Dynamically Tuning IEEE 802.11's Contention Window Using Machine Learning

Yalda Edalat
Department of Computer Science and Engineering,
University of California Santa Cruz, USA
yalda@soe.ucsc.edu

Katia Obraczka
Department of Computer Science and Engineering,
University of California Santa Cruz, USA
katia@soe.ucsc.edu

ABSTRACT

The IEEE 802.11's binary exponential backoff (BEB) algorithm plays a critical role in the throughput performance and fair channel allocation of IEEE 802.11 networks. In particular, one of BEB algorithm's parameters, the *Contention Window* determines how long a node needs to wait before it (re)transmits data. Consequently, choosing adequate values of the *Contention Window* is crucial for IEEE 802.11's performance. In this paper, we introduce a simple, yet effective machine learning approach to adjust the value of IEEE 802.11's *Contention Window* based on present- as well as recent past network contention conditions. Using a wide range of network scenarios and conditions, we show that our approach outperforms both 802.11's BEB as well as an existing contention window adjustment technique that only considers the last two transmissions. Our results indicate that our contention window adaptation algorithm is able to deliver consistently higher average throughput, lower end-to-end delay, as well as improved fairness.

KEYWORDS

machine learning, contention window, IEEE 802.11, backoff, experts, Fixed-share

ACM Reference Format:
Yalda Edalat and Katia Obraczka. 2019. Dynamically Tuning IEEE 802.11's Contention Window Using Machine Learning. In *22nd Int'l ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '19), November 25–29, 2019, Miami Beach, FL, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3345768.3355920

1 INTRODUCTION

The IEEE 802.11 standard, also known as WiFi, specifies two types of MAC protocols, namely the Distributed Coordination Function (DCF) and the Point Coordination Function (PCF). DCF is IEEE 802.11's most widely used medium access mechanism and uses the Carrier Sensing Multiple Access/Collision Avoidance (CSMA/CA) protocol[1]. CSMA/CA arbitrates access to the shared communication medium using a contention-based, on-demand distributed mechanism. One of the key components of IEEE 802.11's DCF is the Binary Exponential Backoff (BEB) algorithm which was introduced to mitigate channel contention and prevent collisions of packets simultaneously transmitted by multiple stations. It delays the retransmission of a collided packet by a random time, chosen uniformly over n slots ($n > 1$), where n is a parameter called *Contention Window*, or (CW). The BEB algorithm works as follows: CW is initially set based on a pre-specified minimum value, (CW_{min}). If a collision happens, the station chooses an exponentially increased CW until it reaches CW's pre-specified maximum value (CW_{max}). As such, CW can significantly impact IEEE 802.11's performance. Choosing small CW values may result in more collisions and backoffs. On the other hand, choosing large CW may result in unnecessary idle airtime and additional delay. In either case, the channel is not used efficiently. Therefore, the value of CW should be adjusted considering the actual level of contention in the channel.

In this paper, we develop a simple, yet effective machine learning approach based on the *Fixed-Share* technique [8] [9] to adjust the value of CW based on recent past network contention. Unlike the original BEB algorithm which increases or decreases CW based solely on the status of the most recently transmitted packet, our method also accounts for recent network contention conditions in addition to last packet's transmission status. We evaluated our contention window adaptation algorithm using a wide range of scenarios including infrastructure-based as well as multi-hop adhoc environments. Our results indicate that our contention window adaptation algorithm is able to deliver consistently higher average throughput, lower end-to-end delay, as well as improved fairness.

The rest of this paper is organized as follows. Section 2 provides a brief overview of IEEE 802.11's Binary Exponential Backoff (BEB) algorithm and presents related work. Our machine learning based approach to dynamically adjust 802.11's contention window is described in Section 3. Section 4 and Section 5 present our experimental methodology and results, respectively. Section 6 concludes the paper and discusses directions for future work.

2 BACKGROUND AND RELATED WORK

IEEE 802.11's Binary Exponential Backoff (BEB) algorithm was introduced in order to decrease the chance of collision of frames simultaneously (re)transmitted by multiple stations. In the original BEB algorithm, if a node wants to transmit a data frame, it first senses the channel for a *DCF Inter frame Space (DIFS) interval* to check whether the channel is idle. If the channel is sensed idle, the node transmits the data packet immediately. Otherwise, i.e., if the channel is busy, the node selects a random backoff time value between 0 and CW (as shown in Equation 1) to re-try and avoid

[1]DCF provides two modes of operation: the *Base Mode* which uses CSMA and the *Collision Avoidance Mode*, which uses CSMA/CA.

MSWiM '19, November 25–29, 2019, Miami Beach, FL, USA

ACM ISBN 978-1-4503-6904-6/19/11…$15.00
https://doi.org/10.1145/3345768.3355920

collisions. The backoff time is decremented every slot thereafter when the node senses the medium idle. When the backoff time reaches zero, the node can then initiate transmission.

$$Backoff\ time = random\,[0, CW] \times slot\ time \quad (1)$$

If the transmission is unsuccessful, CW will be doubled for the next transmission up to a maximum value specified by CW_{max}. In the case of a successful transmission, CW is reset to CW_{min}.

A number of drawbacks with the original BEB algorithm have been identified. Fairness is one of them; for instance, resetting CW to CW_{min} after a successful transmission may cause the node who succeeds in transmitting to dominate the channel for an arbitrarily long period of time. As a result, other nodes may suffer from severe short-term unfairness. Additionally, the current state of the network (e.g., load) should be taken into account in selecting the most appropriate backoff interval.

Motivated by BEB's impact on the performance of 802.11, considerable attention from network researchers and practitioners has focused on optimizing IEEE 802.11's backoff algorithm.

We categorize related work on improving IEEE 802.11's BEB performance in two groups: the first group focuses on how to increase or decrease the size of CW, whereas the second group targets setting the values of CW_{min} and CW_{max}, i.e., CW's upper and lower bounds. Due to space constraints, we only list references to efforts more directly related to our work in each group.

Increasing/decreasing CW

Under this category we highlight the MACAW protocol [3], the History-Based Adaptive Backoff (HBAB) algorithm [1], the Inverse Binary Exponential Backoff (iBEB) approach [2], the Binary Negative-Exponential Backoff (BNEB) [10] protocol, and the New Binary Exponential Back-off (N-BEB) [14].

Setting CW's lower and upper bounds

Another group of papers focuses on optimizing the values of CW_{min} and CW_{max}. Approaches that fall under this category include IEEE 802.11e, [11], and [7].

Using machine learning to improve network protocol performance

Recently, machine learning (ML) techniques have gained significant traction and have been used in a wide range of applications. Specifically in the context on computer networking performance management, in [12], the Fixed-Share algorithm is used to estimate TCP's round-trip time. In our prior work [5], we estimate collision rate using an algorithm called SENSE which employs a combination of Fixed-Share and Exponentially-Weighted Moving Average (EWMA). In [6], SENSE estimates network contention which is then used to enable/disable RTS/CTS in IEEE 802.11 networks.

3 AUTOMATICALLY ADJUSTING IEEE 802.11'S CONGESTION WINDOW

In this section, we introduce the proposed machine learning based contention window adaptation approach. As the complexity and heterogeneity of networks and their applications grow, the use of ML techniques to adequately manage network in order to meet application requirements becomes increasingly attractive for a number of reasons. For instance, machine learning algorithms can learn and adapt to network and application dynamics autonomically. Some ML techniques do not require a-priori knowledge of the operating environment; they acquire this knowledge as they operate and adjust accordingly without needing complex mathematical models of the system.

To the best of our knowledge, our proposed algorithm is the first to use ML to automatically adjust IEEE 802.11's CW based on packet transmission history. We should point out that one of our main goals was to design an algorithm that achieves significant performance gains, yet is simple, low cost, low overhead, and easy to implement. To this end, we designed a simple, yet effective algorithm based on the *Fixed-Share algorithm* [8] to tune IEEE 802.11's CW. We start with a brief description of the Fixed-Share algorithm.

3.1 Fixed-Share Algorithm

The Fixed-Share algorithm is part of the Multiplicative Weight algorithmic family which has shown to yield performance improvements in a variety of on-line problems [8], [12]. This family of algorithms combines predictions of a set of experts $\{x_1,x_2,...,x_N\}$ to calculate the overall prediction denoted by $\hat{y}_t$. Each expert has a weight $\{w_1,w_2,...,w_N\}$ representing the impact of that expert on the overall predictor. Based on the difference between each expert's prediction and the real data represented by y_t, the weight of each expert is updated [5]. Algorithm 1 shows Fixed-Share Experts' pseudo-code. Each expert is initialized with a value within the range of the quantity to be predicted and the weight of all experts is initialized to $\frac{1}{N}$, where N is the number of experts. At every iteration, based on each expert's current weight and value, the prediction for the next trial is calculated as shown in the **Prediction** step of the algorithm. The **Loss Function** step then checks how good the prediction of each expert was using a loss function $L_{i,t}(x_i, y_t)$. The result of the loss function loss for each expert is used in the **Exponential Update** step to adjust the experts' weights by multiplying the current weight of the $i-th$ expert by $e^{-\eta \times L_{i,t}(x_i, y_t)}$. The *learning rate* η is used to determine how fast the updates will take effect, dictating how rapidly the weights of misleading experts will be reduced. Finally, in the **Sharing Weights** step, a fixed fraction of the weights of experts that are performing well is shared among the other experts. The goal of this step is to prevent large differences among experts' weights [9]. The amount of sharing can be adjusted through the *sharing rate* parameter α.

3.2 Proposed Approach

We propose a modified version of the Fixed-Share algorithm to dynamically set IEEE 802.11's CW. More specifically, as illustrated in Algorithm 2, we design loss- and gain functions that account for current network conditions. Our proposed technique works as follows. Similarly to the standard Fixed-Share algorithm (Algorithm 1), in the **Initialization** step in Algorithm 2, the weight of all experts is set to $\frac{1}{N}$, where N is the number of experts. Each expert is assigned a fixed value within the range of $[CW_{min}, CW_{max}]$. In our current implementation, we assign the values of 15, 22, 33, 50,

Algorithm 1 Fixed-Share Algorithm

Parameters:

$$\eta > 0, 0 \leq \alpha \leq 1$$

Initialization:

$$w_{1,1}=...=w_{N,1}=\frac{1}{N}$$

Prediction:

$$\hat{y}_t=\frac{\sum_1^N w_{i,t}\times x_i}{\sum_1^N w_{i,t}}$$

Loss Function:

$$L_{i,t}(x_i, y_t) = \begin{cases}(x_i - y_t)^2 & , x_i \geq y_t \\ 2 \times y_t & , x_i < y_t\end{cases}$$

Exponential Update:

$$\acute{w}_{i,t} = w_{i,t} \times e^{-\eta \times L_{i,t}(x_i, y_t)}$$

Sharing Weights:

$$Pool = \sum_{i=1}^{N} \alpha \times \acute{w}_{i,t} \qquad w_{i,t+1} = (1-\alpha) \times \acute{w}_{i,t} + \frac{1}{N} \times Pool$$

75, 113, 170, 256, 384, 576, 865, and 1023 to 12 experts forming a geometric sequence with ratio of 1.5.

The reason we pick these values is that first of all we wanted to keep CW_{min} and CW_{max} unchanged according to the IEEE 802.11 standard. Furthermore, since BEB's adjustment is considered to be quite aggressive [3], we employ a multiplicative factor of 1.5 which yields less drastic backoff process.

Additionally, we have experimented with different numbers and values of experts and have not seen any significant change in the results. Due to space limitations, we do not include these results.

In the **CW Calculation** step, $C\hat{W}_t$, which is CW's estimate for time t, is calculated based on the current value of the experts and their weights. Clearly, experts with more weight will have more influence on the next CW. In the **Loss/Gain Function** step, the performance of all experts is evaluated based on their value, $C\hat{W}_t$, and whether the previous packet transmission was successful or not.

Loss/Gain Function: The loss and gain functions are designed to adjust CW based on present- as well as recent past network conditions. Our loss/gain function works as follows: if a packet is transmitted successfully, it means that there may be additional bandwidth available in the network. Therefore, we reduce the weight of the experts higher than $C\hat{W}_t$ because they are less aggressive experts. Weight reduction is done proportional to the difference between the value of that expert and CW which means the higher the expert is, the more aggressively its weight will be reduced. We also increase the weight of experts lower than $C\hat{W}_t$ because we want to push for potentially more aggressive CW. This weight increase is done proportional to the value of the expert. Experts with value closer to the current CW will experience higher weight increase. For the experts with value much lower than the current CW, the risk of failure is higher, therefore their weight increase is lower. These weight decrease and increase of experts will result in a lower value for the next CW and, as a result, the next transmission will be scheduled more aggressively.

Analogously, in the case of unsuccessful transmissions, the loss/gain function will increase the weight of experts with values higher than $C\hat{W}_t$ and reduce the weight of experts with values lower than $C\hat{W}_t$. This will result in higher CW for the next packet transmission and less chance of collision.

Overhead: The overhead incurred by our algorithm is a function of the number of experts used. There is a cost-performance tradeoff between the number of experts and how well the algorithm can capture network dynamics. However, as previously discussed, there is a diminishing returns effect, wherein beyond a certain number of experts, there is minimal performance impact. As far as storage overhead, additional storage is used to keep the experts' values and their weights. As for computation overhead, assuming the contention window is adjusted at every attempted transmission, the **CW Calcultation**, **Loss/Gain Function**, and **Sharing Weights** steps in Algorithm 2 are executed. These involve simple arithmetic operations and are not computationally onerous, which are consistent with one of our main design goals, i.e., developing a simple, light-weight algorithm that can run at line rate. In fact, one of our directions for future work is to implement our algorithm in a real testbed to validate its performance.

Algorithm 2 Proposed Algorithm

Initialization:

$$w_{1,1}=...=w_{N,1}=\frac{1}{N}$$
$$x_1 = CW_1, x_2 = CW_2, ..., x_N = CW_N,$$

CW Calculation:

$$C\hat{W}_t=\lfloor\frac{\sum_1^N w_{i,t}\times x_i}{\sum_1^N w_{i,t}}\rfloor$$

Loss/Gain Function:

- If packet received successfully:

$$w_{i,t+1} = \begin{cases}[1 - \frac{x_i - C\hat{W}_t}{x_i}] \times w_{i,t} & , x_i > C\hat{W}_t \\ [1 + \frac{x_i}{C\hat{W}_t}] \times w_{i,t} & , x_i \leq C\hat{W}_t\end{cases}$$

- If packet is not received successfully:

$$w_{i,t+1} = \begin{cases}[1 + \frac{C\hat{W}_t}{x_i}] \times w_{i,t} & , x_i > C\hat{W}_t \\ [1 - \frac{C\hat{W}_t - x_i}{C\hat{W}_t}] \times w_{i,t} & , x_i \leq C\hat{W}_t\end{cases}$$

Sharing Weights:

$$Pool = \sum_{i=1}^{N} \alpha \times \acute{w}_{i,t} \qquad w_{i,t+1} = (1-\alpha) \times \acute{w}_{i,t} + \frac{1}{N} \times Pool$$

4 EXPERIMENTAL METHODOLOGY

In this section, we describe our experimental setup including the scenarios, traffic loads, as well as performance metrics used when evaluating the proposed approach. We compare the performance of our technique against both the original IEEE 802.11 contention window adjustment technique as well as the History-Based Adaptive Backoff (HBAB) algorithm [1]. As such, we also provide a brief overview of HBAB.

4.1 Experimental Setup

We ran experiments using the *ns-3* [4] network simulator and its implementation of the IEEE 802.11n for both infrastructure-based and ad-hoc network scenarios. In our simulations, we use typologies with 100 nodes randomly placed in a $1000x1000m_2$ area. In order to

vary network contention conditions, we vary the number of sender nodes. We explore how dynamically our method is able to adjust the contention window and its effect on network performance. Table 1 summarizes the parameters describing our experimental setup and their values. Note that AODV [13] routing was used only in the multi-hop ad-hoc experiments.

Traffic Load: We used synthetic data traces as well as traces collected in real networks to drive our simulations. Table 2 summarizes the synthetic data parameters and their values. Our real traffic traces ere collected in two different settings, namely: (1) a public hot spot and (2) a company campus network using a wireless sniffer (Table 3). Note that since there are 10 and 5 individual flows in the hot spot and company traces, respectively, we replicate these flows in scenarios with higher number of nodes.

Performance Metrics: We evaluate our contention window adjustment technique by comparing its performance against IEEE 802.11's original mechanism as well as HBAB [1]. As performance metrics, we use average throughput and average end-to-end delay. Average throughput is calculated as the ratio between the number of received packets and the total number of transmitted packets averaged over all nodes. Average end-to-end delay is given by the interval of time between when a packet was received and when it was sent averaged over all received packets. Channel access fairness is an important issue in MAC protocol design. As such, we also evaluate the proposed approach's fairness by comparing its minimum, maximum, and average throughput against those of IEEE 802.11's BEB and HBAB.

Table 1: Simulation setup parameters and their values

Area	1000mx1000m
Number of nodes	100
Traffic	CBR and real traces
IEEE 802.11 Version	802.11n
Number of experts	12
CW_{min}	15
CW_{max}	1023
Routing protocol	AODV

Table 2: Synthetic trace

Simulation time	200s
Traffic type	CBR
Frame size	1024 Bytes
Data rate	54 Mbps

4.2 History-Based Adaptive Backoff

We use HBAB in the performance evaluation of our proposed contention window adjustment mechanism as it represents mechanisms that, similarly to ours, use transmission status history to set CW. History-Based Adaptive Backoff (HBAB) [1] increases or decreases the congestion window CW based on the current- as well as past data transmission trials. HBAB defines two parameters

Table 3: Hot spot and company trace

	Hot spot	Company
Location	Coffee shop	Company campus
Number of flows	10	5
Duration	20 minutes	30 minutes
Frame size	34-2150 byte range	34-11000 byte range
802.11 version	802.11n	802.11n

α and N; α is a multiplicative factor used to update CW and N is the number of past transmission trials considered by the algorithm. The outcome of the previous N transmission trials is stored in $ChannelState$; failed transmissions are represented by 0 while successful ones bt 1. For example, if $N = 2$, $ChannelState$ = {0,1} means that the last transmission succeeded but the previous one failed. Larger values of N mean larger windows into the past but require, albeit relatively small, additional memory.

Algorithm 3 shows HBAB's pseudo-code. Note that we follow HBAB's implementation in [1] and use $\alpha = 1.2$ and $N = 2$, i.e., HBAB examines the status of the two previous and consecutive data transmissions, as well as the current one, to make a decision on how to adjust CW. In case the current transmission is successful, but the two previous transmissions failed, i.e., $ChannelState[0] = 0$ and $ChannelState[1] = 0$, the new value of CW is set to the current CW divided by α. Otherwise, CW is set to CW_{min}. In case the current transmission is unsuccessful, CW is multiplied by α.

As illustrated in Algorithm 3, HBAB's original design presented in [1] is described only for $N = 2$. Another reason that we do not use $N > 2$ in our implementation of HBAB is because, as shown in Algorithm 3, HBAB's state space grows with N which means that we would need to define "manually" how to adjust CW for all the possible outcomes of the previous N transmissions.

Algorithm 3 HBAB Algorithm

Initialization:

$$CW = CW_{min}, \alpha > 1$$
$$ChannelState[0] = 1, ChannelState[1] = 1$$

If current transmission succeeds:

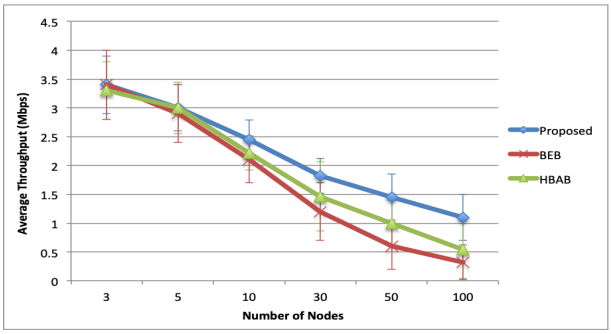
$$CW = \begin{cases} \frac{CW}{\alpha} & , ChannelState[0] = 0 \\ & and\ ChannelState[1] = 0 \\ CW_{min} & , otherwise \end{cases}$$

If transmission failed:

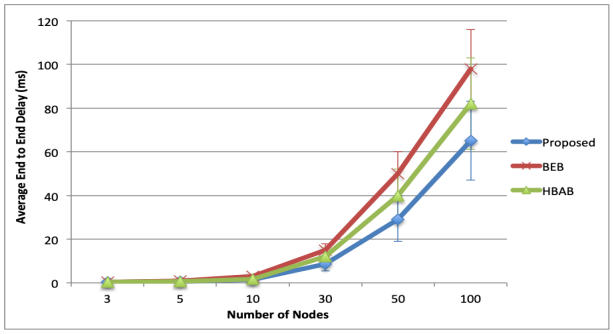
$$CW = CW \times \alpha$$

ChannelState update:

$$\begin{cases} ChannelState[0] = ChannelState[1] \\ ChannelState[1] = 0, last\ transmission\ failed \\ ChannelState[1] = 1, last\ transmission\ succeeded \end{cases}$$

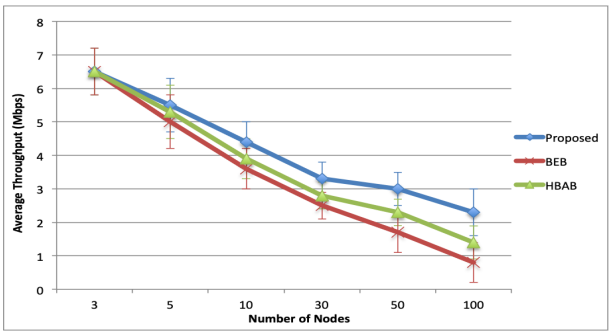


(a) Average throughput

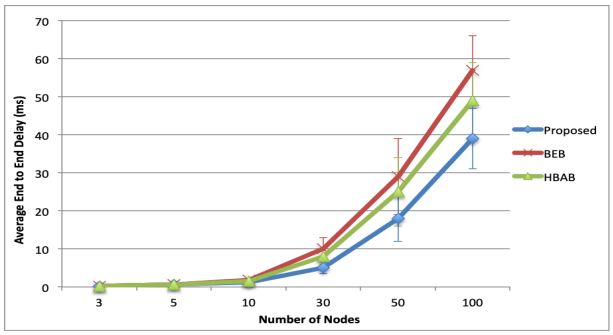


(b) Average delay

Figure 1: Average throughput and delay as a function of number of senders for hot-spot traffic trace in infrastructure-based scenario



(a) Average throughput



(b) Average delay

Figure 2: Average throughput and delay as a function of the number of senders for company campus traffic trace in infrastructure-based scenario

5 RESULTS

As described in Section 4, we evaluate our approach using two types of scenarios, namely: infrastructure-based and infrastructure-less (or multi-hop ad-hoc) networks. We start by presenting results obtained for the infrastructure-based scenarios followed by the infrastructure-less scenario results. In all graphs, each data point is calculated by averaging over 10 runs that use different random seeds.

5.1 Infrastructure-based Scenarios

In the infrastructure-based experiments, randomly selected nodes send traffic to the Access Point (AP) which is placed in the center of the area being simulated. We drive the experiments using the synthetic and real (i.e., hot spot and company campus) traffic traces described in Section 4 and vary the number of senders as follows: 3, 5, 10, 30, 50, and 100.

Average Throughput and End-to-end Delay: We compare the average throughput and end-to-end delay of our method against IEEE 802.11's BEB and HBAB for different number of nodes and traffic traces, i.e., synthetic, hot-spot, and company campus data traces. We observe in all traces, similar trends for both average throughput and end-to-end delay. Figures 1, and 2 show the average throughput and end-to-end delay of our method, BEB and HBAB. Note that, due to space limitations, we do not include average throughput and end-to-end delay results for the synthetic dataset. As expected, average throughput decreases and end-to-end delay increases as the number of senders increases. For lower number of senders, e.g., 3 and 5, all three algorithms perform similarly. However, as the number of senders increases resulting in higher network contention, our approach is able to achieve better average throughput and end-to-end delay performance when compared to IEEE 802.11's BEB and HBAB for all three traffic traces.

Table 4 summarizes the throughput and delay improvement achieved by our congestion window adaptation algorithm when compared to BEB's and HBAB's for 100 senders in the infrastructure-based scenario for all traffic traces. We observe that in such more heavily loaded environments, our approach is able to achieve significant gains both in throughput (up to 220% over BEB and 92% over HBAB) as well as in end-to-end delay (up to 33% over BEB and up to 21% over HBAB).

Fairness: In order to evaluate the ability of our contention window adaptation algorithm to provide a fair share of the channel to participating stations, Table 5 shows the minimum, average, and

Table 4: Throughput and delay improvement of proposed congestion window adaptation algorithm compared to IEEE 802.11's BEB and HBAB in infrastructure-based scenario with 100 senders for all traffic traces

	BEB Throughput	HBAB Throughput	BEB Delay	HBAB Delay
Synthetic	180%	90%	28%	12%
Hot-spot	220%	92%	33%	20%
Company	170%	64%	31%	21%

Table 5: Minimum, average, and maximum throughput, and standard deviation achieved by our approach, BEB, and HBAB for synthetic data trace in infrastructure-based scenario with 100 senders

	Minimum (Mbps)	Average (Mbps)	Maximum (Mbps)	Standard Deviation
Proposed	0.64	1.2	1.96	0.48
BEB	0	0.51	2.12	0.89
HBAB	0	0.75	1.56	0.62

maximum throughput reported by stations when using our algorithm compared against BEB and HBAB for the synthetic data trace in the infrastructure-based scenario with 100 senders. Both the difference between the maximum and minimum throughput as well as the standard deviation (also reported in Table 5) show that our approach yields superior fairness performance when compared to both BEB and HBAB. As previously discussed, the main reason for BEB's less fair channel allocation is due to the reset of CW to CW_{min} upon a successful transmission, which gives certain nodes higher chance to seize the channel. HBAB shows improvement over BEB's fairness by avoiding immediate reset of CW to CW_{min} after single successful transmission, but still only considers short term packet transmission history which results in less fair channel allocation when compared to our approach. We should point out that BEB is able to yield the highest maximum throughput which is consistent with its resetting of CW to CW_{min} upon a successful transmission.

The graphs in Figure 3 showing CW variation over time for the nodes with minimum and maximum throughput for the synthetic trace in the infrastructure-based scenario with 100 senders reiterate our observations. We notice from Figure 3 that for both BEB and HBAB, CW for the node that reports the minimum throughput stays practically constant at CW_{max} for almost the whole experiment. In the case of the maximum throughput node, its CW varies considerably between CW_{min} and CW_{max}, i.e., 1023, during the whole run under both BEB and HBAB. Under our approach, the maximum throughput node's CW is able to reach steady state quite fast around 400.

CW **Variation:** In Figure 3a which shows the CW variation for the node with maximum throughput, we observe significant CW oscillation between CW_{min} and CW_{max} under BEB and HBAB. In the case of our approach, CW stays fairly constant throughout the experiment. The reason is that, after each successful transmission, the weight of experts with value higher than the current CW will be reduced and the weight of experts with value lower than CW will be increased. Therefore, for the next transmission, since the CW is calculated as the weighted sum of all experts, its value decreases slowly. Also, in the case of unsuccessful transmission, the weight of experts with values higher than current CW are increased and the weight of experts with values lower than CW are decreased. And again, since the CW is calculated as the weighted sum of all experts, the next CW will be slightly higher for the next transmission. In other words, through the experts and their weights, our approach is able to account for recent past as well as the present.

Figure 3b shows the variation of CW over time for the node with the lowest average throughput in the infrastructure-based scenario with 100 senders using the synthetic traffic trace. As the results in Table 5 indicate, BEB's and HBAB's minimum throughput is 0 which indicates that there are some nodes in the network that suffer from starvation. From Figure 3b, we observe that, relatively early in the experiment, CW of the node with the lowest throughput stabilizes at CW_{max} which considerably decreased the node's chance to acquire the channel, ultimately resulting in "starvation", i.e., zero throughput.

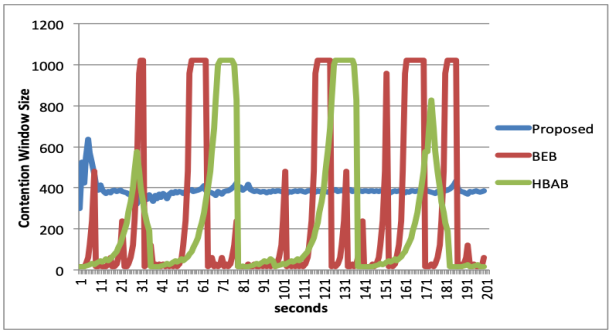
5.2 Infrastructure-less Scenarios

In the ad-hoc experiments, randomly selected senders send data traffic to randomly selected receivers according to the three traffic traces described in Section 4. Similarly to the infrastructure-based experiments, the number of senders vary as follows: 3, 5, 10, 30, 50, and 100.

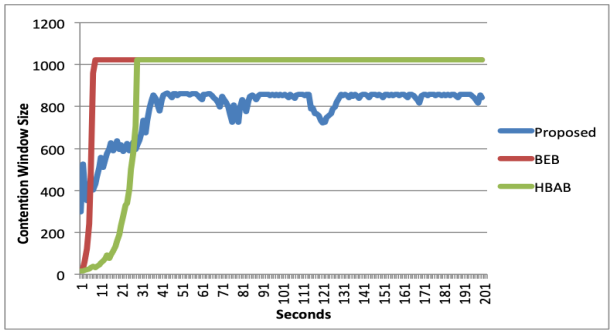
Average Throughput and End-to-end Delay: We compare the average throughput and end-to-end delay of our method compared with BEB and HBAB for different number of senders and traffic traces in the ad-hoc scenario. Similarly to the trend reported in the infrastructure-based experiments, we observe that, for lower number of senders, all three methods perform similarly. However, when the number of senders increase, which result in higher network contention, our method is able to achieve higher average throughput and lower average end-to-end delay when compared to both BEB and HBAB. Figures 4 and 5 show the average throughput and end-to-end delay of our method compared versus BEB and HBAB. Similarly to the infrastructure-based experiments, we do not include average throughput and end-to-end delay results for the synthetic dataset due to space limitations.

Table 6 summarizes the throughput and delay improvement achieved by our congestion window adaptation algorithm when compared to BEB's and HBAB's for 100 senders in the ad-hoc scenario for all traffic traces. Similarly to what was observed for the infrastructure-based experiment, in high contention networks, our approach yields significant improvement both in average throughput (up to 257% over BEB and 78% over HBAB) and average end-to-end delay (up to 37% over BEB and 23% over HBAB).

Fairness: To evaluate our algorithm's fairness in ad-hoc scenarios, we show the minimum, average, and maximum throughput for the synthetic traffic trace with 100 senders in Table 7. Like the results



(a) Maximum throughput



(b) Minimum throughput

Figure 3: Contention window size variation over time for the nodes with minimum and maximum throughput for synthetic trace in infrastructure-based scenario with 100 senders

Table 6: Throughput and delay improvement of proposed congestion window adaptation algorithm compared to IEEE 802.11's BEB and HBAB in ad-hoc scenario with 100 senders for all traffic traces

	BEB Throughput	HBAB Throughput	BEB Delay	HBAB Delay
Synthetic	230%	75%	31%	21%
Hot-spot	240%	78%	37%	23%
Company	257%	63%	35%	17%

Table 7: Minimum, average, and maximum throughput, and standard deviation achieved by our approach, BEB, and HBAB for synthetic data trace in ad-hoc scenario with 100 senders

	Minimum	Average	Maximum	Standard Deviation
Proposed	0.43	1.05	1.6	0.41
BEB	0	0.3	1.8	0.75
HBAB	0	0.6	1.2	0.52

reported for the infrastructure-based experiments, our approach is able to reduce the gap between the minimum and maximum average throughput with a lower standard deviation, an indication of its ability to deliver improved fairness when compared to BEB and HAB.

CW **Variation:** Figure 6 shows *CW* variation over time for both the nodes that yield the maximum and minimum average throughput under our approach as well as under BEB and HBAB in the ad-hoc scenario with 100 senders using the synthetic traffic trace. Like the trend observed in the infrastructure-based experiments, our approach is able to achieve steady state relatively quickly for both the nodes with maximum- and minimum throughput. The graphs in Figure 6 also show that our approach is able to close the gap between the *CW*s of the highest- and lowest throughput nodes which is another indication of improved fairness.

6 CONCLUSION

In this paper, we introduce a simple, yet effective machine learning approach to adjust the value of IEEE 802.11's *Contention Window* based on present- as well as recent past network contention conditions. Using a wide range of network scenarios and conditions, we show that our approach outperforms both 802.11's BEB as well as an existing contention window adjustment technique that only considers the last two transmissions. Our results indicate that our contention window adaptation algorithm is able to deliver consistently higher average throughput, lower end-to-end delay, as well as improved fairness. As future work, we plan to explore alternate loss functions as well as validate our approach in real testbeds.

ACKNOWLEDGMENTS

This research has been partly supported by grant CNS 1321151 from the US National Science Foundation.

REFERENCES

[1] Maali Albalt and Qassim Nasir. 2009. Adaptive backoff algorithm for IEEE 802.11 MAC protocol. *International Journal of Communications, Network and System Sciences* 2, 04 (2009), 300.

[2] Khaled Hatem Almotairi. 2013. Inverse binary exponential backoff: Enhancing short-term fairness for IEEE 802.11 networks. In *ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems*. VDE, 1–5.

[3] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. 1994. MACAW: a media access protocol for wireless LAN's. *ACM SIGCOMM Computer Communication Review* 24, 4 (1994), 212–225.

[4] Gustavo Carneiro. 2010. NS-3: Network simulator 3. In *UTM Lab Meeting April*, Vol. 20. 4–5.

[5] Yalda Edalat, Jong-Suk Ahn, and Katia Obraczka. 2016. Smart experts for network state estimation. *IEEE Transactions on Network and Service Management* 13, 3 (2016), 622–635.

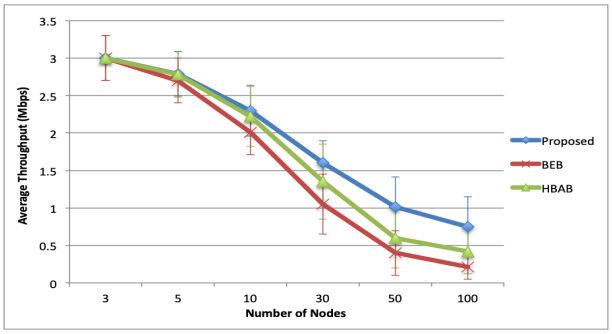
[6] Yalda Edalat, Katia Obraczka, and Bahador Amiri. 2018. A machine learning approach for dynamic control of RTS/CTS in WLANs. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 432–442.

[7] Lassaad Gannoune. 2006. A Non-linear Dynamic Tuning of the Minimum Contention Window (CW min) for Enhanced Service Differentiation in IEEE 802.11 ad-hoc Networks. In *2006 IEEE 63rd Vehicular Technology Conference*, Vol. 3. IEEE, 1266–1271.

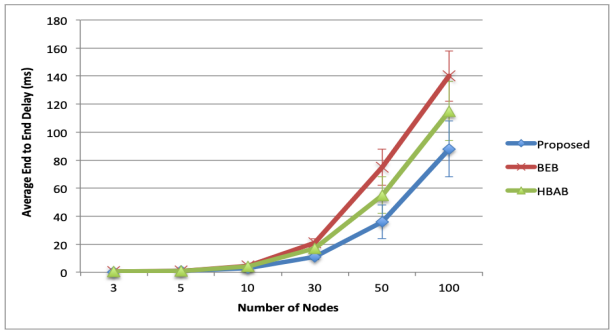
[8] David P Helmbold, Darrell DE Long, Tracey L Sconyers, and Bruce Sherrod. 2000. Adaptive disk spin-down for mobile computers. *Mobile Networks and Applications* 5, 4 (2000), 285–297.

[9] Mark Herbster and Manfred K Warmuth. 1998. Tracking the best expert. *Machine learning* 32, 2 (1998), 151–178.

[10] Hyung Joo Ki, Seung-Hyuk Choi, Min Young Chung, and Tae-Jin Lee. 2006. Performance evaluation of binary negative-exponential backoff algorithm in

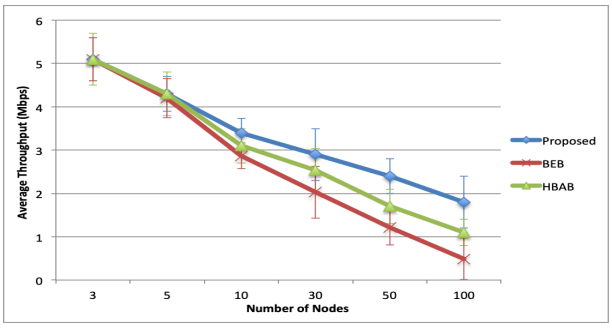


(a) Average throughput

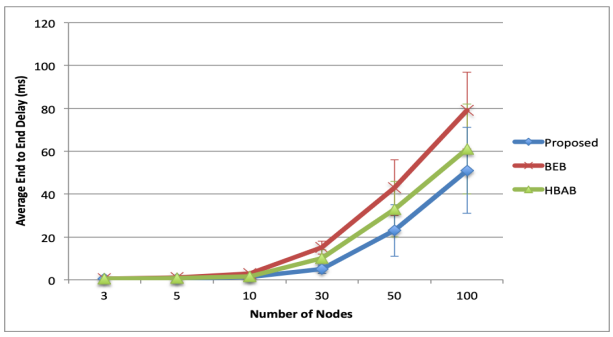


(b) Average delay

Figure 4: Average throughput and delay as a function of the number of senders for hot-spot data in ad-hoc scenarios

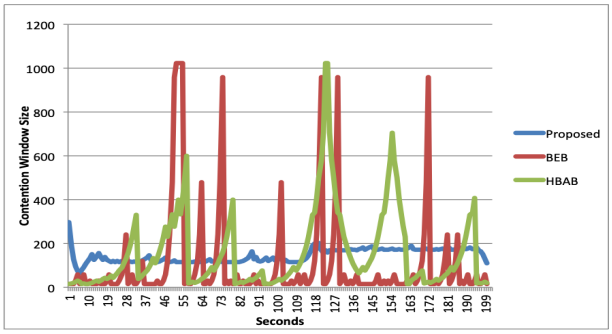


(a) Average throughput

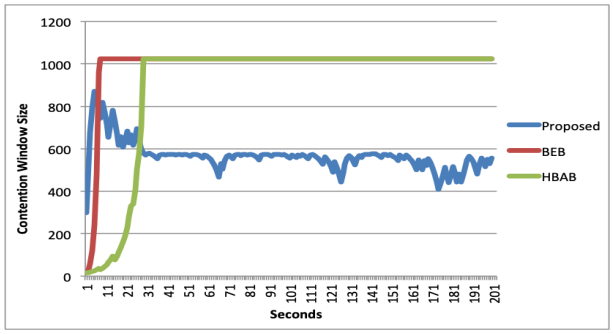


(b) Average delay

Figure 5: Average throughput and delay as a function of the number of nodes for company data in ad-hoc scenarios



(a) Maximum throughput



(b) Minimum throughput

Figure 6: Contention window size variation over time for the nodes with minimum and maximum throughput for synthetic trace in ad-hoc scenario with 100 senders

IEEE 802.11 WLAN. In *International Conference on Mobile Ad-Hoc and Sensor Networks*. Springer, 294–303.

[11] Adlen Ksentini, Abdelhamid Nafaa, Abdelhak Gueroui, and Mohamed Naimi. 2005. Determinist contention window algorithm for IEEE 802.11. In *2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications*, Vol. 4. IEEE, 2712–2716.

[12] Bruno Astuto Arouche Nunes, Kerry Veenstra, William Ballenthin, Stephanie Lukin, and Katia Obraczka. 2014. A machine learning framework for TCP round-trip time estimation. *EURASIP Journal on Wireless Communications and Networking* 2014, 1 (2014), 47.

[13] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. 2003. *Ad hoc on-demand distance vector (AODV) routing*. Technical Report.

[14] Mohammad Shurman, Bilal Al-Shua'b, Mohammad Alsaedeen, Mamoun F Al-Mistarihi, and Khalid A Darabkh. 2014. N-BEB: New backoff algorithm for IEEE 802.11 MAC protocol. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 540–544.