O-BEB: Optimised Binary Exponential Backoff Algorithm for CSMA/CA Protocol

Devaansh Kumar, Lohith C V, Shyam Balaji
Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore, India
8050093839, 8050778187, 9108571214
devaansh.211cs119@nitk.edu.in, lohithcv.211cs136@nitk.edu.in, shyambalaji.211cs154@nitk.edu.in

May 28, 2023

Abstract

A. Background of the problem statement

As a computer network protocol, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) [1] is used to govern the access of devices to the communication medium in a wireless network. In a scenario, where the number of nodes changes frequently [2], CSMA/CA is particularly useful because it allows devices to adapt to the changing network conditions.

In a changing or evolving network structure, nodes may enter or leave the network. In response to these changes, CSMA/CA uses a technique called "backoff" to adjust the waiting period before attempting to transmit. This helps to ensure that nodes can access the communication medium in a fair and efficient manner, even as the network topology changes over time.

B. Challenges and issues of the problem statement

Hidden and Exposed Node Problems: In ever-changing networks devices can move around, which can create situations where some devices are hidden from each other or are too far apart to sense each other's transmissions. This can cause collisions and degraded performance, particularly if some devices are closer to the access point than others. [3]

Unpredictable Channel Conditions: In constantly changing networks the quality of the wireless channel can vary greatly over time. For example, devices may experience interference from other wireless networks or environmental factors such as walls or other obstacles. This can make it difficult to determine the appropriate transmission power or channel access parameters needed to maintain good performance. [4]

Scalability: In ever-changing networks the number of devices can change rapidly, which can make it difficult to manage the network and ensure that all devices can access the channel fairly. As the number of devices increases, the probability of collisions also increases. [5]

C. Existing approaches or methods and their issues

Binary Exponential Backoff (BEB) algorithm: This algorithm dynamically adjusts contention window size. Before transmitting data, a device senses the medium to check for activity. If the medium is idle, it transmits, but if another device transmits simultaneously, a collision occurs, and both devices wait a random time before retrying. [6, 7]

Virtual Carrier Sensing (VCS) algorithm: This algorithm overcomes hidden terminal problems. The device sends an RTS packet to the receiver before transmitting data. The receiver replies with a CTS packet, indicating the medium will be busy during transmission. Devices receiving CTS packets defer transmission. Devices receiving RTS packets defer transmission until receiving CTS packets or transmission times out. [8]

CSMA/CA with Backoff-Free Reservation: Nodes reserve the channel for their transmissions without backoff, reducing collisions and improving throughput in small networks. [9]

CSMA/CA with Energy Detection: Nodes use energy detection [10] to sense the channel and determine whether it is busy or idle, reducing collisions and improving energy efficiency. [11]

D. Your problem statement

Our project seeks to optimize the CSMA/CA protocol by adjusting the backoff period [5] and throughput [12,13] to adapt to changing network conditions. By doing so, this paper hopes to improve the protocol's efficiency and effectiveness in networks where number of nodes is not constant. This can help to reduce collisions and improve network throughput [14] and lead to better network performance and improved user experience.

E. Objectives of the proposed work

1. Increase throughput of CSMA/CA in networks where the number of nodes is not fixed. [12,13]
2. Optimizing the backoff period to minimize waiting time and reduce collisions. [5,15]

1 Introduction

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) [1] is a crucial computer network protocol utilized in wireless networks to regulate the access of devices to the communication medium. It is especially advantageous in a changing or evolving network structure [2] where the number of nodes changes frequently. In such scenarios, devices can adapt to the changing network conditions by employing the backoff technique, which adjusts the waiting period before attempting to transmit.

This technique becomes significant because changes in the network topology can cause nodes to experience long waiting periods or collisions, resulting in a reduction in network performance. By utilizing backoff, CSMA/CA can dynamically adjust to these changes in the network topology, ensuring that nodes have an equitable chance of accessing the medium and maintaining a high level of network performance.

Moreover, CSMA/CA protocol can operate in both infrastructure and ad-hoc network modes, making it a versatile and flexible solution for different types of wireless networks. It is also widely used in various wireless communication technologies such as WiFi and Bluetooth, demonstrating its compatibility and applicability across different platforms.

Therefore, CSMA/CA serves as an efficient and effective protocol that facilitates the smooth functioning of wireless networks in these scenarios.

1.1 Challenges and issues of the problem statement

Hidden and Exposed Node Problems: The hidden and exposed node problems are two common issues that arise in wireless communication networks when multiple nodes try to access the communication medium simultaneously. The hidden node problem occurs when two or more nodes are located far apart from each other, and each node cannot detect the other's transmission due to distance or physical obstructions. As a result, these nodes may transmit data at the same time, leading to a collision in the medium. On the other hand, the exposed node problem occurs when a transmitting node restricts its transmission due to the perceived presence of another node that is already transmitting, even though the other node is not within the range of the receiving node. Both of these problems can have a detrimental effect on the network's overall performance and efficiency. [3]

Unpredictable Channel Conditions: The quality of the wireless channel can be affected by various factors such as distance, interference from other wireless networks, environmental factors, and signal attenuation. As a result, the signal strength and reliability of wireless transmissions can vary greatly over time [4].In a network where structures constantly change or evolve structures, a variability in channel quality can make it challenging to determine the appropriate transmission power or channel access parameters needed to maintain good performance. For example, if the transmission power is set too high, it can lead to interference with other nearby networks, while a low transmission power can result in a weak signal that is easily disrupted. Similarly, if the channel access parameters are not appropriately adjusted, nodes may experience longer waiting periods, resulting in lower network performance.

Scalability: As the number of devices increases, the probability of collisions also increases, leading to a decrease in network performance. This is because more devices competing for the same channel can lead to congestion and higher contention, resulting in longer waiting periods and a higher likelihood of collisions. [5]

1.2 Existing approaches or methods and their issues

Binary Exponential Backoff (BEB) algorithm: The contention window size is adjusted constantly by this algorithm. A device detects the medium to look for activity before transmitting data. When the medium is empty, it transmits; however, if another device transmits at the same time, there is a collision, and both devices must wait a random amount of time before attempting again. [6, 7]

Virtual Carrier Sensing (VCS) algorithm: This algorithm overcomes hidden terminal problems. Before sending data, the device transmits an RTS packet to the receiver. The receiver responds with a CTS packet, signaling that the transmission medium will be busy. Receiving CTS packets delay transfer on their end. When getting RTS packets, devices wait to transmit until they receive CTS packets or until the transmission timer expires. [8]

CSMA/CA with Backoff-Free Reservation: Nodes reserve the channel for their transmissions without backoff, reducing collisions and improving throughput in small networks. [9]

CSMA/CA with Energy Detection: Nodes use energy detection [10] to sense the channel and determine whether it is busy or idle, reducing collisions and improving energy efficiency. [11]

I-BEB: The I-BEB algorithm, a modified version of the Binary Exponential Backoff (BEB) algorithm, is employed in the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocols. It ensures that waiting time for a particular node does not increase too rapidly while also allowing efficient use of the network.

E-BEB: The E-BEB algorithm is a modified version of the Binary Exponential Backoff (BEB) algorithm, which is utilized in network communications to avoid collisions that may occur when multiple nodes attempt to access the medium simultaneously. The E-BEB algorithm adjusts the backoff time threshold and contention window (CW) size based on the number of nodes accessing the medium.

1.3 Problem Statement

Our project seeks to optimize the CSMA/CA protocol by adjusting the backoff period [5] and throughput [12,13] to adapt to changing network conditions. By doing so, this paper hopes to improve the protocol's efficiency and effectiveness in changing or evolving network structures. This can help to reduce collisions and improve network throughput in a network with a variable number of nodes [14] and lead to better network performance and improved user experience even with large number of nodes.

1.4 Objectives of the proposed work

1. Increase throughput of CSMA/CA in changing or evolving network structures. [12, 13]
2. Optimizing the backoff period to minimize waiting time and reduce collisions. [5, 15]

This paper has been organized further as follows: Section 2 contains the Literature Review which includes Existing approaches or methods and their issues (Section 2.1) and then moves on to the comparison of the Existing approaches or methods(Section 2.2). After this, the paper presents our Proposed Methodology(Section 3) where our proposed idea is demonstrated to improve the Backoff algorithm for CSMA/CA protocol along with a Justification (Section 3.1) portion. Section 4 gives description of the Algorithm used followed by Results in Section 5. The paper concludes in Section 6 and outlines areas of future work in Section 7.

2 Literature Review

2.1 Existing approaches or methods and their issues

2.1.1 BEB

Binary Exponential Backoff (BEB) algorithm is widely used in distributed MAC protocol. The BEB simply exponentially doubles the contention window (CW) value whenever collisions occur. This in done so as to avoid repeated collisions and it always resets the CW value to be the minimum CW after a successful transmission. However, this paper assumes that there is a very low-level of network congestion. Hence in highly congested and dense network environments, the BEB scheme in the current CSMA/CA does not provide optimized throughput.

The IEEE 802.11 MAC and Distributed Coordination Function (DCF) procedure can be utilized to manage a large number of access nodes. This involves updating the common medium access backoff interval and setting the shortest and distributed inter-frame spaces(SIFS and DIFS) using the backoff slot time. Whenever a node senses that the medium is busy, it selects a random value within a specified contention window for the backoff time. [16]

The purpose of introducing a waiting time in accessing the medium is to ensure smooth and collision-free communication. The BEB algorithm waits for the wireless channel to become available before initiating a countdown for the waiting time. However, if the channel has remained idle for longer than the DIFS time, the node can proceed to transmit its packet without further delay. When a collision is unavoidable, the node chooses a random waiting time, known as the Backoff time. Although nodes may use the same slot time within the contention window, each node has its own unique Backoff time. After waiting for the Backoff period, the nodes start monitoring the medium. If a node detects that the channel is busy, it stops its transmission attempt and selects a new Backoff time for the next cycle. The nodes choose their Backoff time from a random range between 0 and the size of the contention window. The Backoff time is calculated using the following formula:

$$\text{Backoff Time} = \text{Random } [0, \text{CW}] \times \text{SlotTime } (1)$$

If a node transmission fails because of a collision or deferring, the node doubles its CW size exponentially. The new CW size was recalculated using the following formula:

$$CW_{new} = min(2 \times CW_{old}, CW_{max})(2)$$

The node resets its CW size after every successful or unsuccessful transmission using the following formula:

$$CW = CW_{min}(3)$$

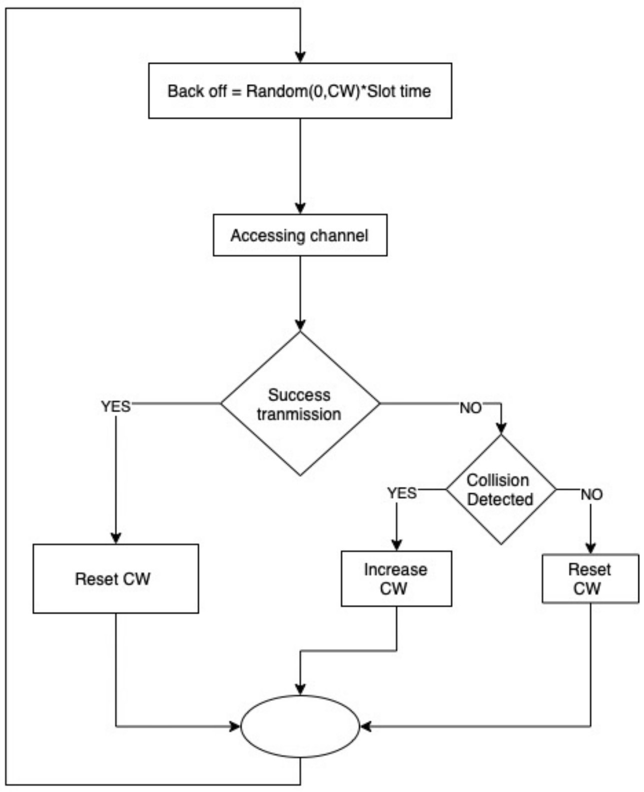


Figure 1: Flowchart for BEB

2.1.2 I-BEB

The I-BEB algorithm, which stands for Improved Binary Exponential Backoff, is a variant of the basic Binary Exponential Backoff (BEB) algorithm used in the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocols.

The I-BEB algorithm is an upgrade to the normal BEB algorithm by incorporating an additional factor into the waiting time calculation. This factor is based on the number of successful transmissions. If this number is less than a certain value let's say "success limit" then the contention window size keeps getting divided by a value let us say "x". It also has a CW$_{min}$ so if CW ever goes below the limit it gets assigned CW$_{min}$. The lower limit is (0.2 * CW$_{min}$). In case the number of successful transmissions crosses the "success limit" then the CW is increased by E*CW$_{min}$ where E is some constant.

By incorporating this additional factor, the I-BEB algorithm ensures that the waiting time does not increase too rapidly while still allowing for efficient use of the network. This leads to better network performance and reduced transmission delays. [16, 17]

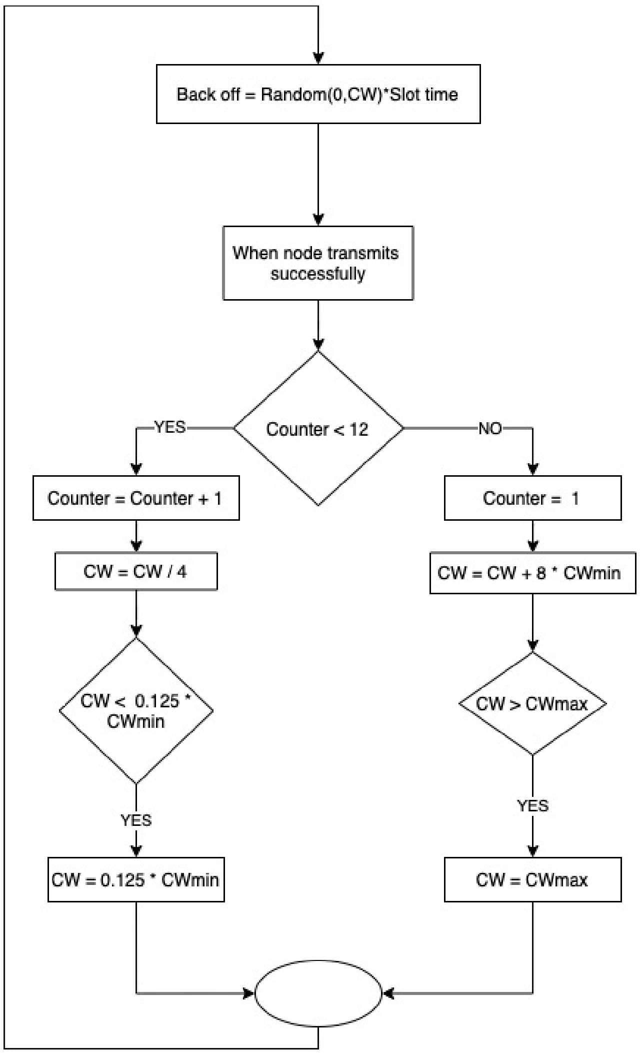


Figure 2: Flowchart for I-BEB

2.1.3 E-BEB

The E-BEB algorithm is a variant of the Binary Exponential Backoff (BEB) algorithm that is used in network communications to prevent collisions when multiple nodes try to access the medium at the same time. The E-BEB algorithm modifies the backoff time threshold (BT_Th) and CW size based on the number of nodes using the medium.

The BT_Th determines the maximum backoff time that a node should use before attempting to retransmit a packet. The relationship between BT_Th and CW size is crucial in preventing collisions. If many nodes are using the medium, a small BT_Th is required in comparison to the current CW size to avoid long backoff times and ensure timely collision resolution. Conversely, a higher BT_Th is necessary when there are only a few nodes using the medium to achieve quick collision resolution. Let us say "slot time" is the time required to transmit a packet of minimum size.

The formula used to calculate the BT_Th value is as follows:

$$BT\_Th = (CW_{min}/(\sqrt{CW_{max}}) \times CW \times (slot\ time)$$

Here, CW_{min} and CW_{max} are the minimum and maximum contention window sizes, respectively.

The E-BEB algorithm also includes a unique counter for each node that keeps track of its highest successful transmission. After a successful transmission, the node starts monitoring its own counter. The algorithm modifies the CW size while taking the node's BT_Th value into account.

The E-BEB algorithm in essence modifies the BT_Th and CW size based on the number of nodes using the medium to prevent collisions and ensure timely transmission of packets. [17]

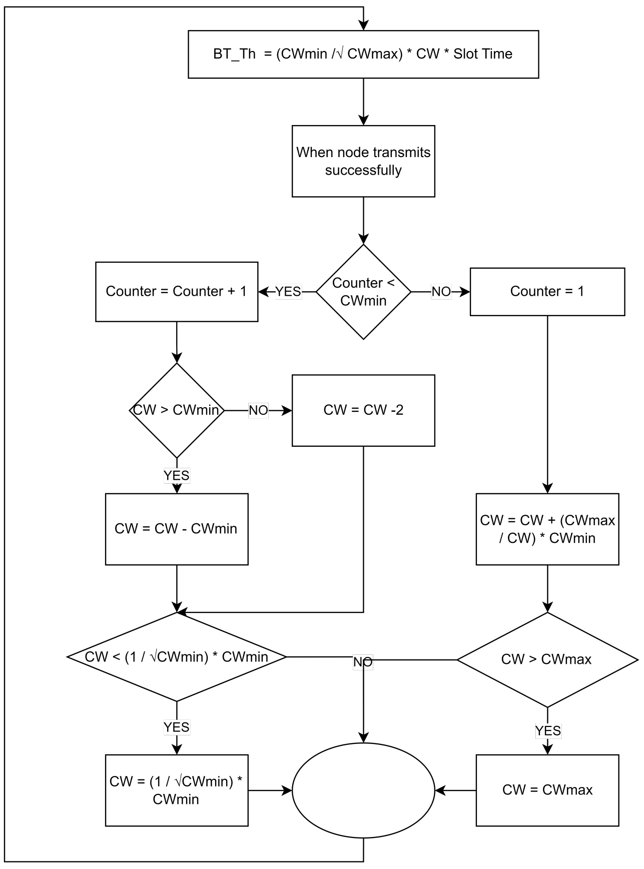

BT_Th = (CWmin /√ CWmax) * CW * Slot Time
When node transmits successfully
Counter < CWmin
YES
NO
Counter = Counter + 1
Counter = 1
CW > CWmin
NO
CW = CW -2
YES
CW = CW - CWmin
CW = CW + (CWmax / CW) * CWmin
CW < (1 / √CWmin) * CWmin
NO
CW > CWmax
YES
YES
CW = (1 / √CWmin) * CWmin
CW = CWmax

Figure 3: Flowchart for E-BEB

2.1.4 Virtual Carrier Sensing (VCS)

Using CSMA/CA, virtual channel sense is a technique used in wireless networks to predict future traffic and minimize packet collisions, which can lead to decreased network efficiency and data loss.

The traditional method of carrier sensing in wireless networks involves a device monitoring the channel for any activity before sending data. However, this technique has certain constraints, particularly in congested networks where numerous devices attempt to transmit simultaneously. In such scenarios, the channel may seem vacant to one device, while another device is already transmitting, resulting in packet collisions.

The Virtual channel sensing predicts upcoming channel traffic by utilizing a timer mechanism that is based on information from the previous frame transmission duration. It utilizes a network allocation vector (NAV), which can be thought of as a clock that gradually decreases its value until it reaches zero.

So with virtual carrier sensing, devices can "sense" the channel without actually listening to it. This is an extremely useful approach and is used in wireless networks that use shared medium, for example Wi-Fi, where various devices are competing for access to the same channel. [18]

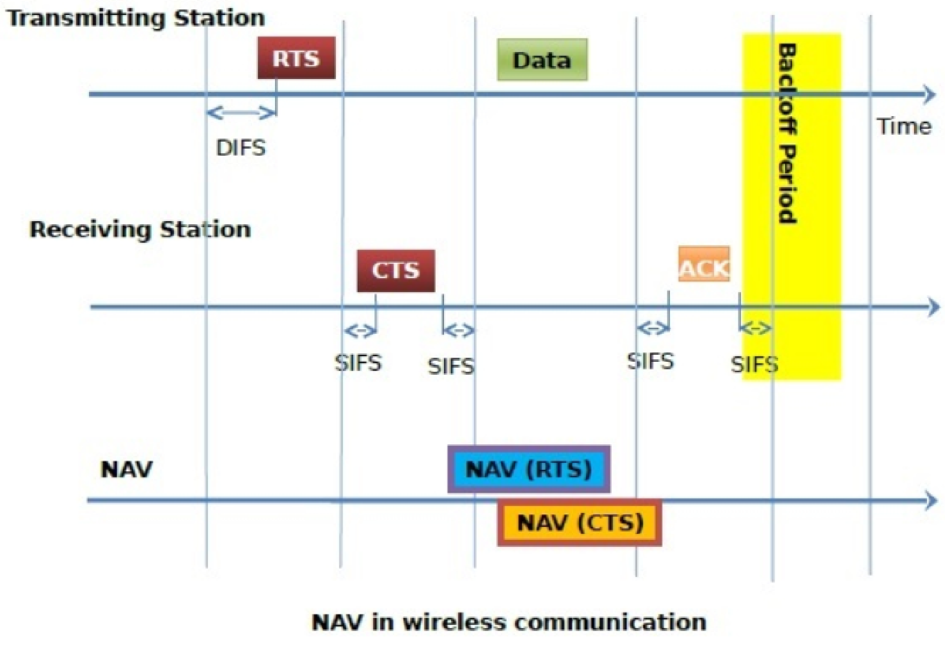


Figure 4: Virtual Carrier Sensing

The transmission stages in VCS are as follows:

1. If the channel is clear, the transmitting station sends a request to send (RTS) after waiting for a period of time equal to Distributed Inter-Frame Space (DIFS).
2. A Network Allocation Vector(NAV) is initialized after sending RTS, preventing any further transmission efforts from other stations.
3. The receiving station gives a Clear To Send (CTS) response after waiting for a Short Inter-Frame Space (SIFS).
4. A NAV is initialized with the CTS. After waiting for a SIFS, the originator sends its data frame.
5. After getting the data frame, the receiver sends an Acknowledgement frame (ACK) and waits for a SIFS.
6. During this time, both NAV numbers decrease to zero.
7. The stations wait for a SIFS and may or may not wait for a backoff period before contending for the channel depending upon implementation.

2.2 Comparison of the Existing approaches or methods:

Below are few graphs showing performance of the above existing approaches mentioned above.

1. This graph shows the throughput comparisons [17] of the different approaches of BEB mentioned above.

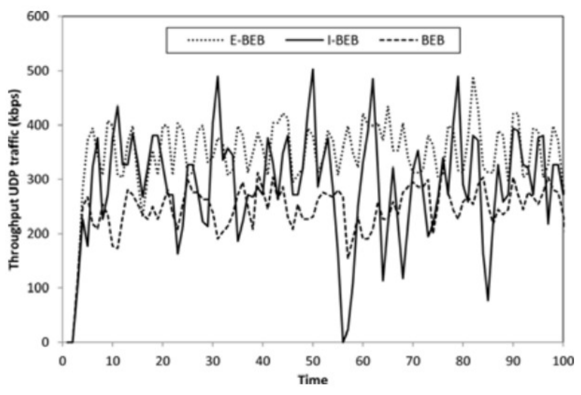


Figure 5: Comparison of BEB, I-BEB and E-BEB in terms of Throughput vs time

The above graph indicates that I-BEB performs better compared to the other.

2. This graph shows the fairness comparisons [17] of the different approaches of BEB mentioned above.

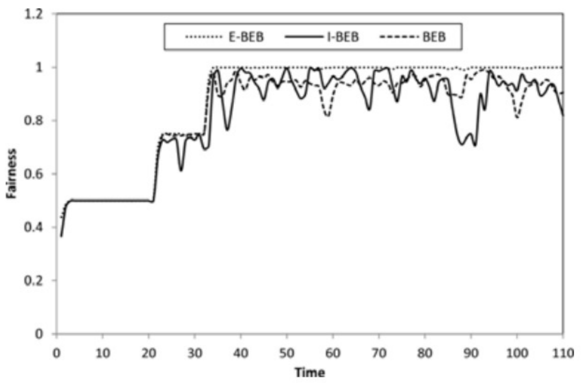


Figure 6: Comparison of BEB, I-BEB and E-BEB in terms of Fairness vs time

The above graph indicates that E-BEB relatively having flatter graph indicates it performs better compared to the other.

3 Proposed Methodology

This paper proposes a new state-of-the-art algorithm to strengthen the original BEB. Our algorithm enhances the opportunity for any given node intending to send packets to access the channel. Our aim is to change the values of Contention Window size depending upon whether there was a successful transmission or unsuccessful transmission. The value proposed below are theoretical for now and will be verified/changed based on simulation results. For instance, if a node cannot access the channel, our proposed algorithm permits the CW to increase its value by $\sqrt{2}$ continuously. To get the best results, these increments are dependent on certain parameters that will be calculated through extensive simulation tests. After two unsuccessful transmissions, the node starts decreasing the value of the CW.

When a transmission is successful as shown in 7, the CW is calculated by:

$$CW = CW/\sqrt{2}$$

This is done for the first x successful transmissions. Once this happens the number of successful transmissions is reset to 1 and CW is now set to CW×y. Our aim is to try and find the optimal x and y values to reduce delay time and increase throughput.

We have researched into why to actually increase CW size when a certain number of successful transmissions is achieved, and we have found that doing this enables nodes whose data were previously unable to be transmitted due to collisions can now do so. This step introduces efficient fairness between the nodes. In case of an unsuccessful transmission as explained in 7, the CW is calculated as:

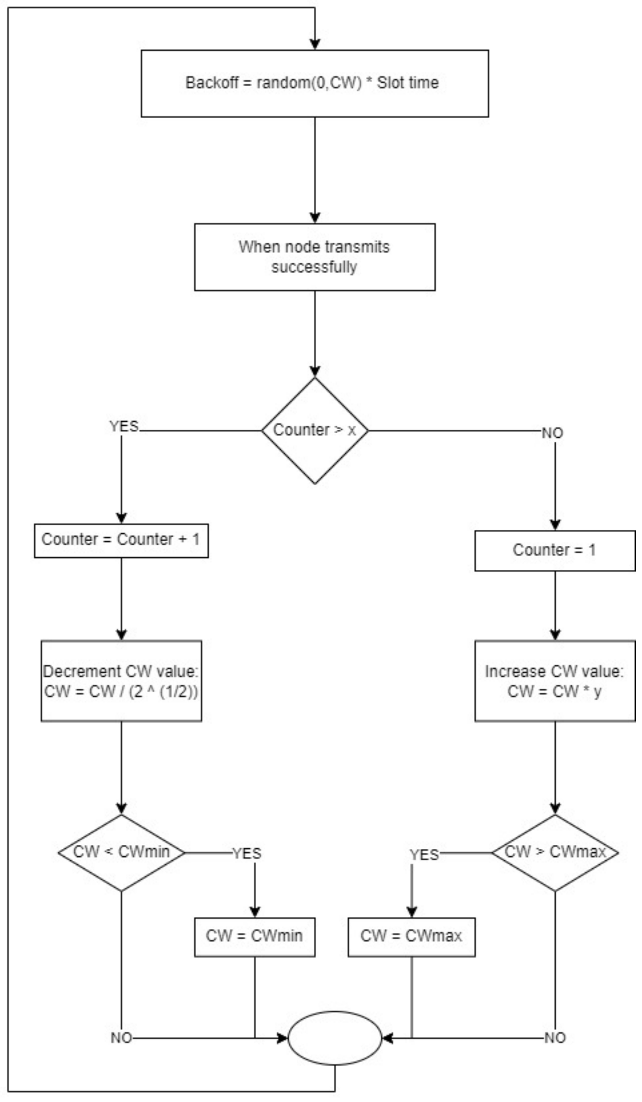
$$CW = CW \times \sqrt{2}$$

This operation will be repeated during the first two unsuccessful transmissions.

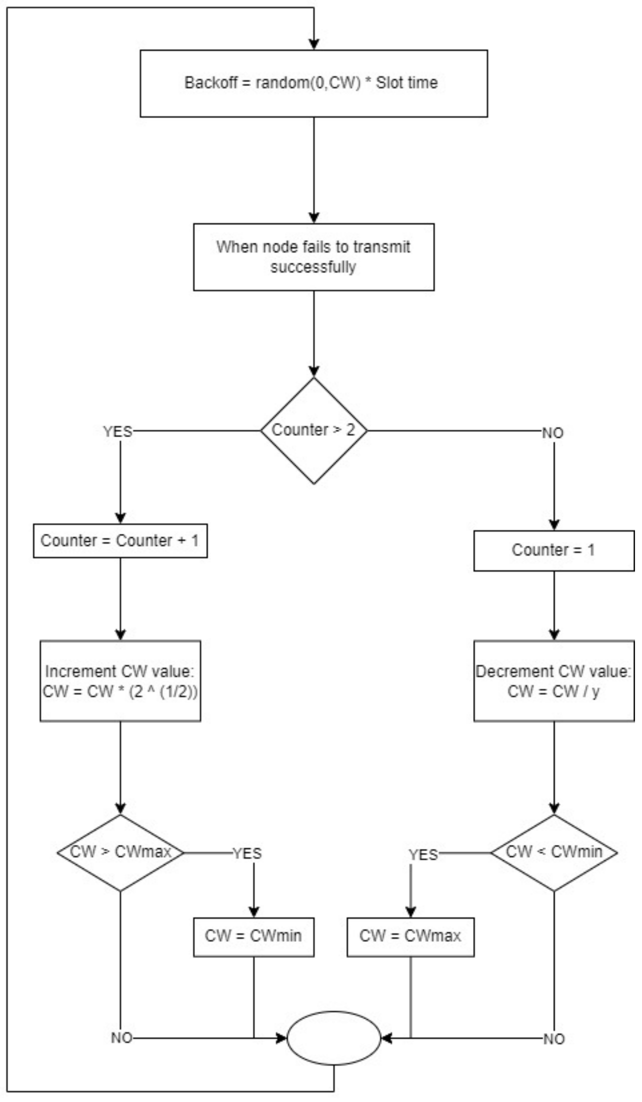
3.1 Justification

The proposed algorithm builds on this approach by dynamically adjusting the CW size based on the success or failure of transmissions. When a transmission is successful, the CW size is decreased, which allows for faster transmission times and reduces the overall delay. Conversely, when a transmission is unsuccessful, the CW size is increased, which reduces the likelihood of further collisions and increases the probability of successful transmissions.

The proposed algorithm also introduces a fairness mechanism by allowing nodes that were previously unable to transmit due to collisions to access the channel. This is achieved by increasing the CW size after a certain


Backoff = random(0,CW) * Slot time
When node transmits successfully
Counter > x
YES
NO
Counter = Counter + 1
Counter = 1
Decrement CW value: CW = CW / (2 ^ (1/2))
Increase CW value: CW = CW * y
CW < CWmin
YES
CW > CWmax
YES
CW = CWmin
CW = CWmax
NO
NO

(a)


Backoff = random(0,CW) * Slot time
When node fails to transmit successfully
Counter > 2
YES
NO
Counter = Counter + 1
Counter = 1
Increment CW value: CW = CW * (2 ^ (1/2))
Decrement CW value: CW = CW / y
CW > CWmax
YES
CW < CWmin
YES
CW = CWmin
CW = CWmax
NO
NO

(b)

Figure 7: (a) Flow for successful transmission in proposed algorithm (b) Flow for unsuccessful transmission in proposed algorithm

number of successful transmissions, which allows previously blocked nodes to have a chance to transmit their data.

The effectiveness of the proposed algorithm is dependent on the values of the parameters x and y, which need to be determined through extensive simulation tests. The algorithm's performance can be evaluated by measuring the delay time and throughput and comparing them with the original BEB algorithm.

Overall, the proposed algorithm has the potential to improve network performance by reducing delay times, increasing throughput, and introducing fairness between nodes. However, further research and testing are required to determine the optimal values of the parameters and to evaluate the algorithm's performance in different network conditions.

4 Implementation

In this section, we present the detailed implementation of our proposed algorithm which is called O-BEB (Optimized Binary Exponential Backoff), designed to enhance channel access opportunities for nodes transmitting packets. The algorithm dynamically adjusts the values of the Contention Window (CW) based on whether the previous transmission was a success or failure, with the goal of increasing throughput.

To evaluate the performance of the O-BEB algorithm and compare it with the above-mentioned algorithms, we have used Python. The implementation captures the key steps outlined above and incorporates necessary data structures and random number generation for backoff time calculation. The Python implementation provides a foundation for conducting simulations and analyzing the algorithm's behavior in various network scenarios. The algorithm was simulated for varying numbers of nodes and time slots to give a good picture of how it performs under different scenarios.

4.1 Algorithm Overview

Algorithm 1 O-BEB

Input: number of nodes, number of time slots

Output: a list containing throughput for each time slot,the total number of transmissions, number of successful transmissions

```
$throughput \leftarrow$ empty list;
$totalTransmissions \leftarrow 0$;
$successfullTransmissions \leftarrow 0$;

foreach $timeSlot$ do
    $transmittingNodes \leftarrow$ nodes trying to transmit;
    if $length(transmittingNodes) \geq 2$ then
        $totalTransmissions \leftarrow$ totalTransmissions + length(transmittinNodes);
        foreach $node \leftarrow transmittingNodes$ do
            $O - BEB - Failure(node)$;
            $performBackoff(node)$;  // backoff is randomly chosen from 0 to (CW-1)
        end
    else
        $successfullTransmissions \leftarrow$ successfullTransmissions + 1;
        $O - BEB - Success(transmittingNodes[0])$;
    end
    throughput.$append$(successfullTransmissions / (currentTimeSlot));
end
return $throughput, successfullTransmissions, totalTransmissions$ ;
```

Algorithm 2 O-BEB-Success

Input: Node

```
if $counterSuccess \leq 14$ then
    Node.counterSuccess = Node.counterSuccess + 1;
    Node.CW = Node.CW / $\sqrt{2}$;
    if $Node.CW \leq Node.CW_{min}$ then
        $Node.CW \leftarrow Node.CW_{min}$;
    end
else
    $Node.counterSuccess \leftarrow 1$;
    $Node.CW \leftarrow Node.CW * \sqrt{2}$;
    if $Node.CW \geq Node.CW_{max}$ then
        $CW \leftarrow CW_{max}$;
    end
end
```

Algorithm 3 O-BEB-Failure

```
Input: Node
if $counterFailure \leq 14$ then
    Node.counterFailure = Node.counterFailure + 1;
    Node.CW = Node.CW * 10;
    if $Node.CW \geq Node.CW_{max}$ then
        $CW \leftarrow CW_{max}$;
    end
else
    $Node.counterFailure \leftarrow 1$;
    $Node.CW \leftarrow Node.CW/\sqrt{2}$;
    if $Node.CW \leq Node.CW_{min}$ then
        $CW \leftarrow CW_{min}$;
    end
end
```

4.2 Algorithm Explaination

Here's an explanation of the pseudocode:

1. **Main Function**

   Variables

   (a) **throughput:** an empty list to store the throughput for each time slot.
   (b) **totalTransmissions:** a counter to keep track of the total number of transmissions.
   (c) **successfulTransmissions:** a counter that counts the number of transmissions that were successful.

   Iterate through the following procedures for every time slot:

   (a) Identify the nodes trying to transmit at the moment and add them to the list of transmittingNodes.
   (b) If there are two or more nodes trying to transmit (contention occurs), update the totalTransmissions counter and carry out the following actions for each node:
   Call the O-BEB-Failure(node) function, which increases the failure counter and adjusts the contention window size (CW). Call the performBackoff(node) function which sets a random value between 0 and (CW-1) as backoff to the node.
   (c) If there is only one node trying to transmit, it is considered a successful transmission: Increment the successfulTransmissions counter and the totalTransmissions counter. Call the O-BEB-Success(transmittingNodes[0]) function, which updates the success counter, adjusts the CW, and handles CW boundary conditions.
   (d) Calculate the throughput for the current time slot by dividing the successfulTransmissions by the current time slot and append it to the throughput list.

   Finally, return the throughput list, successfulTransmissions, and totalTransmissions.

2. **O-BEB-Success() function:**

   (a) If the counterSuccess value is less than or equal to 14, the counter will be increased, and the CW will be reduced by dividing it by the square root of 2. If the CW becomes smaller than or equal to the minimum CW (CWmin), set the CW to CWmin.
   (b) Increase the CW by multiplying it by the square root of 2, then, if the success counter is greater than 14, reset the counter to 1. Set the CW to CWmax if it is greater than or equal to the maximum CW (CWmax).

3. **O-BEB-Failure() function:**

   (a) If the failure counter (counterFailure) is less than or equal to 14, increment the counter and increase the CW by multiplying it by 10. Set the CW to CWmax if it is greater than or equal to the maximum CW (CWmax).
   (b) Reset the counter to 1 and reduce the CW by dividing it by the square root of 2 if the failure counter is greater than 14. If the CW becomes smaller than or equal to the minimum CW (CWmin), set the CW to CWmin.

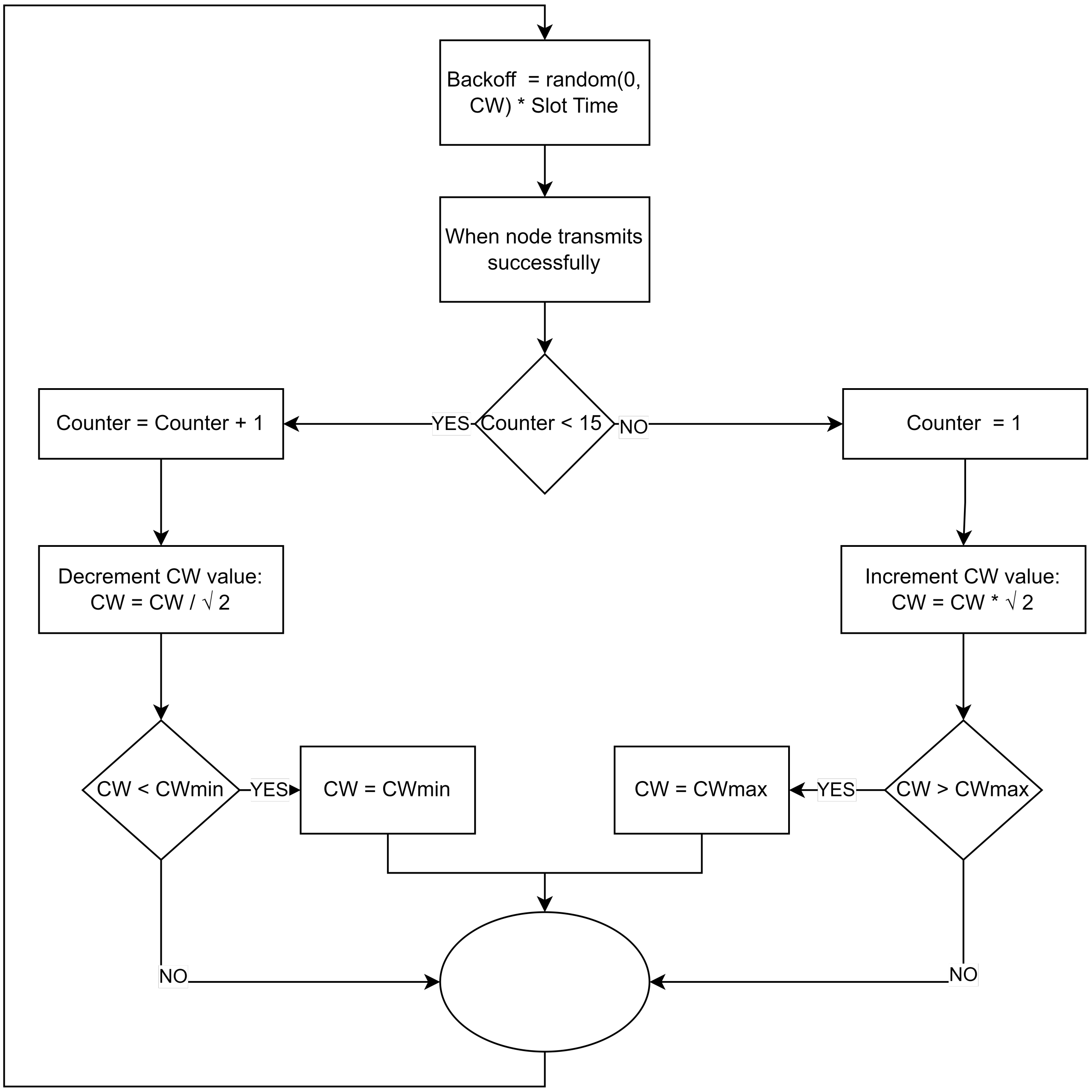

Backoff = random(0, CW) * Slot Time
When node transmits successfully
Counter < 15
YES
NO
Counter = Counter + 1
Counter = 1
Decrement CW value: CW = CW / √ 2
Increment CW value: CW = CW * √ 2
CW < CWmin
YES
CW = CWmin
CW = CWmax
YES
CW > CWmax
NO
NO

Figure 8: Flowchart for O-BEB Success

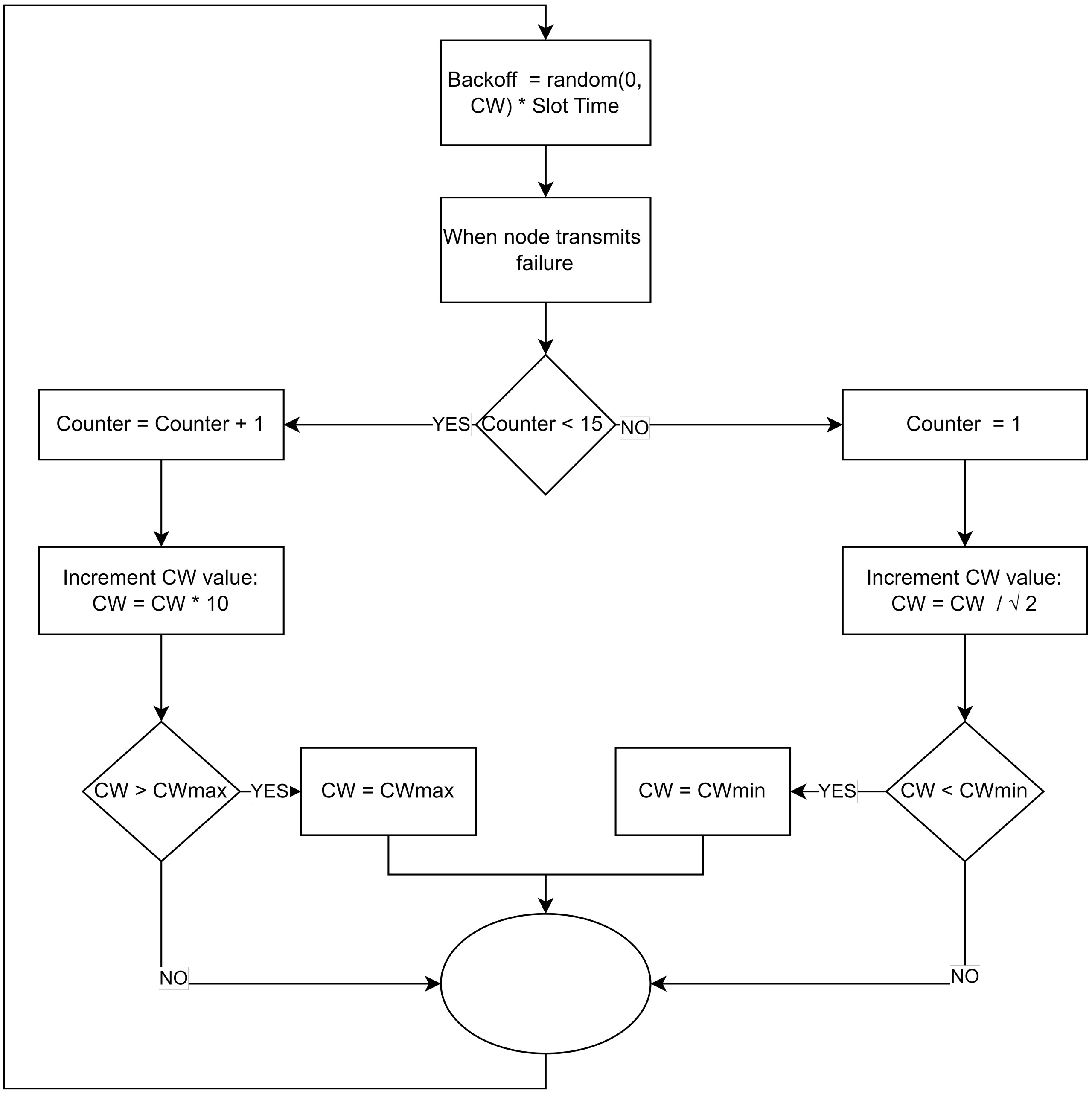

Backoff = random(0, CW) * Slot Time
When node transmits failure
Counter < 15
YES
NO
Counter = Counter + 1
Counter = 1
Increment CW value: CW = CW * 10
Increment CW value: CW = CW / √ 2
CW > CWmax
YES
CW = CWmax
CW = CWmin
YES
CW < CWmin
NO
NO

Figure 9: Flowchart for O-BEB Failure

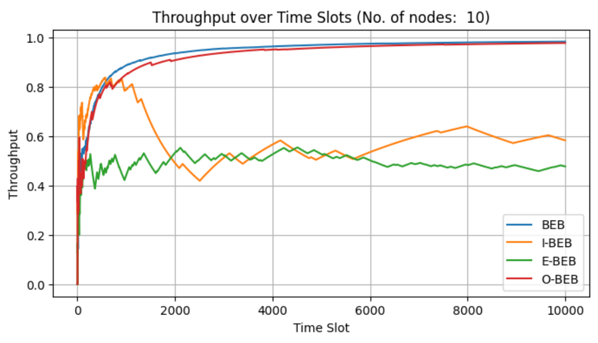
5 Results and Analysis

In this section, the results of the simulation of the O-BEB algorithm are shown in comparison to three other algorithms: BEB, I-BEB, and E-BEB.

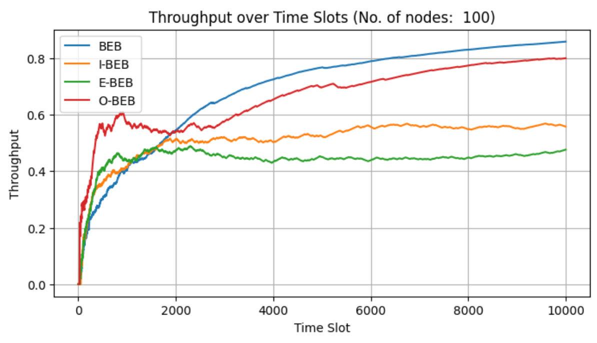
The algorithm's performance is evaluated by conducting experiments with varying numbers of nodes and analyzing the resulting throughput, both in terms of time slots and the number of nodes. Additionally, the ratio of successful packets sent to the total packets sent for varying numbers of nodes is also measured to assess the algorithm's effectiveness in minimizing collisions and maximizing successful transmissions in comparison to the other existing approaches.

5.1 Throughput vs. Time Slots:

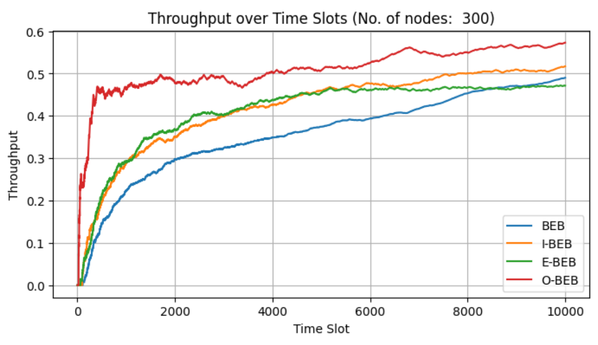
To evaluate the algorithm's performance over time, we simulated scenarios when the network consists of 10, 100, 300, 500, and 1000 nodes. For each scenario, we measured the throughput achieved by the network over time slots.



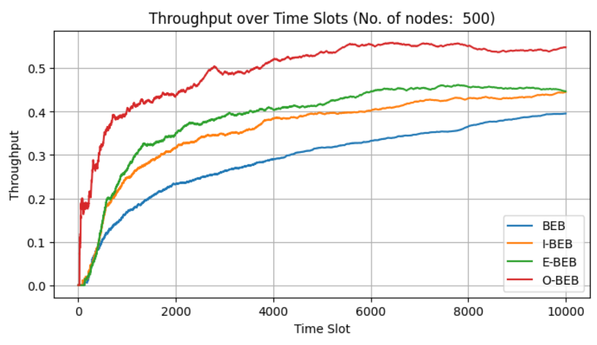
(a)



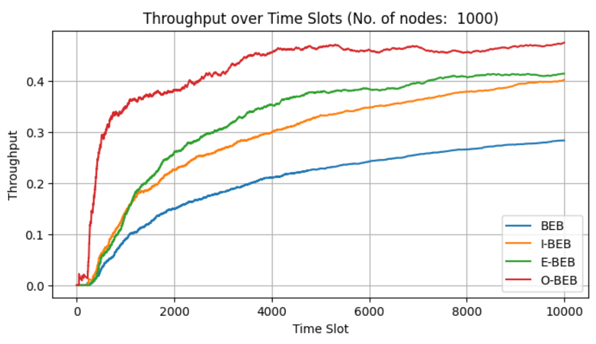
(b)



(c)



(d)



(e)

Figure 10: Throughput vs. Time Slots Graph for 10, 100, 300, 500, and 1000 number of nodes

Figure 10 clearly demonstrates the relationship between the number of nodes and the achieved throughput and how the O-BEB algorithm outperforms all other algorithms as the number of nodes increases. As the number of nodes increases, the throughput initially decreases due to an increase in collisions. However, as the algorithm adapts and adjusts the backoff durations based on collision occurrences, the throughput gradually improves. The curve tends to stabilize as the algorithm successfully mitigates collisions and optimizes transmission times, resulting in higher overall throughput.

It is seen from Figure 10(a) and 10(b) that the O-BEB performs slightly worse but almost on par with the classically used BEB algorithm. The reason behind this is that collisions are relatively less likely to occur in smaller networks. With fewer nodes contending for the medium, the probability of simultaneous transmissions and collisions is lower. In this case, the BEB algorithm can quickly resolve collisions and allow nodes to re-transmit without much delay. However, O-BEB does not perform much worse than BEB.

It can be seen from Figure 10(c), 10(d) and 10(e) that for an increasing number of nodes O-BEB performs much better than the other algorithms, and almost 20% better than classic BEB in case of 1000 nodes.

5.2 Throughput vs. Number of Nodes:

To examine the algorithm's scalability, we analyzed the throughput achieved by the network for varying numbers of nodes. We fixed the simulation duration and measured the average throughput for each node count. Figure 11 presents the relationship between throughput and the number of nodes.

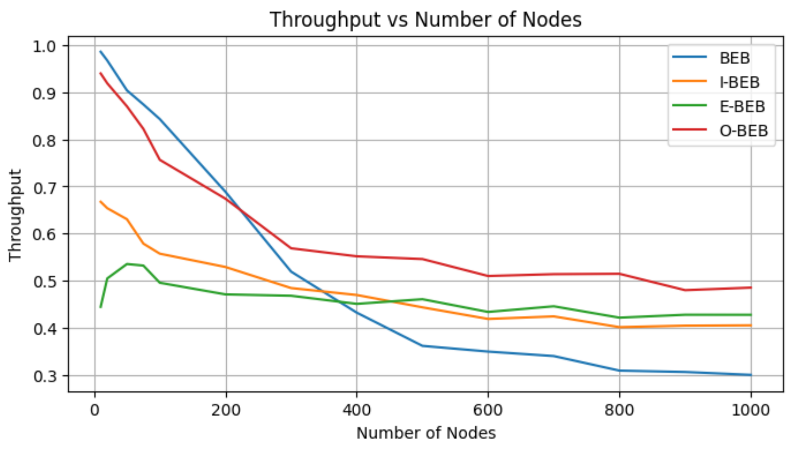


Figure 11: Throughput vs. Number of Nodes

The number of time slots is fixed at 10,000 and the range of the number of nodes is 1000 for this simulation. After 10000 time-slots, o-BEB gives a throughput just below 50% which is 20% greater than BEB, around 10% better than I-BEB, and around 6-7% better than E-BEB. This behavior is expected, as an increase in the number of nodes amplifies the likelihood of collisions and in this scenario, O-BEB is able to manage the size of the contention window in a way that leads to increased overall throughput compared to the other algorithms.

5.3 Ratio of Successful Packets Sent to Total Packets Sent vs Number of Nodes:

The effectiveness of the O-BEB algorithm in minimizing collisions and maximizing successful transmissions was evaluated by calculating the ratio of successful packets sent to the total packets sent in the network for different numbers of nodes. This ratio provides insight into the algorithm's ability to improve network efficiency and reduce unnecessary re-transmissions. For this simulation as well the number of slots is set to 10,000 and the number of nodes is taken to 1000.

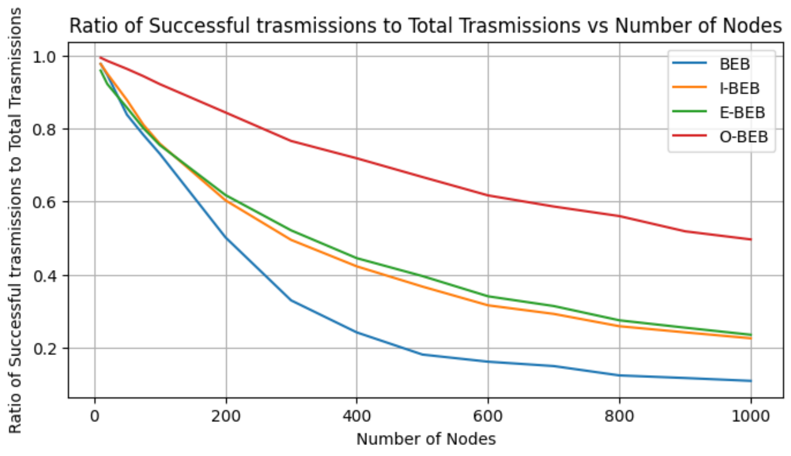


Figure 12: Throughput vs. Number of Nodes

Figure 12 depicts that ratio of successful packets sent to the total packets sent is around 0.5 for O-BEB, 0.2 for I-BEB and E-BEB and 0.1 for BEB which shows that there are 5 times lesser collisions in O-BEB compared to BEB. This improvement indicates that the O-BEB algorithm effectively manages collisions and optimizes transmission attempts, resulting in a higher proportion of successful transmissions.

6 Conclusions

In this research work, we suggested a brand-new algorithm for the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocols named O-BEB (Optimised Binary Exponential Backoff). Our comparison included the performance of BEB (Binary Exponential Backoff), I-BEB (Improved Binary Exponential Backoff), and E-BEB (Enhanced Binary Exponential Backoff), three currently used algorithms. We have shown that the O-BEB algorithm surpasses the other methods in terms of throughput, scalability, and the proportion of successfully delivered packets to all packets sent through thorough simulations and analysis.

Our findings unequivocally show that the O-BEB algorithm surpasses the competition across the board. As shown in the throughput vs time slots for various node counts, it demonstrates excellent adaptability, minimizing collisions and maximizing successful transmissions. The O-BEB algorithm demonstrates its scalability by maintaining a greater throughput than the others even as node density increases.

The O-BEB method regularly results in greater ratios of successful packets transmitted to total packets sent, a measure of network efficiency. Data distribution is made more effective by its capacity to control collisions and reduce retransmissions.

In conclusion, the O-BEB algorithm significantly outperforms the state-of-the-art algorithms, contributing significantly to CSMA/CA protocols. It is a compelling option for boosting network efficiency and data transmission dependability because of its optimisation methodologies, versatility, and improved performance across various network conditions.

7 Future Work

While this study demonstrates the effectiveness of the O-BEB (Optimized Binary Exponential Backoff) algorithm in improving the performance of the CSMA/CA protocol, there are several avenues for future research that can further enhance its capabilities and explore its applicability in specific network environments. The parameters of the algorithm can be further fine-tuned to get better throughput and scalability. It can also be testing using other parameters such as the fairness index. Future research should also consider testing the O-BEB algorithm in real-world scenarios to assess its robustness and effectiveness in practical networking environments while also optimizing it for specific environments.

References

[1] A. Zola, CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance), https://www.techtarget.com/searchnetworking/definition/CSMA-CA (2021).

[2] A. Aldraho, A. A. Kist, Enabling Dynamic Topologies in communication networks, https://ieeexplore.ieee.org/document/6096667 (2011).

[3] R. Doost-Mohammady, M. Y. Naderi, K. R. Chowdhury, Performance Analysis of CSMA/CA based Medium Access in Full Duplex Wireless Communications, https://download.arxiv.org/pdf/1512.04089v1.pdf (2015).

[4] C. Tang, L. Song, J. Balasubramani, S. wu, Comparative Investigation on CSMA/CA-Based Opportunistic Random Access for Internet of Things, https://www.researchgate.net/publication/262231791_Comparative_Investigation_on_CSMACA-Based_Opportunistic_Random_Access_for_Internet_of_Things (2014).

[5] A. Abbas, J. Ali, M. A. Rahman, S. Azad, Comparative Investigation on CSMA/CA-Based MAC Protocols for Scalable Networks, https://ieeexplore.ieee.org/document/7808354 (2016).

[6] T.-S. Ho, K.-C. Chen, Performance analysis of IEEE 802.11 CSMA/CA medium access control protocol , https://www.researchgate.net/publication/3678323_Performance_analysis_of_IEEE_80211_CSMACA_medium_access_control_protocol (1996).

[7] N. Shahin, R. Ali, S. W. Kim, Y.-T. Kim, Adaptively Scaled Back-off (ASB) mechanism for Enhanced Performance of CSMA/CA in IEEE 802.11ax High Efficiency WLAN, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8406219 (2018).

[8] B. Mawlawi, J.-B. Dore, N. Lebedev, J.-M. Gorce, Performance Evaluation of Multiband CSMA/CA with RTS/CTS for M2M Communication with Finite Retransmission Strategy , https://www.sciencedirect.com/science/article/pii/S1877050914014045?ref=pdf_download&fr=RR-2&rr=7a54762e5f229a8d (2014).

[9] N. Shenoy, J. Hamilton, A. Kwasinski, K. Xiong, An improved IEEE 802.11 CSMA/CA medium access mechanism through the introduction of random short delays, https://ieeexplore.ieee.org/abstract/document/7151090 (2015).

[10] B. M. Khan, F. H. Ali, Mobility Adaptive CSMA/CA MAC for Wireless Sensor Networks, https://link.springer.com/chapter/10.1007/978-3-642-21928-3_42 (2011).

[11] H. Masuno, K. Asahi, A. Watanabe, A. Ogawa, A study on energy-efficient protocol based on the CSMA/CA in ad-hoc networks, https://ieeexplore.ieee.org/document/4895382 (2008).

[12] R. Singh, Adaptive CSMA for Decentralized Scheduling of Multi-Hop Networks With End-to-End Deadline Constraints, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9377564 (2021).

[13] M. Derakhshani, T. Le-Ngoc, Adaptive Access Control of CSMA/CA in Wireless LANs for Throughput Improvement, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6831523&tag=1 (2013).

[14] D. M. Blough, C. Harvesf, G. Resta, G. Riley, P. Santi, A Simulation-Based Study on the Throughput Capacity of Topology Control in CSMA/CA Networks, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1599015 (2006).

[15] Y. Amara, R. Beghdad, Improving the Collision Avoidance of the CSMA/CA Medium Access Control Protocol , https://www.researchgate.net/publication/242638743_Improving_the_Collision_Avoidance_of_the_CSMACA_Medium_Access_Control_Protocol (2004).

[16] K. Wang, C. Sun, An Improved Backoff Algorithm of Ad Hoc Networks, https://ieeexplore.ieee.org/document/5362984 (2009).

[17] M. Al-Hubaishi, T. Abdullah, R. Alsaqour, A. Berqia, E-BEB Algorithm to Improve Quality of Service on Wireless Ad-Hoc Networks, https://www.researchgate.net/publication/290762280_E-BEB_Algorithm_to_Improve_Quality_of_Service_on_Wireless_Ad-Hoc_Networks (2012).

[18] F.-Y. Hung, I. Marsic, Effectiveness of physical and virtual carrier sensing in ieee 802.11 wireless ad hoc networks, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4224277&tag=1 (2007).

****** END ******