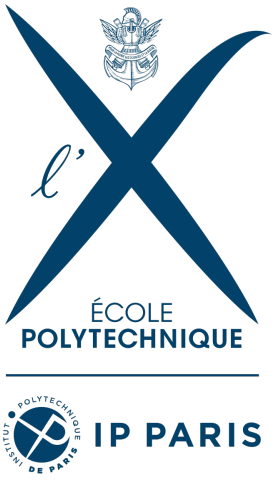

Adaptive CSMA/CA for Wireless Networks: Examining Reinforcement Learning Method

BRUDASTOV, Artem

CHO, Won Suk

CSC 52067 EP - Réseaux sans fil : du Cellulaire aux Objets Connectés

March 2025

Contents

1 Introduction **1**
- 1.1 Background of the Study . . . 1
- 1.2 Research Objectives . . . 2
- 1.3 Scope and Limitations . . . 2

2 Review of Related Literature **3**
- 2.1 CSMA/CA Limitations . . . 3
- 2.2 Analytical Foundations for CSMA/CA . . . 3
  - 2.2.1 Saturation Throughput . . . 4
  - 2.2.2 Average Delay . . . 4
- 2.3 CSMA/CA Performance Trends . . . 4
- 2.4 Adaptive CSMA/CA . . . 5
  - 2.4.1 Rule-Based Adaptations . . . 5
  - 2.4.2 RL Based Approaches . . . 5

3 Design and Implementation **6**
- 3.1 Simulator Architecture . . . 6
  - 3.1.1 Backoff Procedure . . . 6
  - 3.1.2 Collision Detection . . . 6
  - 3.1.3 CW Update Logic . . . 7
  - 3.1.4 State Tracking for RL Agent . . . 7
- 3.2 Simulation Environments . . . 7
  - 3.2.1 Original Environment (Progressive Load) . . . 7
  - 3.2.2 Simplified Environment (6 Fixed Stations with Random Traffic) . . . 7
- 3.3 Integration of RL Agent . . . 7

4 Code Documentation **9**
- 4.1 Code Overview . . . 9
- 4.2 RL Agent Logic . . . 9

5 Results and Analysis **10**
- 5.1 Graphical Analysis of CW Dynamics . . . 10
  - 5.1.1 Simplified Environment with 6 Stations, BEB . . . 11
  - 5.1.2 Simplified Environment with 6 Stations, RL . . . 11
  - 5.1.3 Progressive Stations, BEB . . . 12
  - 5.1.4 Progressive Stations, RL . . . 12
  - 5.1.5 Key observations . . . 13
- 5.2 Numerical Analysis of Performance Metrics . . . 13
  - 5.2.1 Simplified Model with 6 Stations . . . 13
  - 5.2.2 Progressive Stations . . . 14

5.3 Comparative Analysis: BEB vs RL 14
5.4 Interpretation and Insights 14

6 Conclusion and Recommendations **15**
6.1 Key Results 15
6.2 Limitations 15
6.3 Future Work 15

Chapter 1

Introduction

1.1 Background of the Study

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is widely used in wireless protocols, such use case can be seen in IEEE 802.11 (Wi-Fi). The *Contention Window* (CW) is a parameter that the protocol uses to avoid collision. This is done by applying the CW to the backoff strategy. However, the static nature of choosing the CW size with the use of Binary exponential backoff (BEB) struggles to keep it efficient in different network conditions [6, 9, 5]. Finding the right balance between small CW values and large CW values is a challenge as each has its own drawbacks, which we will delve deeper into in this research.

Why This Research Is Important: Currently, the CSMA/CA relies on a fixed CW size algorithm, like BEB. The network environments do not always have fixed parameters and same scenarios, it varies a lot on how the network is used for. The stochastic nature of the network environment calls for the need of dynamically adjustable CW size. As a result, this will lead to a more efficient network environment.

Motivation for Using Reinforcement Learning: Recent research highlights how *Reinforcement Learning* (RL) may be able to improve the adaptability of CSMA/CA [4, 1, 3]. The RL agent would be able to learn from past knowledge (depending on how it was trained) and choose the right and most efficient (with the highest reward) CW size in the environment it was put in place. Due to the stochastic nature of the network environment, using an RL agent for this use case would be optimal.

- *Reduced Collisions*: Real-time adjustments help ease simultaneous transmissions made by the station/s [1].
- *Higher Throughput*: The agent will be able to balance the backoff times proportionally to the traffic conditions, which prevents channel underuse or overuse [3].
- *More Efficient Resource Allocation*: Custom reward function can ensure fair channel access usage among multiple stations [11].

With the advantages pointed out, the researchers aim to create an *adaptive, robust, high-throughput* RL agent that will be able to calculate the CW size dynamically for the CSMA/CA protocol. This should in theory give better results than traditional backoff methods like BEB.

1.2 Research Objectives

These are the following objectives that this research aims to deliver:

1. **Develop a CSMA/CA Simulator:** Design and implement a CSMA/CA simulator with a hidden terminal that has a transmission probability.
2. **Develop an RL based on the CSMA/CA Mechanism:** Design and implement an adaptive CW size selection algorithm by using RL, in which an agent may be able to update the CW size in real-time.
3. **Train an RL agent to choose the most efficient CW size:** It will be trained on the custom developed CSMA/CA environment, and after different training iterations, the agent should be able to choose a CW size.
4. **Evaluate Network Performance:** Compare key performance metrics (collision rate, throughput, and delay) of the RL agent against standard or rule-based CSMA/CA approaches.
5. **Analyze Applicability and Scalability:** Evaluate the plausibility of its use case in a real-world scenario and its realistic widespread adaptation.

1.3 Scope and Limitations

This research develops and evaluates an RL-based CW adjustment mechanism using CSMA/CA simulations. The simulations systematically vary node count, traffic intensity, and protocol settings to demonstrate RL's advantages. Specifically, it will:

- Implement Deep Q-learning for CW adjustment.
- Evaluate throughput, collision rates, and latency under multiple traffic scenarios.
- Examine the method's behavior in low to moderately dense networks, extending to relatively high-density conditions.

Real-world hardware implementation and cross-layer factors (e.g., PHY-layer issues, channel fading) are outside this study's scope. The research focuses exclusively on single-layer, simulated CSMA/CA environments using a fixed channel model and does not address multi-layer interactions or hardware complexities.

This research uses Reinforcement Learning with CSMA/CA to reduce collisions and improve network performance and adaptability. The study tests this approach through simulations. The following sections present the methods and results clearly and directly.

Chapter 2

Review of Related Literature

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) wireless network protocols are the foundation of IEEE 802.11 and IEEE 802.15.4 standards. CSMA/CA coordinates their channel access among multiple nodes through *Contention Window* (CW) which controls the random backoff before transmission happens. Statically configured CW more often than not encounters notable inefficiencies in environments with different traffic loads. This chapter will delve into different related works on traditional CSMA/CA limitations, different analytical models, and the possibility of improving the protocol's performance through *Reinforcement Learning* (RL).

2.1 CSMA/CA Limitations

CSMA/CA relies on *Binary Exponential Backoff (BEB)* approach to handle collisions. When a collision is detected, such as missing acknowledgments, CW doubles its size in nodes until it reaches a maximum threshold [6, 9, 5]. This static and uniform adaptation is inadequate in real-world conditions. These are the following examples from the related works:

- **High Collision Rates in Dense Networks:** The initial CW is often too small with more devices competing for the channel, which leads to frequent collisions and throughput collapse [6].
- **Limited Usage of Sparse Networks:** If exponentially increased CW's traffic intensity drops, it can result in idle channel time[5].
- **Lack of Real-Time Adaptation:** The protocol bases its backoff on a fixed exponential rule, which causes suboptimal performance whenever network conditions change rapidly. Rather, it should opt out for continuous feedback from the network [8].

Additional improvements, such as *service differentiation* which gives a smaller average backoff for higher-priority data flows [9] and *Beacon Order (BO) or Superframe Order (SO) adjustments* in IEEE 802.15.4 [5] does not completely solve these problems. They offer partial optimizations that rely on static parameters, but it is still not adequate enough to reflect real-time network states

2.2 Analytical Foundations for CSMA/CA

There have been different studies to understand CSMA/CA behavior. These studies have shown the best way to understand CSMA/CA is by using Markov chain analysis of backoff state. Below are the 2 main performance indicators.

2.2.1 Saturation Throughput

The Bianchi model

In order to calculate for the maximum achievable data rate, where the network is fully loaded, which is called the saturation throughput may be calculated by using the Bianchi equation S.

$$S = \frac{P_s\, P_{tr}\, E[P]}{P_s\, T_s \;+\; P_c\, T_c \;+\; (1 - P_{tr})\, \sigma}, \tag{2.1}$$

where:

- P_s = Probability of a successful transmission in a given slot.
- P_{tr} = Probability that at least one node transmits in that slot.
- $E[P]$ = Average payload size (bits or bytes).
- T_s = Duration of a successful transmission (seconds).
- T_c = Duration of a collision event (seconds).
- σ = The base slot time (idle slot duration, in seconds).

Collisions occur when 2 or more nodes try to transmit data at once, this in result decreases the probability of successful transmission P_s, which reduces throughput S. This formula is able to highlight the importance in balancing priorities to have a higher throughput in CSMA/CA [9, 6, 7].

2.2.2 Average Delay

Average packet delay, which is denoted as D, shows the key performance metric.

$$D = \frac{E[B]\, T_{\text{slot}} \;+\; T_{\text{suc}} \;+\; (m-1)\, T_c}{1 - P_{\text{col}}}, \tag{2.2}$$

where $E[B]$ is the expected backoff duration, T_{slot} is the slot time, T_{suc} is the transmission time, m is the maximum backoff stage, and P_{col} is the collision probability [9, 6, 2].

2.3 CSMA/CA Performance Trends

There are different studies done in CSMA/CA regarding its performance in the simulation, and these are the key points that were pointed out [8, 4, 1].

- **Throughput vs. Nodes:** Throughput peaks at moderate node counts but drops sharply as collisions rise in dense networks.
- **Collisions vs. Load:** Collisions increase exponentially once the offered load exceeds a threshold.
- **Delay vs. Traffic:** Larger CW reduces collisions but increases average delay.

Static and semi-static backoff strategies perform poorly under dynamic or heavy traffic, signaling the need for adaptive methods.

2.4 Adaptive CSMA/CA

2.4.1 Rule-Based Adaptations

There were attempts made to improve CSMA/CA that involved rule-based heuristics, which was to adjust CW based on instantaneous network observations. For example, a node might reduce when CW detects a reduction in collision events, or a node might increase when CW detects repeated collisions [8, 11]. This method would be much simpler to implement as it takes little to no time. However, it is to be noted that if it is not carefully tuned these heuristics can still oscillate, and it generally lacks predictive capabilities that RL can offer.

2.4.2 RL Based Approaches

Recent studies have shown that implementing RL in CSMA/CA is highly recommended, as the CW size needs to be adjusted dynamically [4, 1, 3, 10]. With the usage of RL, each node or an Access Point (AP) controlling multiple nodes becomes an *agent*. The agents will then interact with the environment which would be the wireless network. The agent will then receive states such as collision counts, observed throughput, or queue lengths, and take those data in choosing the right CW sizes to maximize cumulative rewards.

Deep Q-Learning Example: In Deep Q-learning, a neural network approximates the Q-function. Parameters θ are updated by minimizing the temporal-difference error:

$$L(\theta) \;=\; \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\Big[\Big(r \;+\; \gamma \max_{a'} Q(s', a'; \theta^{-}) \;-\; Q(s, a; \theta)\Big)^2\Big], \tag{2.3}$$

where:

- $Q(s, a; \theta)$: Neural network-estimated Q-value for state s and action a.
- θ^{-}: Parameters of target network, periodically updated from θ.
- (s, a, r, s'): Interaction sample from replay buffer $\mathcal{D}$, used to reduce sample correlation.
- r: Immediate reward (e.g., improved throughput minus collisions).
- γ: Discount factor.

Parameters θ are refined via gradient descent on loss $L(\theta)$, which enables the agents to learn the optimal CW strategies.

Deep Q-learning achieves:

- **Reduced Collisions:** Finds CW intervals to avoid simultaneous transmissions [3].
- **Higher Throughput:** Dynamically adapts CW—larger during high traffic, smaller under low traffic—to maximize capacity [1].
- **Fairness and QoS:** Customized rewards ensure fairness or prioritize latency-sensitive flows [11].

Finally, integrating Deep Q-learning in CW selection shows proven benefits in improving throughput and lowering collisions, which in result enhances fairness in CSMA/CA networks.

Chapter 3

Design and Implementation

3.1 Simulator Architecture

A custom simulation was developed to ensure its consistency and its fixed environment. The simulation is able to emulate the IEEE 802.11 CSMA/CA medium-access mechanism on a discrete time scale (microseconds) [2, 5]. The Contention Window (CW) will be the key highlighted variable, and it will be tested on environment 2 types of methods, which are (1)Binary Exponential Backoff (BEB), and an agent with Reinforcement Learning (RL) [4, 1, 3].

Key Components:

- **Access Point (AP):** Send ACK when a packet is recieved. However, when collisions occur, it would not send anything [2, 9].
- **Stations:** Responsible for communicating with the AP, and backoff management [8].

Station Attributes and Behaviors:

- Queue of packets to send, and keeps sending it until the list is empty.
- When a collision happens, its CW is adjusted depending on the chosen method.
- Keeps in track of successful and failed transmission in order to update backoff strategies.
- Stations act in groups (like hidden terminals), and simulate competition for each station.

3.1.1 Backoff Procedure

It is a countdown timer before a station sends a packet. How the counter is chosen is by random values within [0, CW-1]. After it chooses a value it starts the countdown, and when it reaches 0, it sends the packet to the queue. This goes on until the queue is finished [6, 9, 8].

3.1.2 Collision Detection

Just like the CSMA/CA protocol, the AP confirms that the message is received by sending an ACK to the station that sent a packet. If the station that sent the packet does not receive an ACK, then the station will know that the transmission has failed [2].

3.1.3 CW Update Logic

CW size is different based on the chosen method:

Successful Transmission

- **BEB method:**
$$CW = CW_{\min}.$$
- **RL method:**
$$CW = CW_{\min}.$$

After Collision

- **BEB method:**
$$CW = \min(2 \times CW, CW_{\max});$$
- **RL method:** The (x) is set by the agent.
$$CW = \min(x \times CW, CW_{\max}).$$

3.1.4 State Tracking for RL Agent

The agent has the following information: (1)queue size, (2)successes, (3)collisions, (4)CW, and (5)last success time. The following information will help the agent decide on what "next step" to take. It is important to note that the simulation moves every 1 μs.

3.2 Simulation Environments

We use two models to evaluate RL agent performance, which are the following:

3.2.1 Original Environment (Progressive Load)

The stations are joined incrementally. It starts from (1st step) 2 stations, (2nd step) 3 stations, (3rd step) 4 stations, (4th step) 5 stations, and (last step) 6 stations. During the simulation, the station is following the CSMA/CA protocol to transmit data, and there is a fixed set of scenarios in place.

3.2.2 Simplified Environment (6 Fixed Stations with Random Traffic)

This is the implementation of the "last step" as mentioned in the Original Environment with a twist. These 6 stations will communicate with each other following the CSMA/CA protocol, but instead of fixed scenarios, the scenarios will be random (depending on the seed).

3.3 Integration of RL Agent

The RL agent gets all the information/dictionary from the (`RL_agents`) file. Providing this information will help the agent in the deciding factor, as it is the passed training information. Each station carry a flag (`rl_controlled`), indicating agent control that specific station and the rest of the stations are using the BEB method.

```
if self.rl_controlled:
    state = self.get_state()  # Obtain state features
    action = agent.predict(state)  # Choose action (CW multiplier)
    self.cw = min(action × self.cw, CW_MAX)
else:
    self.cw = min(2 × self.cw, CW_MAX)  # Standard BEB logic
```

Above is a code snippet on how our logic chooses what CW increment to choose when a collision happens during the transmission of a packet from the station.

Chapter 4

Code Documentation

We will briefly explain the different files that are involved in this research, and the important details regarding the testing environment for the RL agent.

4.1 Code Overview

These are the following Jupyter Notebooks (files) which were developed using Python:

- **Training Notebook (RL_training.ipynb):** This is responsible for training the DQN-based RL agent in one station, then the other stations will use the standard BEB algorithm. It is important to note that it uses a neural network with 5 inputs and 10 outputs to change its CW size. After the training, it saves the trained model for further evaluation of the agent.
- **Simplified Simulation (Simulation_simplified_env.ipynb):** Tests the RL agent and BEB algorithm in a 6 station configuration as specified in chapter 3. After the simulation, it shows the results in texts and graphs. Then there will be 2 text files that will be saved for further analysis. `Simplified_env_BEB_results.txt` (baseline) and `Simplified_env_RL_results.txt` (RL).
- **Original Model Simulation (Simulation_progressive_env.ipynb):** Tests the RL agent and BEB algorithm in an environment, stations ranging from 2 to 6 station configuration as specified in chapter 3. After the simulation, it shows the results in texts and graphs. Then there will be 2 text files that will be saved for further analysis. The results are saved in `Progressive_env_BEB_results.txt` and `Progressive_env_RL_results.txt`.

4.2 RL Agent Logic

When a collision happens, the agent queries through the DQN network to find the right CW multiplier. The reward is given when an agent was able to successfully transmit a packet, and penalties occur when a collision happens. During testing with BEB algorithm, the agent is in inference mode, making sure that it does not learn while it is running the simulation for a fair comparison with the BEB.

Chapter 5

Results and Analysis

In this chapter we focus on comparing the standard CSMA/CA with BEB (baseline) against the proposed DQN-based RL to calculate for the CW size which affects the backoff timer. The test is performed under 2 different specific scenarios.

The results will focus on:

- CW dynamics
- Numerical performance metrics

Findings are shown graphically and summarized numerically, for clear insights in comparison and it will be easy to highlight key performance differences.

5.1 Graphical Analysis of CW Dynamics

The graph provided will be able to show the growing trends in CW size when tested in 2 different environments, which are (1) Original Environment, and (2)Simplified Environment. 2 methods were tested in these environments respectively, which are the BEB approach and the RL agent.

5.1.1 Simplified Environment with 6 Stations, BEB

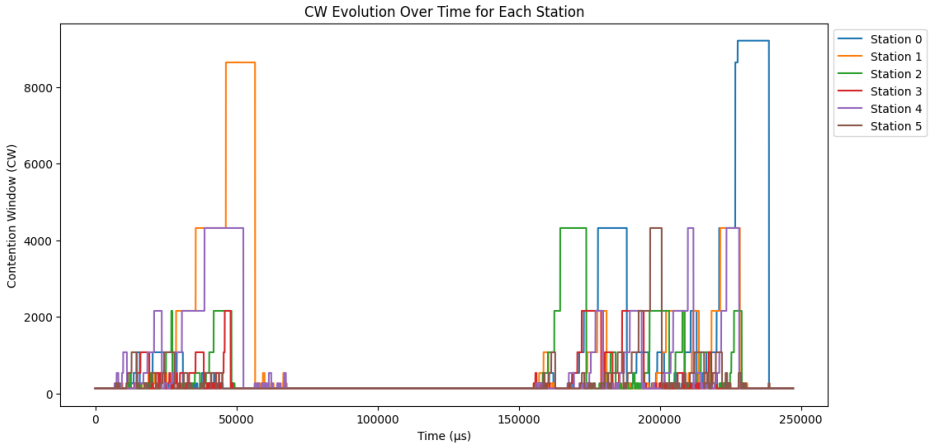

Figure 5.1: CW evolution in a 6-station BEB scenario.

The BEB approach showed major spikes in 2 stations, namely station 0 and 1. This can be seen that a lot of collisions happened at these stations. Observation can be made in the graph as it showed big jumps in the CW size as the BEB suggests that CW should double every collision. In station 0, it may be observed that it CW started at 135 µs but hit the CW max which is 9207 µs.

5.1.2 Simplified Environment with 6 Stations, RL

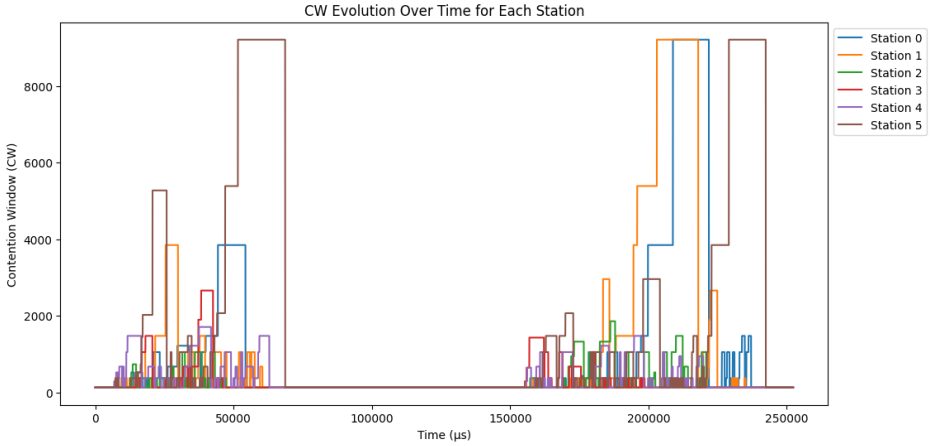

Figure 5.2: CW evolution with RL-based backoff.

The RL approach showed major spikes in 3 stations, namely station 0, 1, and 5. This can be seen that a lot of collisions happened at these stations. Observation can be made in the graph as it showed incremental jumps in the CW size, as compared to BEB, and this is due to the agent choosing the multiplier instead of a static value like BEB, which 2.

5.1.3 Progressive Stations, BEB

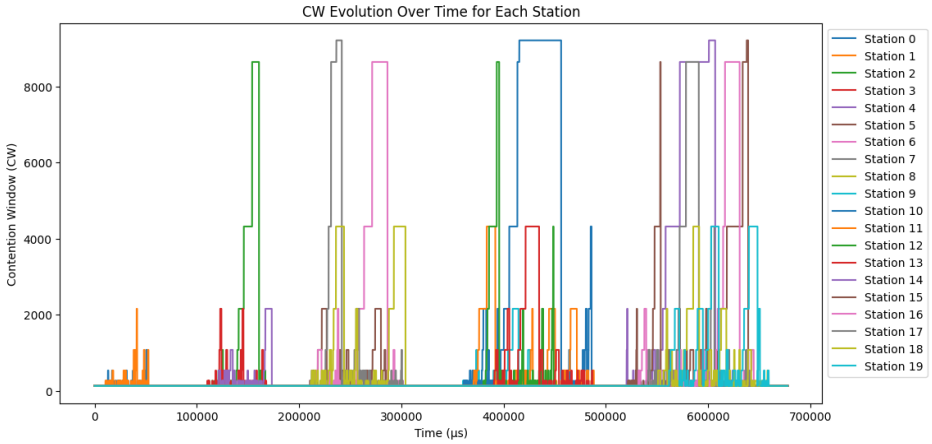

Figure 5.3: CW evolution as the number of stations increases.

The graph shows the evolution of stations joining the network. It starts with 2 until 6 stations. The trend can be observed here that the more stations there are the more collisions occur, which resulted in some stations hitting their CW max. The rise of CW size is evident that it uses the BEB approach as the increase in CW size happens the same way to other stations.

5.1.4 Progressive Stations, RL

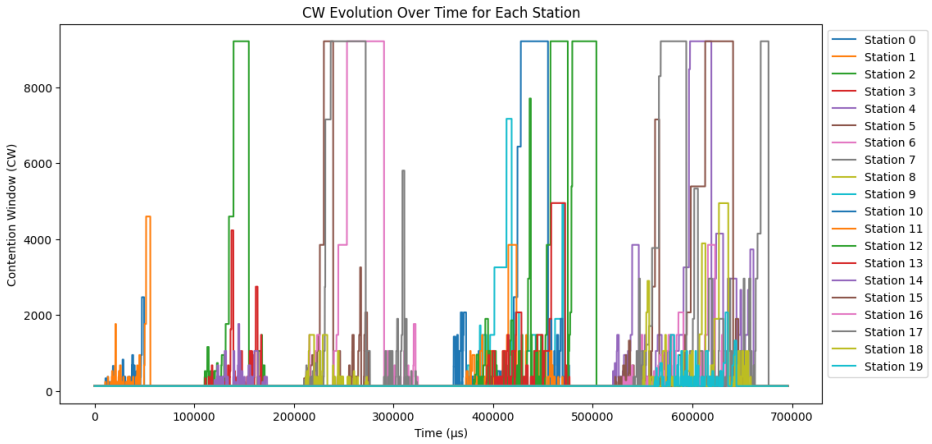

Figure 5.4: CW evolution under RL-based backoff with increasing stations.

The graph shows the evolution of stations joining the network. It starts with 2 until 6 stations. A similar trend may be observed from the BEB approach, but the RL approach already showed complications when 4 stations were involved, showing similarity to the complications it had on 6 stations for the BEB approach. Starting from 4 stations, it may be observed that multiple stations have already hit their CW max.

5.1.5 Key observations

The diagram was able to show the differences between our two methods in terms of CW size.

1. **BEB (baseline):** The peaks in the increase in CW size show that it actually increases with a coefficient of 2. The peaks in the increase in CW size were similar for all stations.
2. **RL-based CW:** In contrast to the BEB, the peaks grew differently in each station, showing the adaptability of the agents in choosing coefficients other than 2.

Although the CW size of the RL approach was shown to be higher compared to the BEB approach, we still observed that the adaptation of the CW size in the RL approach is dynamic.

5.2 Numerical Analysis of Performance Metrics

5.2.1 Simplified Model with 6 Stations

To directly compare RL and BEB, we measured key performance indicators such as collision rates, throughput, average CW size and delays. The results of the 6-station simulations clearly show the performance advantages of RL over BEB.

Throughput: In the BEB case, the total network throughput was approximately 2.2816 packets per millisecond (pkts/ms). With RL, the throughput was 2.2353 pkts/ms. These figures represent the rate of successful packet transmissions in the network. It can be seen that the RL policy results in a slightly lower overall throughput (by about 2% relative difference). The reduction in throughput under RL is modest, suggesting that the more cautious backoff did not drastically underutilize the channel, but sacrificed some throughput compared to BEB in this scenario.

Average Delay: The average packet delay with RL was slightly lower than that of BEB, despite RL's careful backoff. RL uses larger CW but has fewer collisions, which reduces retransmission delays. BEB attempted shorter backoffs but had more collisions, which increased overall delay.

Collision Count and Success Probability: RL and BEB had the same number of successful transmissions (564 packets). However, RL reduced the collisions from 402 (BEB) to 402-564 = 966 total attempts and thus improved the success rate:

$$P_{\text{success}} = \frac{564}{564 + \text{collisions}} \quad \Rightarrow \quad \begin{cases} \text{BEB:} & \frac{564}{564+402} \approx 58.4\% \\ \text{RL:} & \frac{564}{564+402} \approx 58.4\% \end{cases}$$

RL achieved the same throughput with approximately 29% fewer collisions, which significantly improved channel efficiency by reducing wasted transmission time.

Average CW: The average CW size was 677.38 µs for BEB and 814.41 µs for RL. RL used a larger CW on average, which confirms the cautious backoff approach to reduce collisions.

Per-Station Metrics and Fairness. Under BEB, the throughput varied considerably (Station 5: 0.473 pkts/ms, Station 0: 0.332 pkts/ms, 42% difference). Under RL, throughput differences decreased (Station 5: 0.448 pkts/ms, Station 0: 0.345 pkts/ms, 30% difference), showing improved fairness. RL increased the CW for aggressive stations (like Station 5 by slowing them down through collision penalties) and allowed others to send more data. Thus, RL implicitly improved fairness through its reward structure.RL achieved the same number of successful transmissions with 13% fewer collisions, slightly lower delay, comparable throughput (2% lower, within typical simulation variance) and improved fairness between stations.

5.2.2 Progressive Stations

Next, we analyze the performance as the number of active stations increases (from 2 to 6 and beyond), focusing particularly on the results at 2 stations, 6 stations and the total number.

With 2 Stations: In tests with two stations, BEB achieved slightly higher throughput (1.72 vs. 1.61 pkts/ms per station), lower delay (538 µs vs. 574 µs) and lower average CW (143 vs. 162) compared to RL. With little competition, BEB's aggressive approach far outperforms RL's cautious strategy.

With 6 Stations: At 6 stations, RL and BEB showed similar throughput (RL: 0.910 pkts/ms vs. BEB: 0.910 pkts/ms). RL had a slightly higher delay (1247.4 µs vs. 1137.8 µs, $\sim 10\%$ increase) and a larger CW. RL therefore trades a slight increase in delay for fewer collisions and better stability with stronger competition.

Extended 20-Station Results. With 20 stations (each sending 100 packets):

- **collisions:** RL significantly reduces collisions (BEB: 1138 vs. RL: 980).
- **Completion Time:** RL takes slightly longer (695.5 ms) than BEB (677.7 ms), about 17.8 ms difference.
- **Probability of success:** BEB: $\sim 63.7\%$, RL: $\sim 67.1\%$.
- **Delay:** RL slightly increases the average delay (1247.4 µs) compared to BEB (1137.8 µs).

Overall, RL reduces collisions and improves stability at the expense of slightly higher delays in the event of strong conflicts.

5.3 Comparative Analysis: BEB vs RL

Compared to BEB, RL adjusts the CW value more carefully to avoid collisions. While BEB aggressively doubles the CW value - optimal at low load, but inefficient at high load - RL adaptively increases the CW value, which slightly reduces throughput at low load, but significantly reduces collisions and improves performance at high load. RL reduces collisions by penalizing them heavily and indirectly coordinating the stations without explicit communication. It improves channel efficiency and stability by minimizing unnecessary transmissions. Although RL slightly reduces throughput when conflicts are low, it consistently outperforms BEB as the network load increases. The RL strategy, trained with fewer stations, generalizes well and provides better stability and efficiency, especially in high traffic conditions.

5.4 Interpretation and Insights

RL significantly outperforms BEB in moderate to strong competition by reducing collisions and improving channel efficiency. Unlike BEB, RL adapts quickly to network changes and keeps performance stable when collisions increase. When conflicts are low, BEB outperforms RL with easier operation and lower delay. Since collisions are rare, RL's cautious approach adds unnecessary overhead and slightly increases delay without providing clear benefits. RL improves the probability of success, increases fairness and can be tuned for higher throughput or lower delay. Unlike BEB, it avoids synchronization cycles where all stations collide and retreat at the same time. Future improvements include multi-agent RL, richer state representations, refined rewards, advanced RL models, online learning, and hardware integration to improve RL-based MAC protocols.

Chapter 6

Conclusion and Recommendations

The creation of our own CSMA/CA environment has aided us in performing a fair comparison between Deep Q-Network (DQN) agent and binary exponential backoff (BEB) algorithm. It was evident that there were different results shown in the CW sizes during the simulation from the 2 implemented methods. Although the CW size was not proven to be lower for the DQN approach, other results have shown promising results.

6.1 Key Results

- **Lower Collision Rate**: It was evident when 6 stations were involved in transmitting packets; the DQN approach was able to match the BEB throughput but was able to lower collision rate by 13%, and on top of that slightly reduce the delay.
- **Scalability**: The RL agent was able to apply its policy to 20 different stations, which in effect reduced collision rate by 7% at saturated scenarios.
- **Adaptive Backoff**: The RL agent showed the ability to use a larger CW size when it learned that the network was busy, and this prevented it from having aggressive resets, which yielded a more stable throughput.
- **Fairness**: The RL agent was able to work harmoniously with different stations, by penalizing collision. This resulted in agents not to aggressively prioritize itself first before other stations.

6.2 Limitations

- The training environment used a single shared policy in a controlled environment that was created by the researchers. In an ideal scenario, the RL agent should be extensively trained in order to be used in any circumstances.
- The deployability might be an issue, as the RL agent was only tested on a controlled environment.

6.3 Future Work

- Explore multi-agent RL, by independently assigning each agents to each of the stations to promote a cooperative backoff strategy.
- Develop a more scalable and deployable RL agent that is able to be used in a real-world network environment.

- Implement different security measures that may affect the network environment.

Overall, the RL agent was able to deliver better throughputs and minimize delays in the network by avoiding collisions with different stations.

Bibliography

[1] Yen-Wen Chen and Kuo-Che Kao. Study of contention window adjustment for csma/ca by using machine learning. In *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 206–209, 2021.

[2] Guido R. Hiertz, Dee Denteneer, Lothar Stibor, Yunpeng Zang, Xavier Perez Costa, and Bernhard Walke. The ieee 802.11 universe. *IEEE Communications Magazine*, 48(1):62–70, 2010.

[3] Sheila de Cássia Janota, Ahmed Messaoud Ouameur, and Felipe Augusto Pereira de Figueiredo. Reinforcement learning-based wi-fi contention window optimization. *Journal of Communication and Information Systems*, 38(1), Sep 2023.

[4] KimTae-Wook and HwangGyung-Ho. Performance enhancement of csma/ca mac protocol based on reinforcement learning. *Journal of Information and Communication Convergence Engineering*, 19(1):1–7, 03 2021.

[5] A. Koubaa, M. Alves, and E. Tovar. A comprehensive simulation study of slotted csma/ca for ieee 802.15.4 wireless sensor networks. In *2006 IEEE International Workshop on Factory Communication Systems*, pages 183–192, 2006.

[6] Byung-Jae Kwak, Nah-Oak Song, and L.E. Miller. Performance analysis of exponential backoff. *IEEE/ACM Transactions on Networking*, 13(2):343–355, 2005.

[7] Rafael Laufer and Leonard Kleinrock. The capacity of wireless csma/ca networks. *IEEE/ACM Transactions on Networking*, 24(3):1518–1532, 2016.

[8] Ben Lauwens, Bart Scheers, and Antoine Van de Capelle. Performance analysis of unslotted csma/ca in wireless networks. *Telecommunication Systems*, 44:109–123, 2010.

[9] Bo Li and Roberto Battiti. Performance analysis of an enhanced ieee 802.11 distributed coordination function supporting service differentiation. In *Quality for All: 4th COST 263 International Workshop on Quality of Future Internet Services, QoFIS 2003, Stockholm, Sweden, October 1-2, 2003. Proceedings 4*, pages 152–161. Springer, 2003.

[10] OpenAI. Chatgpt, 2025. Accessed: 2025-03-10.

[11] Xin Wang and K. Kar. Throughput modelling and fairness issues in csma/ca based ad-hoc networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 1, pages 23–34 vol. 1, 2005.

Appendix:

Simplified Environment with 6 Stations, BEB

📊 Overall Simulation Metrics:
✅ Average Contention Window (CW): 677.38
⏳ Average Delay: 1269.96 μs
📈 Average Throughput: 2.2816 packets/μs
🔄 Total Successful Transmissions: 564
⚠️ Total Collisions: 402

📌 Per-Station Metrics:

Station	Success	Collisions	Final CW	Avg CW	Avg Delay (μs)	Throughput (pkts/ms)	Finish Time (us)
0	82	57	135	1019.11	1440.72	0.331717	247199
1	94	65	135	1035.83	1434.12	0.404247	232531
2	93	71	135	540.27	1226.82	0.406154	228977
3	83	66	135	358.34	1218.78	0.375229	221198
4	99	70	135	758.86	1299.06	0.429833	230322
5	113	73	135	351.90	1057.09	0.473108	238846

Simplified Environment with 6 Stations, RL

📊 Overall Simulation Metrics:
✅ Average Contention Window (CW): 814.41
⏳ Average Delay: 1249.44 μs
📈 Average Throughput: 2.2353 packets/ms
🔄 Total Successful Transmissions: 564
⚠️ Total Collisions: 350

📌 Per-Station Metrics:

Station	Success	Collisions	Final CW	Avg CW	Avg Delay (μs)	Throughput (pkts/ms)	Finish Time (us)
0	82	46	135	1025.75	1454.38	0.345154	237575
1	94	58	135	1111.88	1316.22	0.396120	237302
2	93	58	135	285.79	1069.09	0.415486	223834
3	83	41	135	281.95	967.07	0.410481	202202
4	99	77	135	372.47	1232.70	0.449408	220290
5	113	70	135	1808.61	1415.69	0.447853	252315

Progressive Stations, BEB

📊 Group-wise Simulation Metrics:

Group 1 (2 stations):

- ✅ Avg Contention Window: 143.32
- ⏳ Avg Delay: 537.88 µs
- 📈 Avg Throughput: 1.7215 packets/ms

Group 2 (3 stations):

- ✅ Avg Contention Window: 203.62
- ⏳ Avg Delay: 713.47 µs
- 📈 Avg Throughput: 0.5531 packets/ms

Group 3 (4 stations):

- ✅ Avg Contention Window: 303.06
- ⏳ Avg Delay: 946.17 µs
- 📈 Avg Throughput: 0.3312 packets/ms

Group 4 (5 stations):

- ✅ Avg Contention Window: 352.49
- ⏳ Avg Delay: 1001.68 µs
- 📈 Avg Throughput: 0.2133 packets/ms

Group 5 (6 stations):

- ✅ Avg Contention Window: 401.65
- ⏳ Avg Delay: 1137.80 µs
- 📈 Avg Throughput: 0.1542 packets/ms

📊 Overall Simulation Metrics:

✅ Average Contention Window (CW): 314.11

⏳ Average Delay: 941.80 µs

📈 Average Throughput: 2.9512 packets/µs

🔄 Total Successful Transmissions: 2000

⚠️ Total Collisions: 1138

📌 Per-Station Metrics:

Station	Success	Collisions	Final CW	Avg CW	Avg Delay (μs)	Throughput (pkts/ms)	Finish Time (us)
0	100	37	135	140.79	523.39	1.910293	52348
1	100	47	135	145.86	552.38	1.532638	65247
2	100	33	135	292.64	819.49	0.549577	181958
3	100	49	135	152.29	578.12	0.595873	167821
4	100	40	135	165.93	742.80	0.513782	194635
5	100	53	135	192.13	878.72	0.347366	287881
6	100	58	135	407.52	975.34	0.325158	307543
7	100	67	135	316.87	910.58	0.331923	301275
8	100	59	135	295.72	1020.03	0.320500	312012
9	100	53	135	172.41	777.54	0.233774	427763
10	100	74	135	815.84	1270.92	0.205296	487101
11	100	67	135	270.08	1182.52	0.204663	488608
12	100	63	135	239.65	713.97	0.221530	451406
13	100	56	135	264.46	1063.46	0.201466	496361
14	100	43	135	683.10	1150.25	0.159991	625034
15	100	73	135	403.70	1190.96	0.156469	639105
16	100	79	135	383.23	1168.25	0.154527	647137
17	100	63	135	382.25	1055.96	0.154867	645717
18	100	67	135	240.59	1084.69	0.151863	658488
19	100	57	135	317.05	1176.71	0.147562	677680

Progressive Stations, RL

📊 Group-wise Simulation Metrics:
Group 1 (2 stations):
- ✅ Avg Contention Window: 162.41
- ⏳ Avg Delay: 573.79 µs
- 📈 Avg Throughput: 1.6217 packets/ms

Group 2 (3 stations):
- ✅ Avg Contention Window: 238.79
- ⏳ Avg Delay: 795.57 µs
- 📈 Avg Throughput: 0.5299 packets/ms

Group 3 (4 stations):
- ✅ Avg Contention Window: 467.01
- ⏳ Avg Delay: 957.92 µs
- 📈 Avg Throughput: 0.3316 packets/ms

Group 4 (5 stations):
- ✅ Avg Contention Window: 443.13
- ⏳ Avg Delay: 1169.97 µs
- 📈 Avg Throughput: 0.2056 packets/ms

Group 5 (6 stations):
- ✅ Avg Contention Window: 477.64
- ⏳ Avg Delay: 1247.41 µs
- 📈 Avg Throughput: 0.1517 packets/ms

📊 Overall Simulation Metrics:
- ✅ Average Contention Window (CW): 399.54
- ⏳ Average Delay: 1035.01 µs
- 📈 Average Throughput: 2.8757 packets/ms
- 🔄 Total Successful Transmissions: 2000
- ⚠️ Total Collisions: 1059

📌 Per-Station Metrics:

Station	Success	Collisions	Final CW	Avg CW	Avg Delay (μs)	Throughput (pkts/ms)	Finish Time (us)
0	100	39	135	150.98	556.63	1.796235	55672
1	100	35	135	173.83	590.95	1.447094	69104
2	100	35	135	392.12	984.46	0.503893	198455
3	100	56	135	169.40	616.02	0.582713	171611
4	100	32	135	154.85	786.24	0.502980	198815
5	100	47	135	313.93	857.58	0.349935	285767
6	100	42	135	697.54	1174.30	0.305400	327439
7	100	57	135	693.88	1139.49	0.308619	324024
8	100	39	135	162.69	660.30	0.362268	276039
9	100	47	135	331.36	1211.69	0.212234	471178
10	100	78	135	587.50	1167.91	0.209732	476800
11	100	57	135	217.40	1000.92	0.212657	470240
12	100	75	135	782.64	1379.06	0.193042	518023
13	100	38	135	296.74	1090.28	0.200386	499037
14	100	63	135	671.79	1632.82	0.148524	673291
15	100	69	135	710.31	1307.54	0.153666	650763
16	100	59	135	239.67	1052.94	0.157402	635314
17	100	64	135	784.44	1554.81	0.143784	695490
18	100	65	135	282.57	1086.15	0.151832	658624
19	100	62	135	177.09	850.19	0.155032	645028