sections VI and VII, respectively.

II. Background: Decision Tree and Random Forest

A classification problem consists of objects that need to be classified into one of several candidate classes. Each object has a set of N_f features, described by a vector $v = \langle v_1, v_2, ..., v_{N_f} \rangle$ in the feature space $\mathbb{R}^{N_f}$, where v_j is a variable that corresponds to the value of the jth feature, $j = 1, ..., N_f$. In Figure 2(a), we present an example of object representation in $\mathbb{R}^2$, i.e., $N_f = 2$, and could belong to either of two classes, i.e., circles or triangles.

A decision tree is an ML structure that is used for classification. Such a tree consists of a root node, internal nodes, terminal nodes (leaves), and branches that connect these nodes, as shown in Figure 2(b). Leaves correspond to the classes that objects could belong to. To classify an object, we start from the root, walk through the tree from top to bottom until we reach the correct leaf that represents the class of the object. The root and the internal nodes are each associated with features and cut-points that specify the path that an object needs to traverse across the tree. The mapping between root/internal nodes and their corresponding features and cut-points is part of the training process and the construction of the decision tree. One of the algorithms used for constructing decision trees is the *recursive binary splitting (RBS)* algorithm [4]. This algorithm finds the best feature, say v_j, and cut-point value, say ϕ_j, for each split, such that it has the minimum loss, where the loss function can be the *Gini index*. The loss function is used to assess how good a split is by measuring the variance of an object class in the branches constructed by that split [4], e.g., a low Gini index over objects that pass a specific branch means that these objects have low dispersion and they are more likely to belong to the same class. Figure 2(b) shows an example of a decision tree for classifying the objects in Figure 2(a). Construction of this decision tree requires learning the proper mapping of v_j and cut-point ϕ_j at each split.

A random forest improves the classification accuracy of a decision tree by using multiple differently constructed decision trees that operate in parallel. A random forest with d decision trees classifies an object by feeding it to each tree and later obtains d classification results. The mode of these d results is then used to produce the final classification output. Figure 3 represents an example of a random forest that classifies the objects in Figure 2(a) into two classes.

III. Intelligent-CW Framework

Our goal is to adapt CW_{min} to achieve fair sharing of unlicensed channels in the presence of aggressive nodes that manipulate their CW_{min} values. To reach this goal, we define

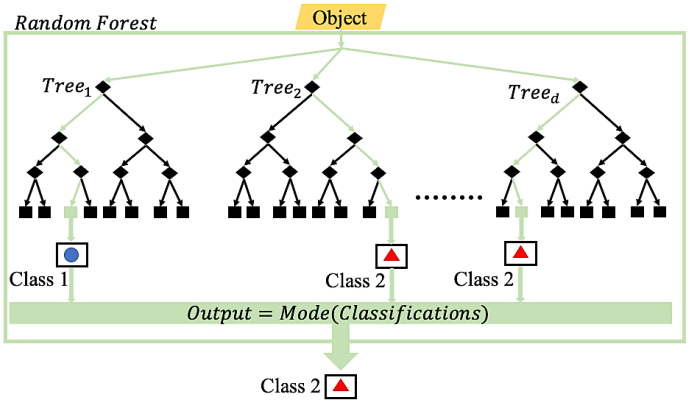


Fig. 3: Example of a random forest with d trees.

a fairness criterion that helps nodes detect the aggressive behavior of other nodes and trigger the adaptation of CW_{min}. This requires nodes to monitor the wireless channel and collect observations and statistics about their neighboring nodes. We now present our system model, the required observations, and the fairness criterion used for CW_{min} adaptation.

A. System Model

We consider a network model in which an intelligent node, say node I, shares an unlicensed channel with N other nodes in its vicinity, as shown in Figure 4. Let $\mathcal{D} = \{D_1, D_2, ..., D_N\}$ be the set of nodes in the neighborhood of node I. Nodes rely on CSMA/CA with exponential backoff to access the channel. Some of them fix their CW_{min} value to a standard value (e.g., 16), while others act aggressively and choose low CW_{min} values, allowing them to receive an unfair share of the channel airtime. Let ω be the CW_{min} value of node I, and let w_j be the CW_{min} value of node $D_j \in D$, $j = 1, ..., N$. Node I adapts ω by implementing ICW, using observations made over a given time period T. It is important to note that node I is not aware of the CW_{min} values used by its neighbors, but it can keep track of the number of active neighbors as well as the durations they occupy the channel. Although our system model is based on one intelligent node I, it can be generalized to include arbitrary numbers of intelligent, well-behaving, and misbehaving nodes. We choose to include one intelligent node for brevity and ease of illustration.

B. Observations

By monitoring the channel for a period T, node I collects the following information:

- B: channel busy time as observed by node I.
- F: total time during which node I occupies the channel.
- n: number of frames sent by node I during the T period.

We denote the CW_{min} value that node I uses during the T period by ω, and the CW_{min} value recommended by its ML module right after the T period by $\hat{\omega}$. In Figure 5, we provide an example that shows the observations made by node I when it shares the channel with two other nodes, i.e., $N = 2$.

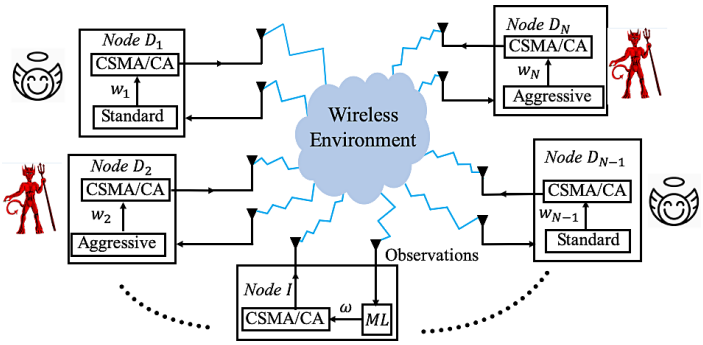


Fig. 4: System model of a WLAN that consists of an intelligent node I that shares an unlicensed channel with N other nodes (nodes D_2 and D_N are aggressive nodes).

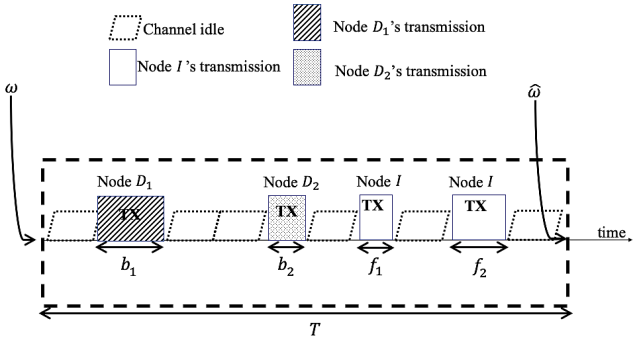


Fig. 5: Example of monitoring the channel over a period of T seconds.

Node I determines the total busy duration as $B = b_1 + b_2$, where b_j, $j = 1, 2$, is the jth duration during which node I finds the channel to be busy by some other node. Node I occupies the channel for a period $F = f_1 + f_2$, where f_j is the jth transmission duration for node I. By the end of T, node I determines that $N = 2$ and $n = 2$, and it passes the set of observations and configurations F, B, T, n, N, and ω as a feature vector $v = \langle F, B, T, n, N, \omega \rangle$ to its ML module to determine $\hat{\omega}$ that will be used over the next T seconds. We select these six specific features to enhance the classification accuracy of the ML module, and because these features include statistics that correlate with the CW_{min} values of other neighboring nodes. It is important to note that T needs to be set large enough to ensure that node I collects sufficient statistics.

C. Fairness Criterion

Node I relies on a fairness rule to detect aggressive behavior and trigger the adaptation of its CW_{min} value. Before we introduce our fairness rule, we define the following metrics.

Definition 1. *The channel-Utilization-by-Neighbors (CUN) ρ is defined as the ratio of the total channel busy time as seen by node I and the observation duration: $\rho = \frac{B}{T}$.*

Definition 2. *The optimal Channel Utilization Factor (OCUF) is defined as $\Omega = \frac{N}{N+1}$.*

Definition 3. *The state of the wireless channel is the vector* $S = \langle w_1, w_2, ..., w_N \rangle$ *of the* CW_{min} *values adopted by the neighbors of node* I.

Our fairness rule from the perspective of node I states that $\hat{\omega}$, the CW_{min} value produced by the ML module, is fair and efficient if the neighbors of node I end up occupying $\frac{N}{N+1}$ fraction of T. In other words, $\hat{\omega}$ should lead to a future ρ value that satisfies $\rho = \Omega$. If node I is aware of the state S, then it can compute an optimal CW_{min} value. In practice, it is hard for node I to know the CW_{min} values of its neighbors, since such values are not broadcasted. Instead, node I can rely on overheard information, i.e., B, F, n, and N, to recommend a fair $\hat{\omega}$ value. The ML module exploits this information to recommend an optimal $\hat{\omega}$ that is aligned with our fairness criterion. In the next section, we explain how the ML module can be trained to achieve this objective.

IV. ML Module Design

The ML module adapts the CW_{min} value of node I based on the vector of features that it receives as inputs, i.e., $v = \langle F, B, T, n, N, \omega \rangle$, and classifies them to a proper $\hat{\omega}$ value to be used in the next T period. We conduct extensive simulations using a customized simulator and collect a large number of input samples along with their best CW_{min} values, obtained through exhaustive search. We use these values as labels to train the ML module in a supervised manner. We denote the optimal CW_{min} value that we expect the ML module to produce by ω^*. In the test phase, we expect the ML module to produce $\hat{\omega} = \omega^*$ values for newly generated inputs. Next, we explain how we derive the ω^* labels and how we build a training dataset for different channel states.

A. Dataset Construction

To train the ML module in a supervised manner, we build a dataset that consists of feature vectors and their associated labels. The feature vector includes the set of observations that node I monitors during T and the configurations that it uses during the monitoring period. To obtain the label for a particular state, say state S, we gradually increase the CW_{min} value that node I uses, i.e., ω, in the interval $\bar{\delta} = \{\delta_1, ..., \delta_2\}$, where δ_1 and δ_2 are two arbitrary positive integers, and monitor the set of observations associated with each ω. We compute $\rho = \frac{B}{T}$, then we search for the ω value that best approaches $\rho = \Omega$. The optimum label for node I can then be found as:

$$\omega^* = \underset{\omega \in \bar{\delta}}{\operatorname{argmin}} \, |\rho - \Omega| \, . \tag{1}$$

Figure 6 depicts a sample of our results when node I shares a channel with two other nodes whose CW_{min} values are set to $w_1 = 9$ and $w_2 = 4$, i.e., $S = \langle 9, 4 \rangle$. In this example, the OCUF is 0.66 (shown by the dashed red line in Figure 6(a)). We vary ω from 2 to 16 and monitor the CUN value. As can be observed in Figure 6(a), there are two candidate ω values that are close to Ω: $\omega = 5$ and $\omega = 6$. We choose the ω value for which ρ has the least difference from Ω to be the label

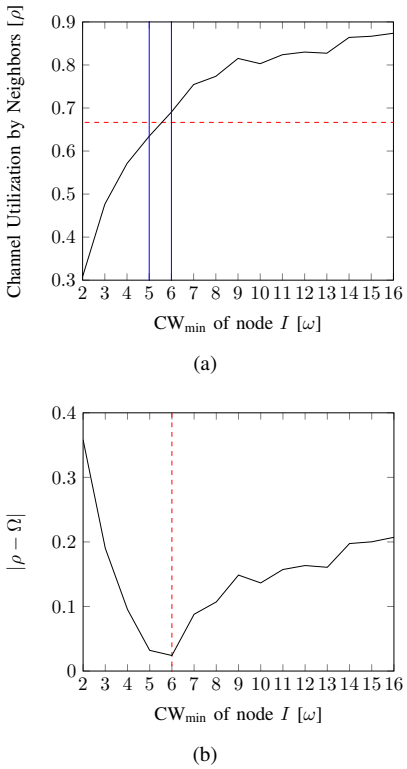


Fig. 6: (a) Channel utilization by neighbors vs. CW_{min} of node I, i.e., ω, when $N = 2$ and $S = \langle 9, 4 \rangle$, and (b) the $|\rho - \Omega|$ vs. ω ($\Omega = 0.66$).

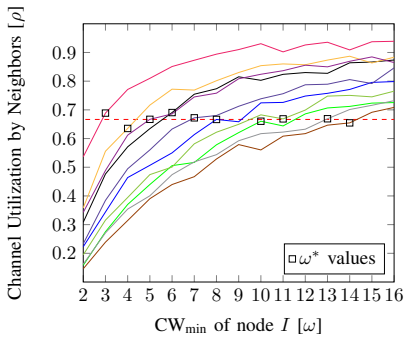


Fig. 7: Channel utilization by neighbors vs. CW_{min} of node I with $N = 2$ for 10 different states (ω^* values are represented by squares).

of this specific state, as shown in Figure 6(b). Accordingly, we set $\omega^* = 6$ to be the label for the state $S = \langle 9, 4 \rangle$. Note that the optimal label ω^* depends on the state. In Figure 7, we plot the CUN versus ω for 10 different states, and we observe that the ω^* value is different for different states.

Figure 8(a) depicts a flowchart that shows the construction of the training dataset for node I. The table in Figure 8(b) includes the training dataset of one state with each row corresponding to one training sample. We set our

customized WLAN simulator to a specific state S, initialize ω and ω^* to δ_1, $\overset{*}{\rho}$ to zero, where $\overset{*}{\rho}$ is the CUN value corresponding to the most recent ω^*, and run the simulator for T seconds. During this duration, we keep track of node I's observations and fill the corresponding row of the table in Figure 8(b). At this moment, we still do not know the correct entry of the last column, i.e., ω^*. We have to wait until we exhaustively run the simulator for other ω values. After each run, we check if $|\rho - \Omega|$ is less than or equal to $|\overset{*}{\rho} - \Omega|$. If the condition is satisfied, then ω^* and $\overset{*}{\rho}$ are updated to the current ω and ρ values, respectively. We increase ω by 1 and repeat the process until ω reaches δ_2. At this point, we fill the last column of the dataset table in Figure 8(b) with the most recent value of ω^*. We repeat the same procedure for a new channel state and construct a large set of training tables. These dataset tables are used to train the ML module in a supervised manner. The set of best CW_{min} values, i.e., ω^*, correspond to the labels of classes.

B. Decision Trees & Random Forests

We first explain the construction of one decision tree, and then we explain the construction of the random forest.

1) Construction of a Single Decision Tree: Let $\mathcal{R}$ be the set of training samples obtained as in Section IV-A. A decision tree of depth J divides the feature space into $L = 2^J$ distinct and non-overlapping regions, $R_1, R_2, ..., R_L$, where each region corresponds to a particular class. Samples of one class could be part of multiple regions. These regions will end up being 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. To fully understand RBS, first we introduce the *Gini index*. Consider an arbitrary set r_m of samples that potentially belong to different classes, i.e., $r_m \subset \mathcal{R}$. The Gini index $G(r_m)$ of the set r_m can be expressed as:

$$G(r_m) = \sum_{k=1}^{K} p_k(r_m)(1 - p_k(r_m)) \tag{2}$$

where K is the number of classes to which samples in the set r_m belong, and $p_k(r_m)$ is the proportion of training samples that belong to class k and are also in the r_m set. Gini index measures the dispersion of the samples in the set r_m. A low value of G_m indicates that the samples in r_m are more likely to belong to the same class.

To build a decision tree, we start with the root of the decision tree and look for a feature v_j and a cut-point value ϕ_j that split $\mathcal{R}$ 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 $r_{12} = \{v : v_i > \phi_i, v \in r_1\}$ such that 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 J.

2) Construction of Random Forest: A random forest of depth d has d uncorrelated decision trees inside it. Each tree is constructed as explained before, but for each split we only consider $\lfloor\sqrt{N_f}\rfloor$ random features, i.e., $\lfloor\sqrt{N_f}\rfloor = \lfloor\sqrt{6}\rfloor = 2$ in our case. After building the random forest, the sample that needs to be classified is fed to each one of these d trees, while each tree executing its own classification. The random forest takes the mode of these d classifications as its final output.

C. Flexible Testing Accuracy

Depending on the state of the channel, we may end up with multiple CUN ρ values that are close to the OCUF. For example, in Figure 6(a), we have two candidate ω values whose ρ value is close to Ω. We define the *accepted classification range* to relax the classification accuracy of the ML module so as to include multiple $\overset{*}{w}$ values as potential labels.

Definition 4. *Accepted Classification Range (ζ)*
We define ζ as an upper bound on the absolute difference between CUN and OCUF for which $\hat{\omega}$ is accepted to be an optimal label. In other words, $\hat{\omega}$ is accepted as an optimal label if it satisfies the following:

$$|\rho - \Omega| \leq \zeta. \tag{3}$$

Previously, we saw in Figure 6(a) that the only optimum label was $\omega^* = 6$; however, by taking $\zeta = 0.05$, we will have two acceptable ω^* values, i.e., $\omega^* = 6$ and $\omega^* = 5$ (see red circles in Figure 9).

D. Fairness Measures

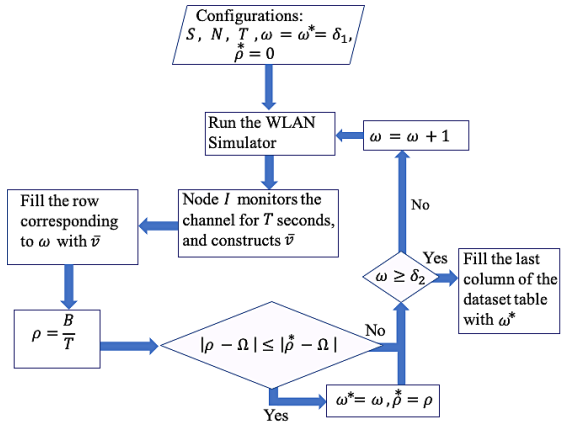
We use two fairness measures to assess how ICW improves fairness from both network and node I perspectives. Jain's index is a fairness metric that can be used for measuring the fair allocation of throughput from a network level [5]. Consider a network of m nodes, where the ith node has a throughput of Γ_i. Then Jain's index (J) is expressed as follows:

$$J = \frac{(\sum_{i=1}^{m} \Gamma_i)^2}{m \sum_{i=1}^{m} \Gamma_i^2}. \tag{4}$$

When $J = 1$, all m nodes in the network have equal throughput. Hence, the channel has been fairly allocated between them.
Node I should be fair to other nodes in its vicinity. To study whether or not node I is being fair to others, we define a new metric called *One-Way Fairness (OWF)* denoted by β:

$$\beta = \left| \frac{B}{F} - N \right|. \tag{5}$$

When $\beta = 0$, then node I is fair to other nodes in its vicinity.



(a)

Dataset

Feature Vector ($\vec{v}$)						Label
F	B	T	n	N	ω	ω^*
....	....	....	....	....	δ_1	$\delta_1 + 3$
....	....	....	....	....	$\delta_1 + 1$	$\delta_1 + 3$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
....	....	....	....	....	$\delta_2 - 1$	$\delta_1 + 3$
....	....	....	....	....	δ_2	$\delta_1 + 3$

(b)

Fig. 8: (a) Flowchart for constructing the training dataset in the ICW framework, and (b) example of the dataset table for one channel state.

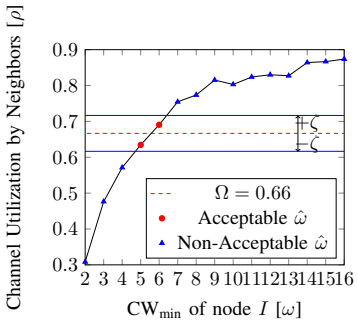


Fig. 9: Channel utilization by neighbors (ρ) vs. ω (Accepted CW_{min} ω^* are shown as red circles and non-accepted values are shown as blue triangles for $\zeta = 0.05$).

V. Evaluation

In this section, we first evaluate the classification accuracy of the ML module, followed by the performance of the ICW framework, including throughput, latency, and fairness measures (Jain's fairness index and OWF). Our evaluation is based on a C++-based discrete-event-based simulator called CSIM [6]. CSIM includes functions and classes for generating and synchronizing process-oriented events. We implement the DCF as detailed in the IEEE 802.11 ac standard, including all timing requirements. An indoor scenario is considered, where a number of Wi-Fi devices are uniformly distributed in a square area of length 40 meters.

A. Classification Accuracy of ML Module

We consider four WLAN scenarios. For the first three scenarios, we fix the number of node I's neighbors to $N = 2$, $N = 5$, and $N = 8$, respectively. In the fourth scenario, we randomly select N from $\{2, 5, 8\}$. We denote this last case as the "combined" scenario. For all scenarios, we have eight classes of ω^*, ranging from 2 to 9. We consider 10000 samples to represent each class in the first three scenarios. The ML module is trained using 53600 samples and tested over 26400 samples for these scenarios. In the fourth scenario, each class is represented by 30000 samples. Training for this scenario is done using 160800 samples and testing is done using 79200 samples. We investigate the impacts of d, the random forest depth, and J, the depth of its constituent decision trees, on the accuracy of classifying the optimal CW_{min} value. In Figure 10, we plot the classification accuracy vs. J for all four scenarios. The depth of the random forest is fixed at $d = 200$, and the accepted classification range is set to $\zeta = 0.04$. We notice that the classification accuracy reaches a fixed value after $J = 20$ for all four scenarios. Accordingly, we set $J = 20$ in our subsequent experiments. In Figure 11, we plot the classification accuracy of the random forest versus d when $J = 20$ and $\zeta = 0.04$. Notice that the classification accuracy saturates after $d = 20$. Therefore, we select 20 as the number of trees in the ML module. In Figure 12, we plot the classification accuracy for the four scenarios versus ζ. At $\zeta = 0.04$, we can achieve classification accuracy of 92.24%, 93.65%, 97.07%, and 93.37%, for the four scenarios, respectively. Under a fixed ζ, we observe that the classification accuracy improves with N. This is due to the fact that a larger N results in more candidate ω^* values within the fixed ζ range. It can also be observed that the classification accuracy under the combined scenario mediates the classification accuracy of its constituent three cases.

B. WLAN Performance

We compare the performance under the ICW framework with two other mechanisms for controlling the CW_{min} value at node I: The standard IEEE 802.11 DCF and a randomized (Rnd) CW_{min} mechanism. In the DCF mechanism, CW_{min} is fixed to 16. In the Rnd mechanism, CW_{min} is randomly selected from $\{2, 3, ..., 16\}$. We run the simulator 10000 times and report the mean of these runs. We investigate the impact

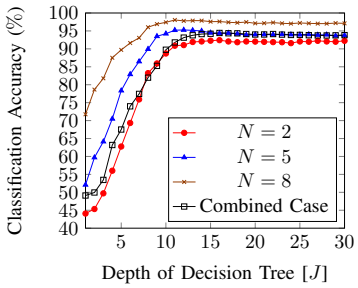


Fig. 10: Classification accuracy of random forest vs. the depth of its decision trees (J).

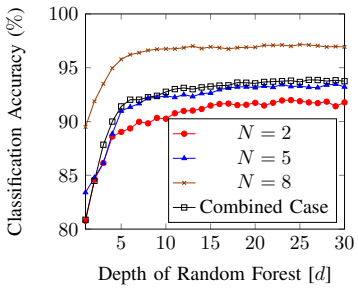


Fig. 11: Classification accuracy of the random forest vs. its depth (d).

of CW_{min} of node I on the throughput, latency, and fairness of an arbitrary WLAN setup with two other nodes (labeled as D_1 and D_2) and state $S = \langle 4, 8 \rangle$. In Figure 13(a), we plot the network throughput versus ω. In Figure 13(b), we plot the per-frame latency versus ω. We notice from Figures 13(a) and 13(b) that varying ω changes the throughput and latency over the network for all nodes. To derive the optimal ω that maximizes fairness for the given state S, we plot Jain's index versus ω in Figure 13(c). Node I achieves the second-highest Jain's index of 0.895 when $\omega = 5$. To see which ω results in node I being the fairest to its two neighbors, we plot OWF versus ω in Figure 13(d). We can observe that $\omega = 5$ achieves the least value of OWF. By setting CW_{min} to 5 node I treats the two other nodes fairly and also utilizes the channel efficiently.

To compare ICW's performance with DCF and Rnd mechanisms, we consider 6 different states S, namely, $\langle 4, 4 \rangle$, $\langle 8, 8 \rangle$, $\langle 16, 16 \rangle$, $\langle 4, 8 \rangle$, $\langle 4, 16 \rangle$, and $\langle 8, 16 \rangle$.

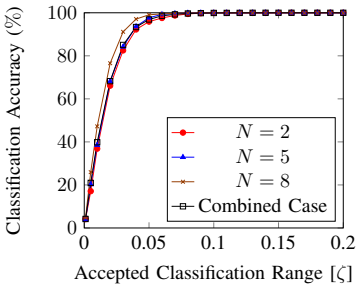


Fig. 12: Classification accuracy of random forest vs. ζ.

1) Throughput: In Figure 14, we plot the per-node throughput resulting from the three different CW_{min} selection mechanisms. Under the DCF mechanism, we observe that node I behaves conservatively, leading to low throughput. On the other hand, Rnd mechanism makes node I too aggressive in some cases, e.g., $S = \langle 16, 16 \rangle$. Node I is the fairest to its neighbors when it implements the ICW mechanism. An interesting observation is the ICW performance under channel state $S = \langle 16, 16 \rangle$. In this case, the ICW mechanism forces node I to behave as in the standard DCF, meaning that ICW triggers node I to be standard-compliant when other nodes are also compliant. We also computed the *average* throughput of node I under ICW and for various values for S and compared it with the DCF throughput. Overall, we found that ICW improves the throughput by 153.9%.

2) Latency: In Figure 15, we plot the average per-frame latency for each node vs. the state S under the three selection mechanisms. Under ICW, node I achieves a lower latency than DCF for all values of S, while it has the same performance as DCF in the standard state, i.e., $S = \langle 16, 16 \rangle$. To calculate the improvement in latency, we compute the *average* per-frame latency of node I under ICW and compare it with DCF scheme. We found that ICW achieves 64% of the DCF latency.

3) Fairness: In Figure 16, we plot Jain's index for the three mechanisms vs. S. We observe that ICW has the best fairness index for all states, while it has the same performance as the DCF in the standard setting, i.e., $S = \langle 16, 16 \rangle$. In Figure 17, we plot OWF vs. S. It can be observed that ICW has the lowest OWF value for all states and this proves that node I, under ICW, is the fairest to its neighbors. To calculate the improvement in fairness, we calculated the *average* Jain's index improvement of ICW over all states except the standard case, i.e., $S = \langle 16, 16 \rangle$. ICW provides up to 19.34% improvement in Jain's index when compared to DCF. We also found that ICW could reduce the OWF down to 90.81% when compared to DCF, which translates to about $\times 10$ improvement in OWF fairness.

VI. Related Works

In [7] the authors discussed how exponential backoff harms the performance. To tackle this issue and improve both fairness and network throughput, Chetoui et al. [8] provided each node in the WLAN with a set of backoff windows. Nodes select one of the windows based on their downlink bitrates rather than complying with the DCF protocol. To achieve high fairness among nodes and stability in throughput and delay, the authors in [9] proposed a deterministic algorithm for adjusting the CW bounds (i.e., upper and lower bounds of the interval from which the counter is randomly selected) to keep the collision rate low and reduce the number of retransmissions. Instead of only doubling the upper bound of CW, their algorithm increases both backoff range bounds. After each successful transmission, the CW bounds are decreased based on the current and previous network load. The authors

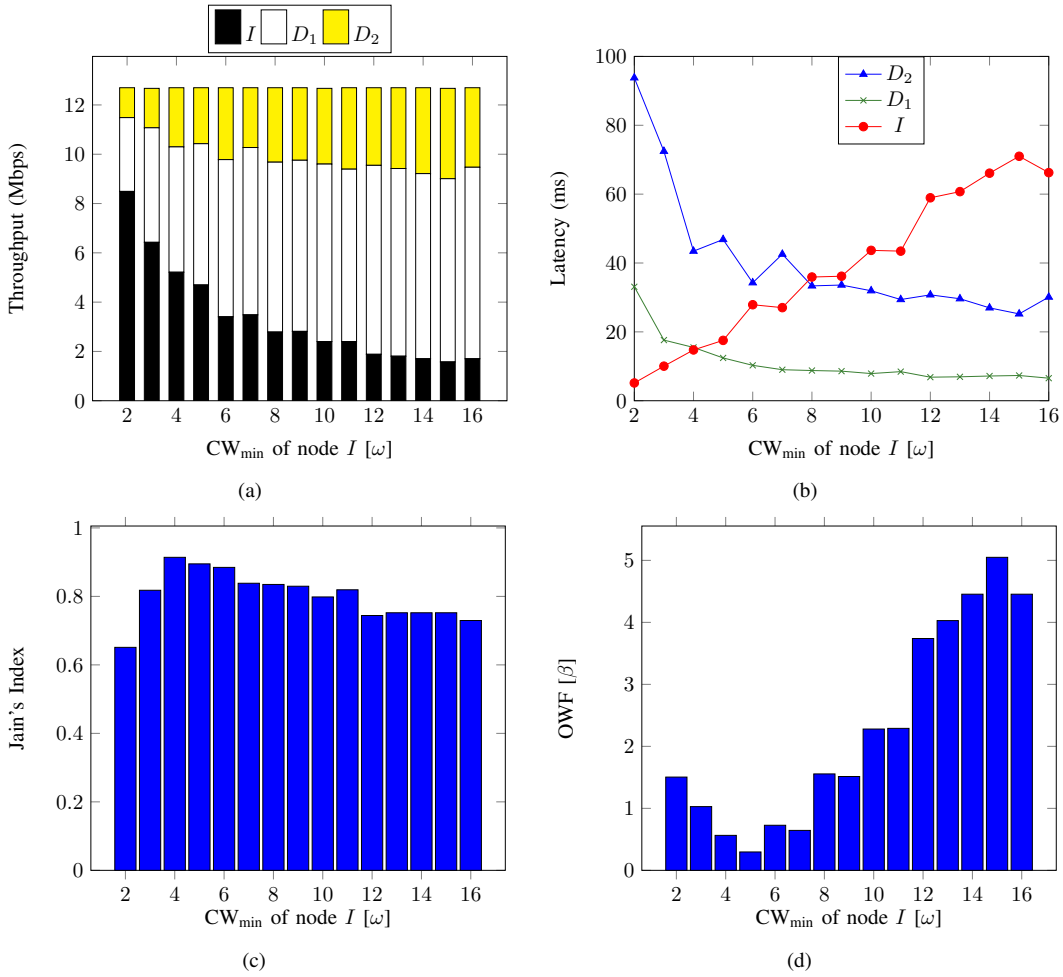


Fig. 13: (a) Downlink throughput, (b) per-frame latency, (c) Jain's index, and (d) OWF vs. the CW_{min} of node I.

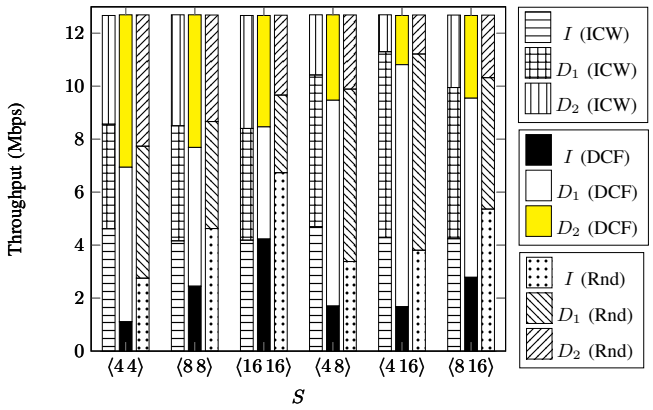


Fig. 14: Downlink throughput under the three different CW_{min} selection mechanisms for different states S.

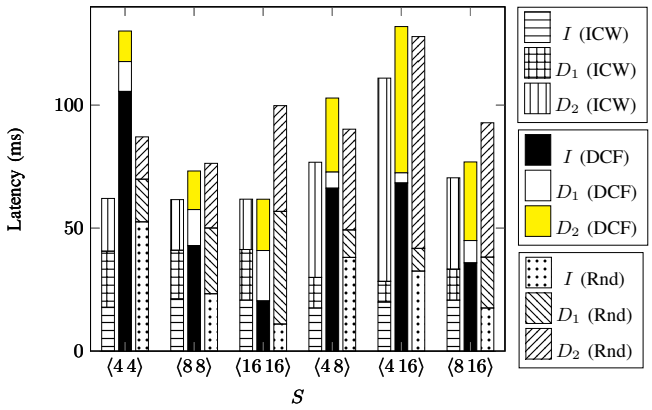


Fig. 15: Latency for the three different mechanisms for different states S.

in [10] used a feedback-control system with a Proportional-Derivative (PD) controller for adjusting the CW value at each node so as to maximize channel utilization. Pries et al. [11] introduced a measurement-based scheme to adapt CW_{min} and CW_{max} so as to minimize the contention delay and maximize throughput. The authors in [12] proposed a game-theoretic model for contention control, they applied a distributed update algorithm to achieve the equilibrium of the model, whereby

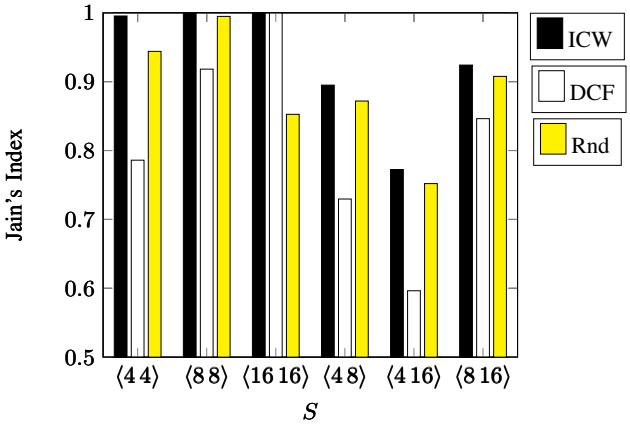


Fig. 16: Jain's index for the three different mechanisms vs. S.

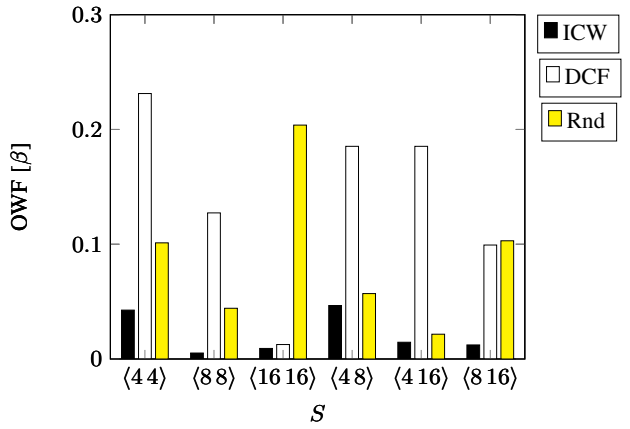


Fig. 17: OWF (β) for the three different mechanisms vs. S.

the network achieves high throughput, low collision, and short-term fairness. Kang et al. [13] presented an estimation-based algorithm to produce a new CW value to enhance both throughput and delay after each collision or a successful transmission by estimating the number of active nodes and network load variation. To enhance throughput and reduce delay, Patras et al. [14] developed a *proportional-integral* (PI) controller for adapting CW_{min} based on the number of successful transmissions and the number of retransmissions over a period of time. Chun et al. [15] derived an optimal CW setting to increase short-term fairness and improve the total network throughput. Their approach is based on the number of active nodes in WLAN. The authors developed an algorithm to predict the future number of nodes based on current network status and used this information to configure the CW value for all the nodes in the network. Nodes with different applications in the WLAN have different expectations from the network, e.g., real-time services, such as video conferencing and voice over IP require certain throughput and have a delay constraint so that they meet their Quality of Service (QoS). To meet such QoS, the authors in [16] proposed an adaptation mechanism for infrastructure 802.11 WLANs. They used deep neural networks to set the CW_{min} value and the Arbitrary Inter-Frame Space (AIFS) value for all nodes based on changes in the channel conditions and the number of active nodes. Unlike our work, the authors in [16] did not consider the existence of aggressive nodes in the WLAN. Also, their adaptation relies on communication between each AP and its corresponding stations, whereas in ICW, nodes configure their CW_{min} value in a distributed fashion. In [17], the authors proposed an adaptive backoff algorithm to maximize total network throughput and achieve high fairness by producing a new CW value based on the estimated number of active nodes and channel state probabilities. In [18], the authors proposed a modified DCF backoff process, in which the backoff counter is decremented based on channel idleness as well as the number of active nodes on the same channel. To reduce collisions between WLANs and duty-cycled LTE-unlicensed (LTE-U) transmissions, the authors in [19] exploited AI-based techniques whereby WLAN devices adapt their transmission rates and the direction of communication depending on LTE-U interference. Their work has also been extended to increase the sum-throughput for asymmetric full-duplex-enabled WLANs, serving QoS-enabled traffic [20].

VII. Conclusions

In this paper, we discussed how the aggressive setting of CW_{min} value can impair the performance of nodes that adopt a standard-based CW_{min} setting. We introduced the ICW framework, which adapts the CW_{min} value for standard-compliant nodes when the network contains aggressive nodes. ICW has the capability of rolling back to standard settings when aggressive nodes abandon their aggressive behaviors. It achieves high throughput efficiency while maintaining fair allocation of the unlicensed channels with other neighboring nodes. One of the most important features of ICW is that its learning structures can be easily trained and optimized to ensure high spectrum efficiency while being fair to other coexisting systems. ICW can be used in heterogeneous WLANs to provide fairness between different technologies. Our adaptation module predicts the correct CW_{min} value with 93.37% accuracy, achieves up to 10.89 times higher fairness, and increases throughput by 153.9% compared to the DCF mechanism in WLANs containing aggressive nodes.

References

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

[2] 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.

[3] 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.

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

[5] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984.

[6] "Csim20," [http://www.mesquite.com], accessed: 2019-08-30.

[7] 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.

[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] 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.
[15] 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.
[16] 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.
[17] I. Syed and B.-h. Roh, "Adaptive backoff algorithm for contention window for dense IEEE 802.11 WLANs," *Mobile Information Systems*, 2016.
[18] 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.
[19] 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.
[20] M. Hirzallah, W. Afifi, and M. Krunz, "Provisioning QoS in Wi-Fi systems with asymmetric full-duplex communications," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 942–953, Dec 2018.