FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# IEEE 802.11ax Networks: Study and Assessment of New Techniques for the MAC Layer

**Paulo Alexandre Baptista Ferreira da Silva**

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Professor Nuno Teixeira de Almeida

11th of July, 2017

# Abstract

The ever-increasing utilization of the Internet and, specially, the wireless technologies, in the every day use, has led the scientific community to research more ways to deploy better and faster wireless accesses and connections. In fact, this turns to be of crucial importance, particularly, in dense wireless networks, where a huge number of users may see their quality of experience (QoE) severely degraded. Thus, the quality of service (QoS) is more demanding than ever, in a world where reliable connectivity and fast communications are of further and further importantance.

To achieve this, a new amendment for the most widespread wireless network protocol used, the Institute of Electrical and Electronics Engineers (IEEE) 802.11, was proposed. For this amendment (called 802.11ax), several technical areas are under study, in order to provide better physical and medium access protocols, that would allow faster wireless connectivity, in more difficult scenarios.

The absence of simulators for the new IEEE 802.11ax amendment leaves the scientific community with no practical means to assess the suggested enhancements the new standard is assumed to have.

Hence, this dissertation work seeks to develop a simulation model, that would provide ways to test the envisaged new mechanisms of 802.11ax Media Access Control (MAC) layer, under different configuration scenarios. The obtained simulation results could help the scientific community in the testing/understanding of 802.11ax behavior, thus, making more clear the MAC layer protocol/technical details that will improve the network performance.

The proposed approach is studying the current state of the art literature and evaluate the points where the current medium access mechanisms lack efficiency. A new protocol is envisaged and proposed for certain scenarios and is detailed, both theoretically and with the help of simple usage scenarios.

To validate the proposed solutions, testing current MAC layer protocols and mechanisms, using a network simulator, is the first step. Later, the new enhancements are to be applied to this same simulator. A new run of simulations should confirm and validate the proposal and the obtained results will be compared to the current MAC layer protocol.

# Acknowledgements

I would like to thank all the people involved in this work, that helped me with getting results of better quality. I am grateful to my supervisor, Professor Nuno Almeida, for all the suggestions and document proofreading, plus helping me publish a poster at the Workshop on ns-3 (WNS3), the annual Workshop of network simulator 3 (ns-3), which took place at Faculdade de Engenharia da Universidade do Porto (FEUP) on the 13th and 14th of June, 2017. I would like to thank Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC) researchers Carlos Leocádio, for helping me with implementation issues I was facing and all the very valid suggestions, and Helder Fontes, for questioning some of my first simulation results, which made me look into it and get better results.

I am also grateful to Professor Manuel Ricardo for his suggestions and availability to help whenever possible.

I am grateful to all the students at FEUP laboratory network, who gave me time to work and develop this dissertation, avoiding interrupting me whenever possible.

I would like to thank my fellow co-workers and students for their feedback, cooperation and friendship, specially Paulo Vaz who took the time to read this dissertation, pointing out some errors and giving suggestions.

I would like to thank Pedro Rodrigues for recommending me CLion IDE, which greatly improved my productive time.

I would like to thank my family: my parents and my brothers and sister for supporting me spiritually throughout the time I was studying for my degree and throughout my dissertation development and writing, for giving me the peace and stability needed for working and helping me through all the late nights. I want to thank my fiancee for all her support and understanding throughout this period of my life.

Paulo Silva

*"Who hath measured the waters in the hollow of his hand, and meted out heaven with the span, and comprehended the dust of the earth in a measure, and weighed the mountains in scales, and the hills in a balance?"*

Isaiah 40:12

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| AC | Access Category |
| ACK | Acknowledge |
| AMC | Adaptive Modulation and Coding |
| AP | Access Point |
| B-ACK | Block Acknowledgment |
| BEB | Binary Exponential Backoff |
| BSS | Basic Service Set |
| CAP | Controlled Access Phase |
| CCA | Clear Channel Assessment |
| CFP | Contention Free Period |
| CF-Poll | Contention Free Poll |
| CP | Contention Period |
| CSI | Channel State Information |
| CSMA/CA | Carrier Sense Multiple Access Collision Avoidance |
| CSMA/CDA | Carrier Sense Multiple Access Collision Detection and Avoidance |
| CSMA/CA-DB | Carrier Sense Multiple Access Collision Avoidance - Deterministic Backoff |
| CSMA/ECA | Carrier Sense Multiple Access Enhanced Collision Avoidance |
| DBCA | Dynamic Bandwidth Channel Access |
| DCF | Distributed Coordination Function |
| DIFS | Distributed Coordination Function Interframe Space |
| DLS | Direct Link Setup |
| DS | Distribution System |
| DSSS | Direct Sequence Spread Spectrum |
| EDCA | Enhanced Distributed Channel Access |
| EDCF | Enhanced DCF |
| ESS | Extended Service Set |
| FEUP | Faculdade de Engenharia da Universidade do Porto |
| FHSS | Frequency Hopping Spread Spectrum |
| GSM | Global System for Mobile Communications |
| HC | Hybrid Coordinator |
| HCCA | HCF Controlled Channel Access |
| HCF | Hybrid Coordination Function |
| HEW | High Efficiency WLANs |
| IEEE | Institute of Electrical and Electronics Engineers |
| IFS | Interframe Space |
| INESC TEC | Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência |
| IR | Infrared |

| | |
|---|---|
| ISM | Industrial Scientific Medical Band |
| LAN | Local Area Network |
| LLC | Logical Link Control |
| MAC | Media Access Control |
| MIMO | Multiple-input and Multiple-output |
| MPDU | MAC Protocol Data Unit |
| MU-MIMO | Multi-user Multiple-input and Multiple output |
| NAV | Network Allocation Vector |
| NIC | Network Interface Card |
| ns-3 | Network Simulator 3 |
| OFDM | Orthogonal frequency-division Multiplexing |
| OFDMA | Orthogonal frequency-division multiple access |
| PCF | Point Coordination Function |
| PDU | Protocol Data Unit |
| PER | Packet Error Rate |
| PHY | Physical Layer |
| PIFS | Point Coordination Function Interframe Space |
| PLCP | Physical Layer Convergence Procedure |
| QAM | Quadrature Amplitude Modulation |
| QoE | Quality of Experience |
| QoS | Quality of Service |
| RF | Radio Frequency |
| RTS/CTS | Request to send/Clear to send |
| RTT | Round Trip Time |
| SIFS | Short Interframe Space |
| SNIR | Signal to Interference plus Noise Ratio |
| STA | Station |
| STR | Simultaneous Transmit and Receive |
| SU-MIMO | Single-user Multiple-input and Multiple-output |
| TCP | Transmission Control Protocol |
| TS | Traffic Streams |
| TXOP | Transmission Opportunity |
| UDP | User Datagram Protocol |
| WLAN | Wireless local area network |
| WME | Wireless Multimedia Extensions |
| WMM | Wifi Multimedia |
| WNS3 | Worshop on ns-3 |

# Chapter 1

# Introduction

## 1.1 Context

The IEEE 802.11 wireless local area network (WLAN) standard, and its amendments, constitute the basis for essentially all the local wireless Internet connectivity throughout the world. The other important form of wireless connectivity is the use of cellular mobile networks. The current demand, though, insists continually in more throughput to the end user, due to the ever increasing amount of users and the kind of applications emerging, especially those involving multimedia data. The ongoing development of future IEEE 802.11ax WLAN standard (with expected publication in 2019) allows a window of time for a study regarding the possible enhancements that can be applied to the current WLAN standards. Several enhancements have been proposed in the scientific community, which will make a part of this study, and it is intended to acquire a good understanding, characterization and evaluation of the main issues to be resolved in this area. Thus, this dissertation is focused on producing a study in the area of IEEE 802.11ax networks, involving the evaluation of new features envisaged for the user, particularly in the MAC layer.

## 1.2 Problem/Motivation

The usage scenarios for wireless Internet access are expanding every day. The demand for higher throughput is here and IEEE has a task group currently working on the development of a new amendment, the IEEE 802.11ax.

The increasing number of devices and networks, and the traffic patterns demanded for the next few years motivated the development of a new Physical (PHY) and MAC layers IEEE 802.11 amendment to respond to the new challenges and usages [1]. One of the most demanding scenarios that WLANs face nowadays is the provision of high data rates in areas where the density of WLAN users is very high. For these sorts of scenarios, user requirements are not currently satisfied with present network technologies. Figure 1.1 describes three of those scenarios, presented as a motivation to the problem by the IEEE 802.11ax task group [1]: a stadium, a train, and an apartment building.

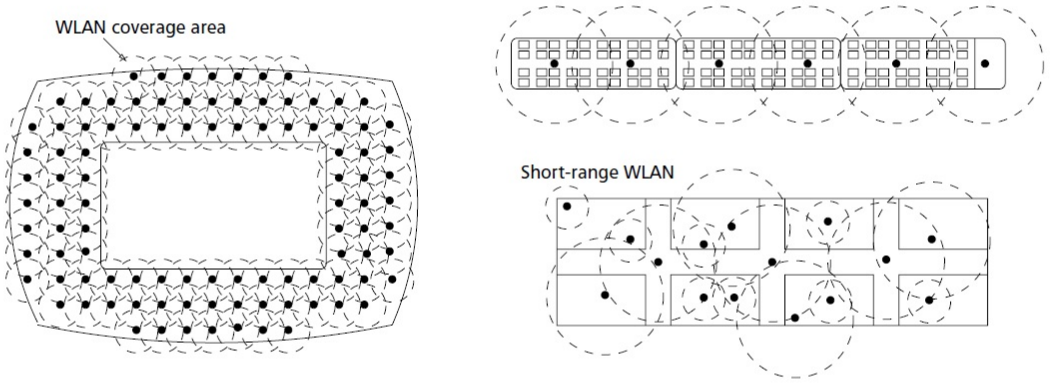


Figure 1.1: Dense WLAN scenarios, a stadium, a train, a floor of a building with several apartments [1]

## 1.3 Objectives/Challenges

The first objective of this work is to characterize the IEEE 802.11 and its amendments, and which contributions these amendments provided through time until the current standards, the IEEE 802.11n and the IEEE 802.11ac.

Another objective of this work is to identify the main technical contributions to the MAC layer that can be incorporated in the new IEEE 802.11ax amendment and compare these contributions with current standards. The analysis and comparison will be supported by a network simulation environment in order to quantify the expected performance achievements, in several different scenarios of usage, or testing different enhancements. Thus, taking into account the current progress of future 802.11ax standard, a main and important part of the work will be the adaptation/development of useful MAC layer improvements into a simulation model. This model, after its validation work, may become a relevant contribution to the network simulation community.

The objectives, requirements and challenges faced by every new IEEE 802.11 amendment are:

1. Coexistence: WLANs operate as unlicensed devices in the Industrial Scientific Medical (ISM) bands. A new protocol should deal with other wireless communication protocols operating in the same frequencies of IEEE 802.11;

2. Offer higher throughput: compared with 802.11ac amendment, IEEE 802.11 ax aims for a 4-fold throughput increase. To achieve this, multiple technologies may be used, such as Dynamic Clear Channel Assessment (CCA), Orthogonal frequency-division Multiple Access (OFDMA) and some advanced multiple-antenna techniques [1];

3. Energy efficiency: a fixed objective is that devices compliant with new amendment consumes no more energy per data rate, than devices compliant with the previous amendments;
4. Backward compatibility: it has always been a requirement for new IEEE amendments. IEEE 802.11ax WLANs should support devices using legacy 802.11 specifications.

The objectives for this work can be structured into the following aspects:

- Review the main improvements the IEEE 802.11 standard has acquired through the years and the MAC layer characteristics of present IEEE standards (802.11n and/or 802.11ac);
- Acquire relevant information of 802.11ax task group activities related to the MAC layer;
- Relate the new characteristics of 802.11ax MAC layer, with the ones available in current standards and other possible improvements;
- Find a new enhancement for the MAC layer that would result in performance improvements in dense WLAN scenarios.

It is important to define a strategy to analyze the MAC characteristics, delineating different configuration scenarios and using afterwards a network simulator tool (such as “ns-3”) to confirm the throughput enhancements and other variables of interest.

## 1.4 Document Structure

The remainder of this document is structured as follows. Chapter 2 shows an overview of the state of the art literature, delving into the PHY and MAC layer mechanisms in 802.11 standards, the current MAC layer protocols available and the IEEE 802.11 amendments. It focus then on the new enhancements 802.11ax is supposed to have. Chapter 3 describes the new proposed solution, a new MAC layer protocol to improve temporal efficiency. Chapter 4 describes the implementation procedures developed on the network simulator. Chapter 5 carries the obtained results for CSMA/CA and the new protocol developed and draws comparisons between the two. Chapter 6 presents the main conclusions and future work. Finally, appendix A provides the ns-3 source code for the two scenarios used in this work.

# Chapter 2

# State of the Art

In this chapter, an overview over the IEEE 802.11 standard is presented, with focus on the PHY and MAC layer mechanisms. The current MAC layer protocols available are detailed. Next, IEEE 802.11 amendments are presented, with focus on the new 802.11ax and its main objectives and challenges.

## 2.1 What is IEEE 802.11?

IEEE 802.11 is a set of specifications created by IEEE to standardize WLANs. This set is composed by MAC and PHY specifications, which regulate how communications should be addressed and which protocols should be used on each of the communication layers.

There are a set of fundamental characteristics that differentiate WLANs from common wired Local Area Networks (LANs). In a wired LAN it is assumed that each address corresponds to a physical machine. In WLANs that is not always true. The concept of station (STA) represents the addressable unit in these networks.

### 2.1.1 IEEE 802.11 PHY Layer

The physical layer specifications in IEEE 802.11 are different from the wired LANs. IEEE 802.11 PHY layer [10]:

1. Uses a medium that has neither absolute nor readily observable boundaries outside of which STAs with conformant PHY transceivers are known to be unable to receive network frames;

2. Is unprotected from other signals that are sharing the medium;

3. Communicates over a medium significantly less reliable than wired PHYs;

4. Assumes dynamic topologies;

5. Lacks full connectivity, and therefore the assumption normally made that every STA can hear every other STA is invalid (i.e., "hidden node" problem);

6. Has time-varying and asymmetric propagation properties;

7. Might experience interference from logically disjoint IEEE 802.11 networks operating in overlapping areas.

Because of limitations on wireless physical layer ranges, WLANs intended to cover large areas will have to be built over basic building blocks. In fact, these building blocks, nowadays, are growing with the number of users per area unit, which generates several problems that will be addressed later in this work. When providing QoS services, the MAC endeavors to provide QoS “service guarantees” within the limitations of the medium properties identified above [10]. This means that, due to the specificity of the medium and physical layer used, several protocols, techniques and standards have to be used in order to provide a reliable service.

The basic building block is the Basic Service Set (BSS). Figure 2.1 shows two BSSs, connected to a distribution system (DS) and forming an Extended Service Set (ESS).

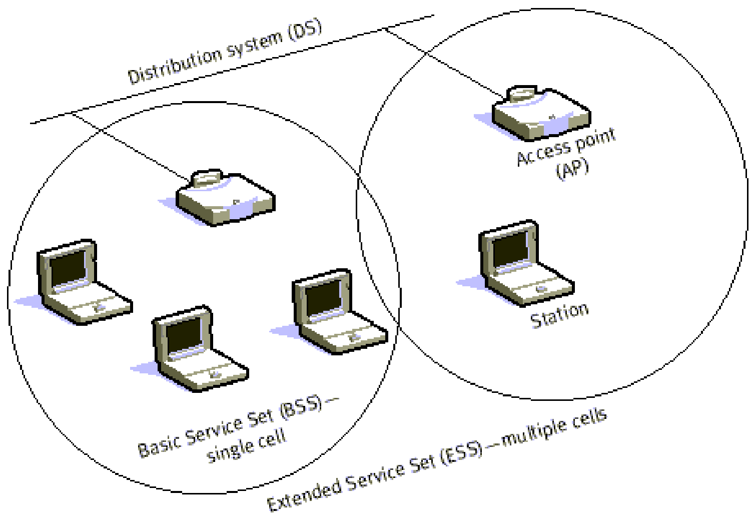

Distribution system (DS)
Access point (AP)
Station
Basic Service Set (BSS)—single cell
Extended Service Set (ESS)—multiple cells

Figure 2.1: Example of Basic and Extended Service Sets (BSS and ESS) with corresponding Distribution System (DS)

Another interesting concept to understand is the Area of coverage. For wireless PHYs, while the architecture diagrams show sharp boundaries for BSSs, this is simply a representation and not a reality. In fact, well-defined coverage areas do not exist. Due to its characteristics, wireless wave propagation is dynamic and unpredictable. Small changes in position or direction can result in big differences in the signal properties [10]. Figure 2.2 shows a signal strength map taken with Radio Mobile application simulation.

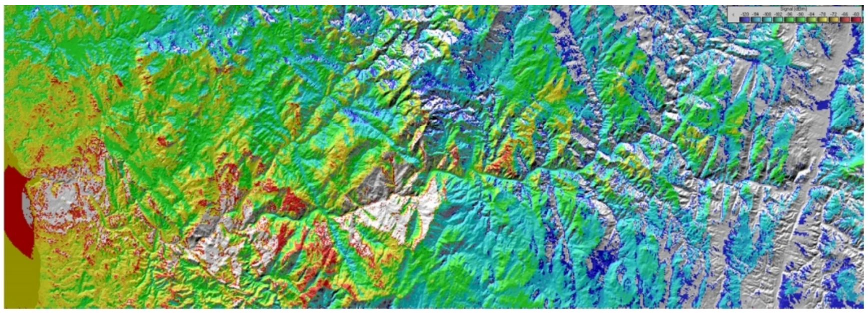


Figure 2.2: Example of a map created by a radio coverage application simulation

Figure 2.2 is a static map. However, in a common wireless network coverage scenario, propagation patters change dynamically as STAs and objects move [10]. In Figure 2.2 the blue and grey areas of the map show regions where the signal strength is very poor. The figure shows differences in field strength with different color patterns and indicates the variability of field strength in a static environment, as a simple example.

### 2.1.2 IEEE 802.11 MAC Layer

The basic 802.11 MAC layer defines access mechanisms for channel access: the Distributed Coordination Function (DCF) and Point Coordination Function (PCF). The Hybrid Coordination Function (HCF) is defined in 802.11e.

Figure 2.3 illustrates the MAC architecture [2].

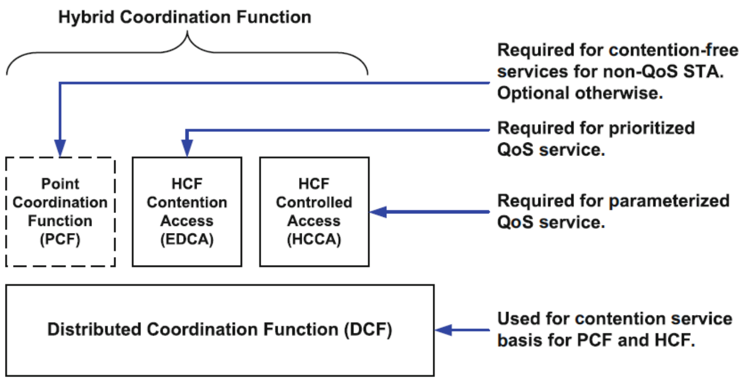

Hybrid Coordination Function
Required for contention-free services for non-QoS STA. Optional otherwise.
Required for prioritized QoS service.
Point Coordination Function (PCF)
HCF Contention Access (EDCA)
HCF Controlled Access (HCCA)
Required for parameterized QoS service.
Distributed Coordination Function (DCF)
Used for contention service basis for PCF and HCF.

Figure 2.3: MAC architecture [2]

#### 2.1.2.1 IEEE 802.11 DCF

DCF is the fundamental block for the 802.11 WLANs. It is mandatory, even for new amendments, for legacy compatibility reasons, and the basic medium access mechanism for both ad hoc and infrastructure WLAN modes, using CSMA/CA algorithm. In DCF, STAs contend for the transmission opportunities (TXOPs) in a distributed manner. A TXOP is a time interval when a particular STA has the right to initiate transmissions. Using a backoff mechanism, each station should wait for a random backoff interval before transmitting or retransmitting a frame for collision avoidance.

DCF provides best-effort service for asynchronous data transmission. Different from the wired networks such as Ethernet, wireless networks may suffer from the hidden terminal problem. In a WLAN with an AP, STAs communicating with the AP may not hear each other (such as STAs on the opposite edge of the geographical coverage area of the AP). This is because the wireless signal may attenuate too much. Figure 2.4 illustrates the hidden terminal problem [3].

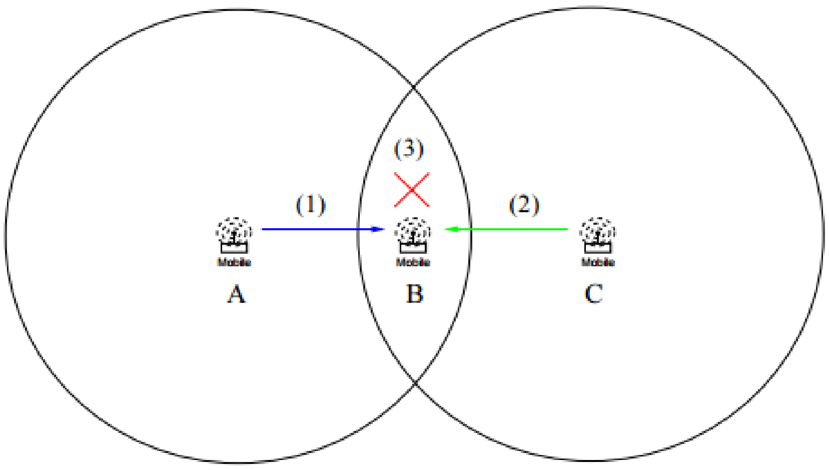


Figure 2.4: Hidden terminal problem [3]

In this case, A transmits to B (1); C does not sense A and transmits to B (2); interference then occurs at B and there is a collision (3).

In order to avoid the hidden terminal problem, DCF introduces a virtual carrier sense mechanism which is optional for a wireless device. In this mechanism, the source and destination stations exchange short request-to-send (RTS) and clear-to-send (CTS) frames with the network allocation vector (NAV) to block the neighboring stations including the hidden terminals from transmitting simultaneously. The RTS/CTS scheme can solve the hidden terminal problem, but at the same time it introduces the exposed terminal problem, i.e., it may unnecessarily block other transmissions that will not interfere with the ongoing one [2].

Figure 2.5 illustrates the exposed terminal problem [3].

In this case, B transmits to A (1); C wants to transmit to D but senses B (2); C refrains from transmitting to D.

One of the most representative characteristics of WLANs is then the use of CSMA/CA as the MAC protocol. It has been used in the 802.11 legacy and it is important that new amendments

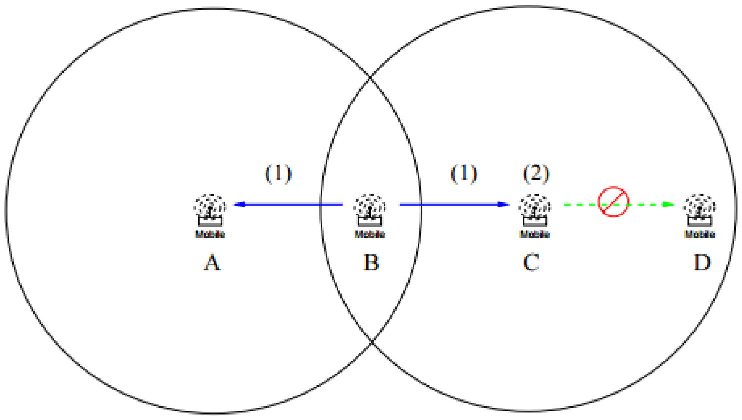


Figure 2.5: Exposed terminal problem [3]

deal with its characteristics to guarantee full compatibility.

#### 2.1.2.2 CSMA/CA

Figure 2.6 illustrates a simplified CSMA/CA flowchart [4].

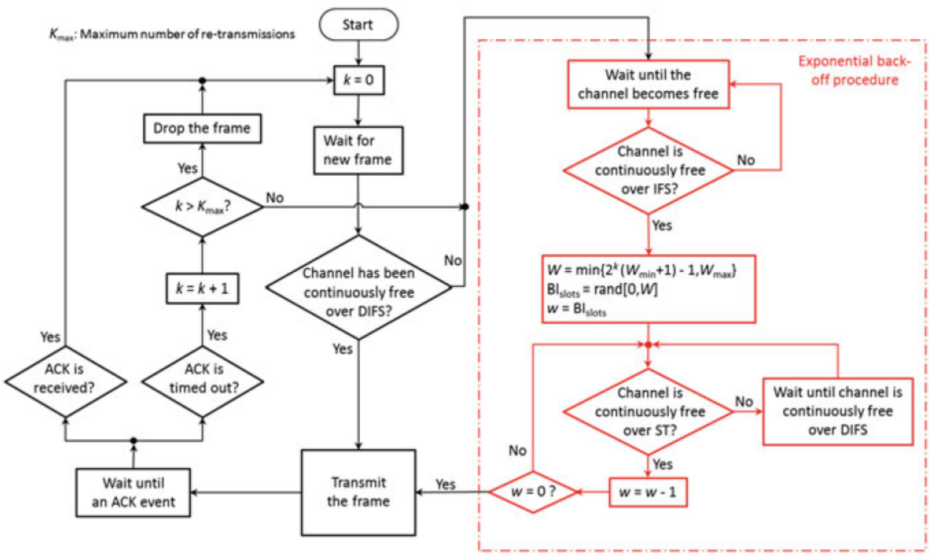

$K_{max}$: Maximum number of re-transmissions
Start
$k = 0$
Drop the frame
Wait for new frame
Yes
$k > K_{max}$?
No
Channel has been continuously free over DIFS?
No
Yes
$k = k + 1$
Yes
Yes
ACK is received?
ACK is timed out?
Wait until an ACK event
Transmit the frame
Exponential back-off procedure
Wait until the channel becomes free
Channel is continuously free over IFS?
No
Yes
$W = \min\{2^k(W_{min}+1) - 1, W_{max}\}$
$BI_{slots} = rand[0,W]$
$w = BI_{slots}$
Channel is continuously free over ST?
No
Wait until channel is continuously free over DIFS
Yes
No
$w = 0$ ?
$w = w - 1$
Yes

Figure 2.6: Simplified flowchart of CSMA/CA [4]

In CSMA/CA, when a node has a packet to transmit, it listens to the channel. When it detects the channel as free (i.e. the energy level on the channel is lower than the clear channel assessment (CCA) threshold), the node starts the backoff count [1]. By selecting a random value, the node that is trying to transmit a packet starts decreasing the counter while sensing the channel. If another transmission is sensed on the channel, from other nodes within the same WLAN or originated on other WLANs, the backoff counter is paused. When the channel is detected as free again, the backoff counter is then resumed. As soon as it reaches the value of zero, the node is able to transmit [1].

802.11 WLANs are defined by the BSS. This is a group of STAs under the control of a coordination function, either DCF or PCF. This way, all stations in a BSS can communicate, but due to interference, some STAs can be hidden from other nodes. The MAC layer is responsible for a series of functions, such as channel allocation procedures, error checking, fragmentation, frame formatting and Protocol Data Unit (PDU) addressing. In contention mode, the MAC layer coordinates STAs trying to transmit in a way that each STA should “contend” to access the channel. The transmission medium can operate only in this mode, Contention Period (CP) or alternate between a CP and Contention Free Period (CFP), in which the AP controls the medium and STAs do not need to contend [6]. At the MAC layer level, three types of frames can be distinguished:

- Management frames: STA associate or dissociate the AP, timing and synchronization, authentication and deauthentication;
- Control frames: handshaking and acknowledge (ACK);
- Data frames: transmission of data.

Figure 2.7 shows an IEEE 802.11 frame:

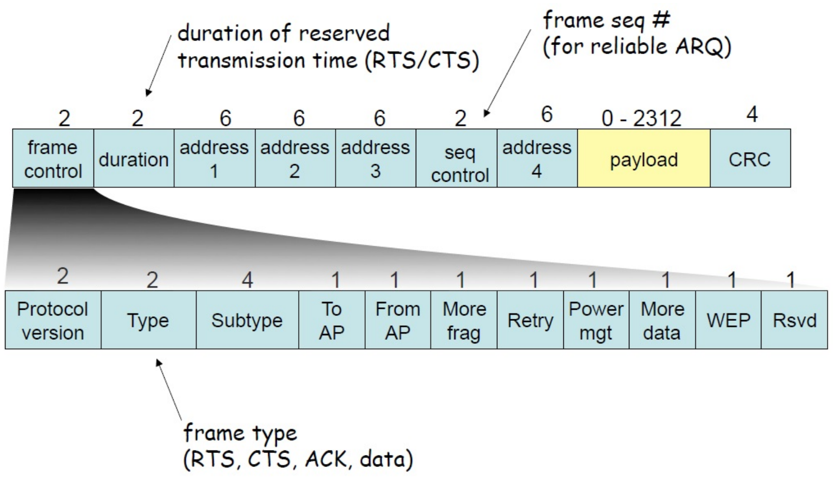


Figure 2.7: 802.11 frame [5]

The complete data unit from MAC to PHY is referred to as MAC Protocol Data Unit (MPDU), which contains the payload and header information. The time to complete a successful transmission is stored in the duration field, which is used by other STAs to calculate their NAV. The NAV is used to know the time until the current transmission is finished, and after this time the channel can be sensed for idle status again [6].

Between the transmission of frames, priority access to the wireless medium is controlled by the Interframe Space (IFS) time intervals [6]. In 802.11 there are 3 interframe space intervals: short IFS (SIFS), the smallest, followed by Point Coordination Function IFS (PIFS) and DCF interframe space (DIFS). SIFS has the highest priority and STAs that are only required to wait this time have priority access to the channel [6]. When an STA senses the channel idle, before sending a packet, the STA waits a DIFS period and then senses the channel again. If the channel is still idle it transmits its packet. The receiving station, after a checksum verification, waits a SIFS period and the sends the ACK. Figure 2.8 shows how transmission of data without RTS/CTS works.

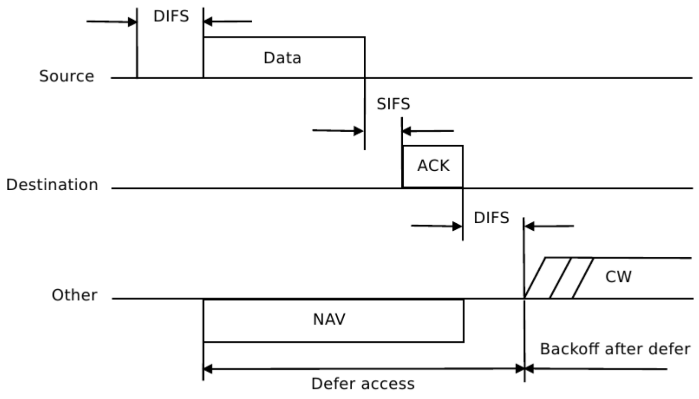


Figure 2.8: Transmission of a data frame without RTS/CTS [6]

Due to the source STAs inability to hear its own transmission, if a collision occurs the STA continues to send the whole packet, which results in a waste of bandwidth and it is clearly a source of inefficiency.

Using RTS/CTS, the STA sends a RTS control frame to a specific destination. All other STAs hear the RTS frame and adjust their NAV. After a SIFS period of time, the destination STA responds with a CTS packet. This packet is also heard by the other STAs and the NAV is once again adjusted. If the source STA successfully receives the CTS packet, it is assured that the channel is reserved and stable. The STAs can update their NAV based on both the RTS packet and the CTS packet, which helps to avoid the hidden terminal problem [6]. Figure 2.9 shows how transmission of data with RTS/CTS works.

The incident of collisions is reduced through a random backoff procedure. When an STA senses the channel busy it waits a DIFS period until the channel is idle. Thereafter it computes a random backoff time. Each STA decrements its timer when the channel is idle. If the channel becomes busy, the timer stops. If the timer reaches zero, the STA can send its packet [6]. Main problems with CSMA/CA are:

- CSMA/CA is used for multiple access, sharing the air using media access, which results in hidden terminal problems, and near-far problems;

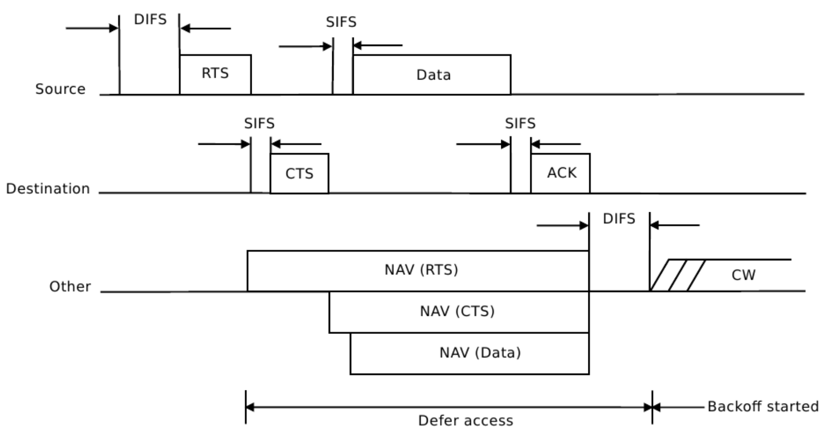


Figure 2.9: Transmission of a data frame with RTS/CTS [6]

- The AP admin chooses a frequency for the AP, which causes possible interferences, since the channel can be the same chosen for a neighboring AP;

A basic pseudo-code for CSMA/CA algorithm is presented in figure 2.10 [7], which will be crucial for the new enhancements proposed later in this work. In the figure, r represents the current number of retransmissions, s represents the current backoff stage. R represents the maximum number of retransmissions allowed by the protocol, and S represents the maximum backoff stage. b is the binary exponential backoff (BEB) number.

#### 2.1.2.3 IEEE 802.11 PCF

PCF is an optional MAC technique defined in the IEEE 802.11 standards by which the channel access control is centralized. PCF adopts a centrally controlled polling method and an AP in a WLAN serves as the point coordinator to coordinate the frame transactions within the network. Therefore, with PCF, the network is operated in the “centralized” mode. PCF is performed based on DCF in the 802.11 MAC sublayer architecture. An AP waits for a PIFS duration instead of the DIFS duration to access and occupy the channel. Since PIFS is shorter than DIFS, the AP is granted a higher priority to access the wireless channel. The AP sends contention-free-poll (CF-Poll) frames to the stations that are able to operate in the PCF mode, and permits one of them to transmit a frame.

Due to the priority of PCF over DCF in channel access, the DCF-only stations might not gain access to the medium. To ensure that all stations have transmission opportunities, alternative intervals have been designed to provide both (contention free) PCF access and (contention-based) DCF access. The AP sends beacon frames with fixed intervals, for example every 0.1 s. The time between two beacon frames is divided into two periods, the CFP and the CP, which are repeated continuously. When stations hear a beacon frame, they start their NAVs for the CFP period to halt

```
while the device is on do
    $r \leftarrow 0$; $s \leftarrow 0$;
    $b \leftarrow \mathcal{U}[0, 2^s \mathrm{CW}_{\min} - 1]$;
    while there is a packet to transmit do
        repeat
            while $b > 0$ do
                wait 1 slot;
                $b \leftarrow b - 1$;
            Attempt transmission of 1 packet;
            if collision then
                $r \leftarrow r + 1$;
                $s \leftarrow \min(s + 1, S)$;
                $b \leftarrow \mathcal{U}[0, 2^s \mathrm{CW}_{\min} - 1]$;
        until $(r = R)$ or (success);
        $r \leftarrow 0$;
        $s \leftarrow 0$;
        if success then
            $b \leftarrow \mathcal{U}[0, 2^s \mathrm{CW}_{\min} - 1]$;
        else
            Discard packet;
            $b \leftarrow \mathcal{U}[0, 2^s \mathrm{CW}_{\min} - 1]$;
    Wait until there is a packet to transmit;
```

**Algorithm 1:** CSMA/CA

Figure 2.10: CSMA/CA pseudo-algorithm [7]

the channel access. During the CFP time, the AP sends CF-Poll frames to all of the stations. It sends one frame to an STA at a time in order to provide it an opportunity to send a frame. Then during the CP, DCF is used and stations can contend for the channel.

PCF allows for a better management of QoS by the centralized control. However, although PCF is more suitable to support synchronous data transmissions in a WLAN, classes and priorities of traffic which are usually adopted in other QoS mechanisms are not defined. PCF is optional and not required in the interoperability standard by the WiFi Alliance. Consequently, it is rarely implemented in the 802.11 network interface cards in practice [2].

#### 2.1.2.4 IEEE 802.11e Enhanced Distribution Channel Access (EDCA) in HCF

Multimedia applications such as video and audio have QoS requirements such as bandwidth, delay, jitter, and packet loss which are different from data service. Wireless multimedia extensions

(WME) is an interoperability certification of the WiFi Alliance, and also known as WiFi multimedia (WMM). The 802.11e amendment specifies a set of modifications to the MAC layer to enhance QoS provisioning.

A new coordination function, named HCF, has been added in 802.11e to enhance the MAC protocols of DCF and PCF. HCF is similar to the mechanisms specified in the legacy 802.11 MAC and has defined the contention-based and reserved contention-free channel access schemes: EDCA and HCF controlled channel access (HCCA), respectively. In order to support QoS provisioning in 802.11 WLANs, HCF introduces a number of QoS-oriented mechanisms and frame subtypes. Four access categories (ACs) of traffic are prioritized by WME, which are voice, video, best effort, and background. The primary principle to provide QoS in EDCA is to give the multimedia traffic a high priority and the best-effort data traffic a low priority in channel access. The channel access of each AC follows DCF but uses a set of differentiated enhanced DCF (EDCF) channel access parameters. If a frame from a higher-priority flow is to be sent, it waits for less time on average than that with a lower priority. As a result, the traffic with a higher priority has a better chance of accessing the channel and being sent. This is accomplished through the modification of the backoff parameters in the traditional CSMA/CA. Thus, delay sensitive data are protected and QoS is better supported.

Figure 2.11 illustrates the parallel backoff entities in a single IEEE 802.11e station. The traffic flows belonging to the four ACs are handled by four independent backoff entities, and an arbitration is performed inside an STA to handle the internal collision among the entities. In EDCA, an STA can access the channel without contention during the period of its TXOP. Within the bounded time period of a TXOP, an STA can send a number of frames given that the transmissions do not exceed the duration limit of the TXOP. In the case that a frame is too large to be transmitted within a single TXOP, the station should fragment the frame into multiple frames with a smaller size. By using the time limit of TXOPs, it can be avoided that a low-rate station occupies too much channel time to transmit frames in an 802.11 WLAN. In addition, block acknowledgment (B-ACK) is adopted which can acknowledge an entire TXOP by using a single ACK frame. This scheme can reduce the overhead of the acknowledgment especially when TXOPs are long and multiple frames are delivered within one TXOP. Furthermore, in supporting QoS, the class of service is defined with two values: QoSAck and QoSNoAck. QoSNoAck is used to inform that a frame is not acknowledged. Thus, retransmissions of highly time-critical data, which are unnecessary, can be avoided. The released IEEE 802.11-2007 standard has included this amendment to provide statistical instead of hard QoS support.

#### 2.1.2.5 IEEE 802.11e HCCA in HCF

HCCA is similar to PCF. However, there are several critical differences between them as listed below.

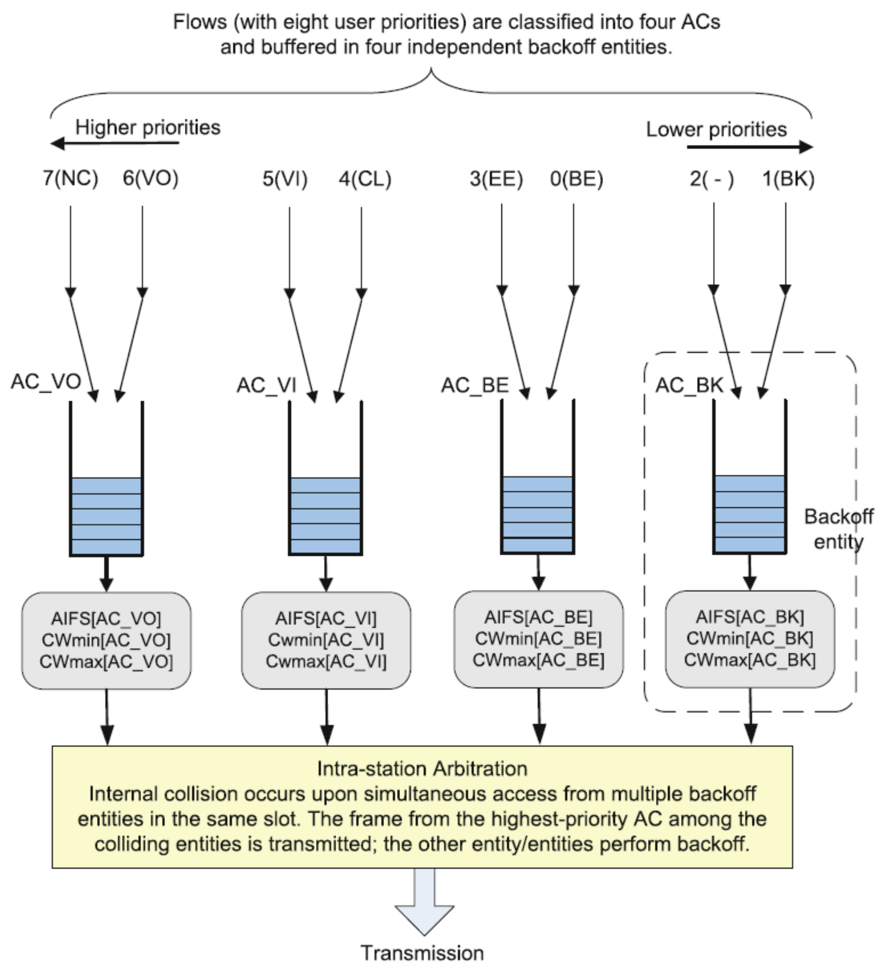

Flows (with eight user priorities) are classified into four ACs and buffered in four independent backoff entities.
Higher priorities
Lower priorities
7(NC)
6(VO)
5(VI)
4(CL)
3(EE)
0(BE)
2( - )
1(BK)
AC_VO
AC_VI
AC_BE
AC_BK
Backoff entity
AIFS[AC_VO] CWmin[AC_VO] CWmax[AC_VO]
AIFS[AC_VI] Cwmin[AC_VI] Cwmax[AC_VI]
AIFS[AC_BE] CWmin[AC_BE] CWmax[AC_BE]
AIFS[AC_BK] CWmin[AC_BK] CWmax[AC_BK]
Intra-station Arbitration
Internal collision occurs upon simultaneous access from multiple backoff entities in the same slot. The frame from the highest-priority AC among the colliding entities is transmitted; the other entity/entities perform backoff.
Transmission

Figure 2.11: Four parallel backoff entities for the ACs with different EDCA parameter sets and intracontention in one IEEE 802.11e station [2]

- In PCF, the time duration between two adjacent beacon frames is partitioned into two intervals, CFP and CP. In HCCA, CFPs are allowed to start at anytime inside a CP. Such CFP is referred to as the controlled access phase (CAP) in 802.11e. An AP can initiate a CAP any time when it wants to transmit a frame to or receive a frame from an STA by contention-free channel access. During a CAP, the access to the wireless channel is controlled by the hybrid coordinator (HC), i.e., the AP. On the other hand, inside a CP, all stations contend for the channel access via EDCA.

- As mentioned earlier, PCF does not define prioritized classes. HCCA specifies ACs and traffic streams (TS). Thus an HC can construct a queue for each session (stream), rather than for each station. In addition, HCF can coordinate these streams or sessions in any fashion (not just round-robin). Meanwhile, the STAs report their queue lengths of all ACs to the HC, and then the HC may adjust the scheduling accordingly.

- In HCCA, an STA may send multiple frames in a burst inside a given period of time determined by an HC, while this mechanism is not provided in PCF.
- During a CAP in HCCA, stations can also send CF-Poll frames to the HC to request data transmissions.

It is considered that HCCA is the most advanced (and complex) coordination function in the IEEE 802.11 standard family. In HCCA, QoS for each individual traffic flow can be specified with high accuracy by specific transmission parameters (such as data rate, jitter, etc.). Therefore, multimedia applications can be supported more effectively in a WLAN. However, since HCCA functionality is not mandatory in 802.11e, there are few (if any) AP products that implement HCCA at present [2].

## 2.2 Challenges for Dense WLAN scenarios

IEEE 802.11ax will be designed to solve problems of connection deployment in dense WLAN scenarios, which not only involves a higher throughput rate, but it also requires a better management of the multiple medium access, at the MAC layer level. The challenges faced by a new amendment on dense WLAN scenarios are interference issues, which increase the packet error rate and reduce the number of simultaneous transmissions, since in a certain space the neighboring WLANs are prevented from access the channel. Another problem has to do with two or more stations reaching zero at the backoff counter simultaneously, when many STAs are gathered under the same WLAN (increased collision rate) [11]. The current WLANs adopt the IFS, a backoff window size and beacon to control the MAC. In a dense scenario, these collision avoidance mechanisms are not ideal for a number of reasons [12]:

- CSMA/CA can lead to high collision rates and channel usage can then be worst. The reason is CSMA/CA choosing a low initial value for the backoff window because of an optimistic idea that there is a low level of congestion. This means CSMA/CA is not programmed by default for dense scenarios;
- CSMA/CA can lead to a “fairness problem” because of its BEB algorithm, favoring an STA having the last successful transmission. This can lead to a degree of starvation, where certain STAs will have a huge amount of time to wait between transmissions;
- RTS/CTS, which is intended to solve the “hidden node” problem, has an overhead caused by RTS and CTS packets;
- RTS/CTS may not solve the “exposed terminal problem”, when an STA that is close by, but associated with another AP, overhears the exchange and afterwards it is signaled to do backoff and stop transmitting during the RTS time;

- Wireless mediums are noisy and unreliable which results in packet loss even when no collisions occurred;
- Contention window control mechanisms are very complex to implement.

## 2.3 IEEE 802.11 Amendments

IEEE 802.11 standard, as it was stated before, is a set of PHY and MAC specifications to implement WLANs among STAs, with or without APs [2]. The radio frequency (RF) communications are operated in the 2.4, 5, and 60 GHz frequency bands and are utilized in the PHY layer and there are a set of MAC protocols used for channel access and resource management. The first approved version of IEEE 802.11 was released in June 1997. Initially, the goal was to develop wireless connectivity reaching 1Mbps data rates. Afterwards, enhancements have been developed by the work groups and Table 2.1 shows a brief summary of those enhancements.

Table 2.2 shows future amendments, still in progress, where 802.11ax is indicated. A few of the amendments will be addressed briefly to highlight some of the enhancements.

The first version, IEEE 802.11-1997 describes three solutions for the PHY layer: frequency hopping spread spectrum (FHSS), direct sequence spread spectrum (DSSS) and infrared PHY schemes. FHSS and DSSS use RF transmission, operating at the 2.4GHz band, the ISM band. The spectrum is sub-divided into 14 channels and each channel spans over 5MHz. The center frequency of the first channel is 2.412GHz. As a result, the center frequencies of two STAs overlapped must be separated by four channels to avoid signal frequency overlapping [2]. 802.11a use Orthogoanl Frequency Division Multiplexing (OFDM) for the PHY layer in the 5GHz band and adopts the same MAC protocol and frame format as 802.11 legacy standard, though it increased the throughput. Using 5GHz band has the advantage of potentially less interference [2]. In 802.11b, as the channel has a bandwidth of 22MHz, there are now three "non-overlapping" channels (with indexes 1, 6, and 11). There was also a throughput increase with this amendment. The amendment 802.11n was published in October 2009 and improved the other previous standards by introducing Multiple-input and Multiple-output (MIMO) antennas. The throughput was set at a maximum limit as high as 600Mbps. IEEE 802.11ac was based on 802.11n in the 5GHz band, but included a modulation up to 256 Quadrature Amplitude Modulation (QAM), rather than 64-QAM. It also introduces Multi-user MIMO (MU-MIMO). This implementation supported a total throughput up to 1.3Gbps [2].

## 2.4 MAC and PHY Layer Enhancement Areas for 802.11ax

IEEE 802.11ax will develop new PHY and MAC layer level enhancements that will improve performance and throughput. The four main technical improvements can be listed as:

Table 2.1: The IEEE 802.11 standard and its amendments [2]

| Number | Title | Comment |
|---|---|---|
| 802.11-1997 | IEEE Standard for WLAN MAC and PHY Specifications | Initial standard. 1 and 2Mbps, 2.4 GHz RF and infrared (IR) standard |
| 802.11-1999 | Part 11: WLAN MAC and PHY Specifications | |
| ISO/IEC 8802.11-1999 | IEEE Std 802.11-1999 (R2003) | International standard |
| 802.11a | High Speed Physical Layer in the 5 GHz band | 54 Mbps OFDM PHY @ 5GHz |
| 802.11b | Higher Speed PHY Extension in the 2.4 GHz Band | 11 Mbps DSSS PHY @ 2.4 GHz |
| 802.11c | MAC Bridges | Bridging in wireless bridges or access points, included in IEEE 802.1D standard |
| 802.11d | Operation in Additional Regulatory Domains | Allow devices to comply with regional requirements |
| 802.11e | MAC Enhancements | Support for QoS |
| 802.11f | Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation | Released as 802.11.1 and withdrawn by IEEE-SA Standards Board on 2006/2/3 |
| 802.11g | Further Higher Data Rate Extension in the 2.4 GHz Band | 54 Mbps OFDM PHY @ 2.4 GHz |
| 802.11h | Spectrum and Transmit Power Management Extensions in the 5 GHz Band in Europe | In Europe, 5 GHz devices must implement 802.11h |
| 802.11i | MAC Security Enhancements | MAC security enhancements, known as WPA and WPA2 from WiFi Alliance |
| 802.11j | 4.9 GHz to 5 GHz Operation in Japan | Compliance with Japanese 5 GHz spectrum regulation |
| 802.11k | Radio Resource Measurement of Wireless LANs | Discover the best available access point |
| 802.11ma | 802.11 Standard Maintenance Revision | Prepared for 802.11-2007 that supersedes 802.11-1999 |
| 802.11mb | 802.11 Accumulated Maintenance Changes | Second maintenance, prepared for 802.11-2012 |
| 802.11n | Enhancements for Higher Throughput | Increase the maximum net data rate to 600 Mbps using MIMO, frame aggregation, etc. @ 2.4 and 5 GHz |

| | | |
|---|---|---|
| 802.11p | Wireless Access for the Vehicular Environment | Car to car communication, closely related to IEEE 1609 |
| 802.11r | Fast BSS Transition | Permit continuous connectivity handoffs in a seamless manner |
| 802.11s | Mesh Networking | Transparent multi-hop operation |
| 802.11t | Recommended Practice for the Evaluation of 802.11 Wireless Performance | Develop 802.11.2, administratively withdrawn by IEEE-SA on 2006/2/3 |
| 802.11u | Interworking with External Networks | Convergence of 802.11 and Global System for Mobile Communications (GSM) |
| 802.11v | Wireless Network Management | Allow client devices to exchange information about network topology |
| 802.11w | Protected Management Frames | Increase the security of management frames |
| 802.11y | 3650–3700 MHz Operation in USA | High powered equipment to operate using the 802.11a protocol on a co-primary basis |
| 802.11z | Extensions to Direct Link Setup (DLS) | AP independent DLS |
| 802.11-2012 | New Release | Include Amendments k, n, p, r, s, u, v, w, y, and z |
| 802.11aa | Video Transport Streams | MAC enhancements for robust audio and videostreaming |
| 802.11ac | Very High Throughput <6 GHz | Enhancements for >1 Gbps throughput below 6 GHz |
| 802.11ad | Very High Throughput 60 GHz | Enhancements for >1 Gbps throughput @ 60 GHz band |
| 802.11ae | Prioritization of Management Frames | Communicate management frame prioritization policy |
| 802.11af | TV White Spaces | Geolocation-based spectrum databases and channel sensing |

Table 2.2: The IEEE 802.11 amendments in progress [2]

| Number | Title | Comment |
|---|---|---|
| 802.11mc | Standard Maintenance | Roll-up of 802.11-2012 with the aa, ac, ad, ae and af amendments, prepared for 802.11-2016 |
| 802.11ah | Sub 1 GHz band, Machine-to-Machine communications | Enhanced power saving mechanisms and efficient small data transmissions |
| 802.11ai | Fast Initial Link Setup | Reduce link setup time to below 100 ms |
| 802.11aj | China Millimeter Wave | Very high throughput WLAN using mmWave Ml MO @ 45 GHz |
| 802.11ak | General Links | Support IEEE 802.11 links for transit use in bridged networks |
| 802.11aq | Pre-association Discovery | Further discover the services running on a device or provided by a network |
| 802.11ax | High Efficiency WLANs | Dynamic channel bonding, multi-user uplink MIMO, and full-duplex wireless channel |
| 802.11ay | Enhancements for Ultra High Throughput in and around the 60 GHz Band | Extension of 802.11ad to extend throughput, range, and use-cases by channel bonding, MIMO and higher modulation schemes |
| 802.11az | Enhancements for Positioning | Enables determination of absolute and relative position with better accuracy |

1. Spatial reuse;
2. Temporal efficiency;
3. Spectrum sharing;
4. Multiple-antenna technologies.

As in article [11], these four enhancements can achieve better throughput, having in mind the need for backwards compatibility towards 802.11 legacy and energy consumption requirements.

### 2.4.1 Spatial Reuse

Using CSMA/CA and conservative CCA and transmit power levels may result in a very limited spatial reuse. A configuration set per se at conservative levels minimizes interference between different WLANs, supporting thus higher transmission rates. The number of transmissions in a concurrent fashion is reduced, which may decrease the throughput. Alternatives to reach the best balance between individual transmission rates and the amount of concurrent transmissions that maximize throughput include adapting dynamically the CCA level and the use of directional transmissions (beamforming).

- Dynamic adaptation of the transmit power and CCA levels

Reducing the transmission power reduces also its area of influence, which benefits the spatial reuse. This may result in a larger number of errors and lower transmission rates, and also increase the WLAN chances to transmit. The nodes increase their CCA level, thus demanding higher energy levels to consider the channel as occupied and, consequently, pausing the backoff countdown. The downside trade-off is interference.

- Beamforming

Current transmission schemes spread the energy in all directions, which fills the channel with energy where it is not necessary. When conglomerating the energy at the desired destination improves the spatial reuse because the devices placed in other points will sense the channel as idle and will start their own transmissions at the same time. However, nodes outside the beam may also become hidden nodes.

### 2.4.2 Temporal Efficiency

CSMA/CA channel access scheme has its own overhead caused by the backoff countdown, its packet headers, interframe spaces and both collisions and retransmissions involved in the protocol.

- Control Packets

Can result in large overheads since they are transmitted at a low rate for legacy compatibility purposes.

- Packet headers, aggregation and piggy-backing

Introduced in 802.11n to reduce temporal overheads by combining short packets into one larger packet. Then it is possible to reduce their size through variable size header mechanisms. The piggybacking of ACKs with DATA will improve efficiency, as stated in [11].

- Efficient retransmissions

With CSMA/CA, each packet error will require a full packet retransmission. A work around, would be the use of incremental redundancy–based ARQs, which can reduce the time spent in retransmissions.

- Simultaneous transmit and receive

Full-duplex communication can theoretically double the channel capacity. Using CSMA/CA, the only way they can have full duplex communication is if they finish their backoff countdown simultaneously. This would require a pseudo random backoff counter mechanism. Another way, presented by the IEEE 802.11 High Efficiency WLANs (HEW) Study Group, is implementing a Simultaneous Transmit and Receive (STR) mechanism in WLAN, with increase complexity only at the AP, where the AP uses knowledge of path loss between STAs to Group STAs for In band STR transmission, which is compatible with other legacy 802.11 deployments [13].

- Collision-free MAC protocols

Two possibilities, according to Bellalta: "move to a centralized solution (HCCA – never used in WLAN) or enhanced CSMA/CA (i.e. CSMA/ECA – Enhanced Collision Avoidance, with backward compatibility, easily implemented and outperforming CSMA/CA)" [7], [14], [1], [15] and [11].

### 2.4.3 Spectrum Sharing

- Dynamic Channel Bonding

Extending the Dynamic Bandwidth Channel Access (DBCA) from 802.11ac. This way, only the available channel width is used at each transmission, so the other nodes on the network adapt to the spectrum occupancy at a given time instant. This improves spectrum sharing as well.

- OFDMA

This enhancement will improve throughput since it "divides the channel width into multiple narrow channels. Then, these narrow channels can be used to transmit to multiple users in parallel. Each 20 MHz subchannel can be independently allocated to a different user. The RTS packets have been extended to announce the subchannels allocation to the STAs. OFDMA may enable the use of non-contiguous channel bonding and remove the requirement to use only 20 MHz consecutive channels. The parallelization of temporal overheads clearly improves the throughput" [11]. According to Deng et al, "Via proper radio resource allocation and optimization at scheduler, bandwidth efficient physical layer transmission is enabled; OFDMA PHY creates a new and fundamental challenge in 802.11ax MAC design: the major interaction between MAC and PHY lies in adaptive modulation and coding (AMC) and CCA. Legacy IEEE 802.11 PHY actually transmits data through all data sub-carriers of the single carrier frequency at one time, thus CSMA/CA protocol and its variants can perfectly work. The CCA can also be reliably executed. However, CCA and CSMA/CA face new challenges for OFDMA" since one user occupies a specific carrier frequency, that per se does not necessarily imply non-permissible for other users to access the radio blocks at this carrier frequency [12].

### 2.4.4 Multiple-antenna Technologies

IEEE 802.11ax will continue implementing Single-user MIMO (SU-MIMO) and Downlink MU-MIMO, as found in IEEE 802.11ac. However, it may also include or provide support to:

- MU-MIMO

Enables multiple simultaneous transmissions to different STAs from the AP in the downlink, and from multiple STAs to the AP in the uplink. In the downlink, a challenge for 802.11ax is to reduce the channel sounding overheads using the same approach as 802.11ac, depending on the channel sounding rate and number of STAs. To support MU-MIMO transmissions in the uplink, 802.11ax must also detail how the uplink Channel State Information (CSI) from each STA is obtained by the AP. Introduce a mechanism to signal a group of STAs to simultaneously start a transmission and including techniques to overcome channel calibration and timing issues is also necessary to implement, according to [11]. This involves the design of an efficient mechanism to create groups of STAs with low spatial correlation and close channel quality.

- Massive MIMO

The AP has many more antennas than STAs and uses them to create an identical number of point-to-point links as the number of STAs, according to [12].

- Network MIMO

This approach can be used to minimize the interference among simultaneous transmissions from different APs. Different APs can thus coordinate their transmissions, working as a large array of antennas, reducing the inter-transmission interference and increasing spatial reuse.

# Chapter 3

# Problem Approach and Proposed Solution

In this chapter a new protocol enhancement is presented and explained.

Considering the exposed and proposed previously in this document, it is intended to use ns-3 simulator to assess new mechanisms/techniques foreseen for IEEE 802.11ax, having in view the challenges already referred in this work.

Since, at the time of this writing, there are still no simulation models, in the ns-3 community, for the envisaged MAC layer of future 802.11ax standard, it is a problem when trying to assess the new mechanisms proposed for this amendment.

Thus, the proposed approach is adapting/developing simulation models that would allow the assessment of the new mechanisms, expected to be implemented.

After the new protocol detailing, and to conclude this chapter, a simple example is provided to stand out the differences between this new proposal and the current standard.

## 3.1 Problem Approach and ns-3

There are four categories, which were already seen in the previous chapter, where the new IEEE 802.11ax is expected to develop new solutions at the MAC layer level: Spatial Reuse, Temporal Efficiency, Spectrum Sharing and Multiple-antenna Technologies.

In order to analyze the impact of these enhancements, it is important to have access to a simulation platform that allows the user to simulate different usage scenarios, whether they be simple scenarios (one AP with two or more STAs associated in an infrastruture network) or more complex scenarios (dense WLAN scenarios). A platform allowing an user to simulate packet sending, adapting the new MAC layer enhancements to the simulation model, and then measuring important variables, such as packet loss ratio, throughput, collisions, or error rates, is of the utmost importance.

In order to achieve what is above stated, it is proposed an approach based on using existing simulation models for a network simulation tool (ns-3), models that can be based on existing standards, e.g., the use of the CSMA/CA protocol for the MAC layer on the IEEE 802.11g standard, and adapting the new techniques at the MAC layer level on these models, in order to be possible the simulation of the proposed enhancements.

First, it is important to acquire a good knowledge on how the network simulator operates and then understand where are the source codes of any MAC layer functionality that is to be tested. Simulations based on current standards are to be tested first, so that subsequent work can have a comparison basis and a standard from which to begin the development of the new functions.

The next step is to implement the proposed enhancements at the MAC layer and run simulations again. The new data gathered will then be compared with the previous simulation results obtained with the available current standards. This, per se, will permit a comprehensive analysis on the effects that the new features bring and, eventually, to be considered to the new IEEE 802.11ax.

By the end of this work, it is expected to have a simulation model available that allows an user to test the new enhancements and measure throughput results in several conditions. If this goal is achieved, it can be used as an important contribution for the ns-3 community, which has reckoned the need for a simulation model capable of translating the theoretical improvements described on the state of the art into numerical results, as stated in [16].

An assessment of the new techniques, under discussion, to the MAC layer for the future IEEE 802.11ax standard was made. Among these techniques, some new enhancements for the MAC layer protocols were studied in order to evaluate the probability of their inclusion in the new amendment.

Two protocols were studied in detail to examine the possible enhancements CSMA/CA can go through: CSMA Enhanced Collision Avoidance (CSMA/ECA) [7] and [14] and CSMA with Collision Detection and Avoidance (CSMA/CDA) [15]. After this study it was felt that the development of a new enhancement to CSMA/CA, was still possible, in the context of this dissertation work. In fact, this has led to the proposal described from now on and throughout the rest of the document.

## 3.2 CSMA/CA-Deterministic Backoff

The new proposed protocol focuses on a deterministic backoff generated by the AP and sent to the STA on each ACK frame, after each successful transmission. Thus, the new proposed protocol is named Carrier Sense Multiple Access Collision Avoidance with Deterministic Backoff (CSMA/CA-DB). The CSMA/CA-DB protocol follows the current medium access protocol on the first transmission attempt by an STA, implementing then a few changes in order to improve efficiency, translating this efficiency into throughput gain for STAs and APs in an infrastructure network.

Once the STA is associated it will sense the medium and, if the channel is not busy, it will wait DIFS and then begins the countdown of time slots until it has waited a backoff number of time slots (backoff number: a random number generated by an STA). If the channel is busy, the STA will keep sensing the medium until sensing it free. After SIFS, every STA should expect an ACK from the AP to avoid collisions with the successful STA. This ACK frame has currently the format shown in Figure 3.1.

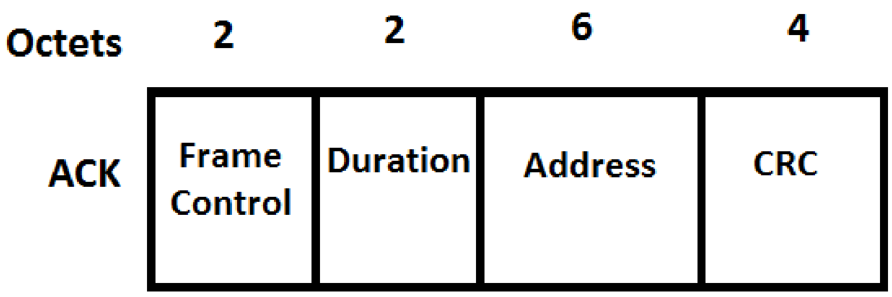


Figure 3.1: Current ACK frame format

The first step of the protocol is to gain access to the channel on the first attempt to send a packet. As explained, this step follows the current CSMA/CA protocol. Once an STA has access to the channel, it can begin the algorithm for sending and receiving packets. One first change compared with CSMA/CA is the fact that the ACK frame will contain one extra byte, with an integer number corresponding to the next backoff number for the next data transmission for this STA. If there are no mode packets to be sent to the AP, this next backoff number will be discarded by the STA. This extra field on the ACK frame corresponds to the number of time slots the STA should wait before transmitting the next DATA frame. This extra byte will be sent on each ACK frame before the CRC field, as shown in Figure 3.2.

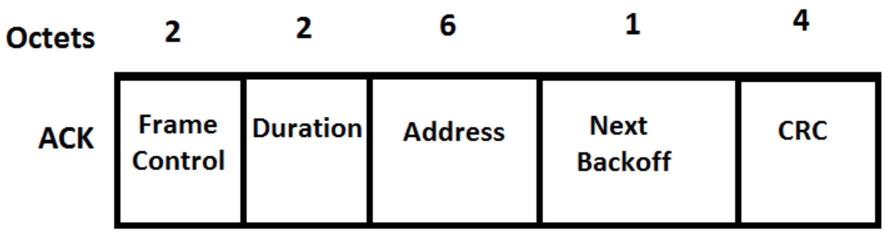


Figure 3.2: New ACK frame format

This represents a small increase of complexity both for the AP and for the STA. The AP keeps a data structure with information about the BSS, in terms of active STAs and each STA has to read its next backoff number, which is included on the ACK frames. The AP is then responsible to provide the STAs with the next backoff number they are going to use, which will make the communication more efficient, avoiding collisions. When the AP receives a packet from an STA, it selects a next backoff number for that particular STA and it sends that information on the respective ACK. To support that decision, the AP keeps a table containing the MAC address of an STA and its corresponding backoff number, which is called "scheduling table". This way, the only way to occur a collision is in the case the first packet to be transmitted by two STAs have the same binary exponential backoff associated, or when other frames are sent to the channel not corresponding to DATA or ACK frames (as seen in previous chapters of this work, there are other frames involved in a WLAN scenario, such as management frames, control frames and beacons).

The scheduling table is a simple structure in table format, containing, in each entry, the STA's MAC address and the next backoff number associated to the next data frame to be sent, being the backoff decremented as the time slots decrease. This way, the AP synchronizes the channel access and has knowledge about when the channel is going to be busy and how to avoid collisions. Table 3.1 shows the content of a scheduling table.

Table 3.1: AP scheduling table

| STA | Next Backoff |
|---|---|
| MAC1 | backoff - 2 |
| MAC2 | backoff - 1 |
| MAC3 | backoff |

As it will be explained below, when the protocol is in its normal operation phase, the backoff value is, in fact, equivalent to the number of active STAs on the channel. However, in a periodic basis, the backoff value assumes, also, the current contention window size.

Explaining in a detailed way, on the first DATA frame received by the AP, the AP will place the STA on the first position of the scheduling table and the backoff will have the value "current contention window". This value indicates the size of the "contention window", which CSMA/CA uses as the interval in which the backoff number can be obtained. In CSMA/CA-DB, the default value for the minimum contention window is 15. If right after this ACK containing number 15 another STA sends to the AP a DATA frame, this MAC address will go to the next position on the scheduling list with the next backoff value corresponding to the first position on the table plus one. The AP is responsible at all times to control the value of the current contention window size. If on the scheduling table there are more entries than half of the contention window size, the AP should double its value. If on the scheduling table there are less entries than a quarter of the contention window value, the AP should reduce its value to the half. If the number of STAs is less than the minimum contention window, when contention for the channel starts, this represents some waste of time, while the first STA waits for the next opportunity to transmit, but at the same time, it allows a big opportunity window for the other STAs to access the channel without generating collisions.

The first DATA frame transmitted receives the current value of the contention window to give an opportunity to other STAs that might want to contend for the channel. After gaining access to the channel, all STAs enter in this cycle controlled by the AP, through the scheduling table. While trying to gain access to the channel, STAs should listen to the medium and in case the channel is busy, they should wait for the ACK from the AP and see what will be the backoff number for the next STA and try to adapt their backoff number in order to avoid a collision. This will reduce the chances of a collision when an STA is trying to transmit for the first time.

When the first STA gets another turn to transmit a DATA frame (15 time slots have passed) that means any STA that did not get into contention will have to wait for another opportunity, since, from now on, the scheduling table has all STAs in sequence for sending their corresponding frames. So after a number of packets sent, that can be determined using the current number of STAs contending for the medium, the AP will open an "opportunity window". That is, the next backoff given by the AP is not last scheduling table position's backoff + 1 but is again the value of the contention window. This will create room for new STAs that might be waiting to gain access to the medium. When any DATA frame transmission fails, the AP will not send any ACK, and this STA will have to wait for an "opportunity window" to get access to the medium. The scheduling table will remain fulfilling its algorithm. When an "opportunity window" is opened, the STAs should select a random backoff number between zero and the last backoff given by the AP, which was sensed by the STAs on the ACK sent by the AP, the current contention window size.

The purpose of this new protocol is to improve the efficiency of the current CSMA/CA protocol. CSMA/CA-DB intends to be an enhanced CSMA/CA for high density scenarios, in which the main focus is to reduce the idleness of the channel, reduce collisions, retransmissions, thus improving the overall throughput on the channel, by giving a deterministic backoff which is the most efficient to each STA.

Since high density scenarios are expected to have many STAs contending for the channel with many DATA frames to be sent and received through the wireless channel, the focus of this document is implement the long term situation, in which many STAs are contending for the channel and have filled queues with DATA packets to be sent. In a real situation, in case mobility and idleness would make the channel have less STAs contending for the channel, the AP and the STAs could go back to CSMA/CA.

The following algorithm demonstrates how CSMA/CA-DB work at STA level. In the pseudo-algorithm, a few variables were defined, expressing some of the protocol parameters. "b" represents the random backoff number generated by the STA to send a DATA packet to the AP. "r" is the current number of retransmissions. "R" is the maximum number of retransmissions allowed. Each retransmission due to packet loss or collisions triggers a new backoff stage. The current backoff stage is given by "s" and the maximum backoff stage permitted by the protocol is given by "S". "next _b" represents the random backoff number for the next transmission read on the ACK frame received from the AP.

```
while the device is on do
    b = U[0, CWmin -1]; // the backoff number;
    r = 0; //the current number of retransmissions ;
    s = 0; //the current backoff stage ;
    next_b = 0; //the backoff number for the next packet ;
    while there is a packet to transmit do
        repeat
            while b>0 do
                wait 1 slot ;
                b = b-1 ;
            end
            Attempt transmission of 1 packet;
            if collision then
                Wait for opportunity window;
                r = r+1 ;
                s = min(s+1,S) ;
                b = U[0,CWmin-1] ;
            else
                read next_b on ACK;
            end
        until (r=R) or (success);
        r = 0 ;
        s = 0 ;
        if success then
            b = next _b ;
            next_b = 0 ;
        else
            Discard packet ;
            Wait for opportunity window;
            b = U[0, CWmin-1] ;
            next_b = 0 ;
        end
    end
    Wait until there is a packet to transmit;
end
```

**Algorithm 1:** CSMA/CA-DB algorithm for STA

Figure 3.3 shows a flowchart for CSMA/CA-DB at the STA:

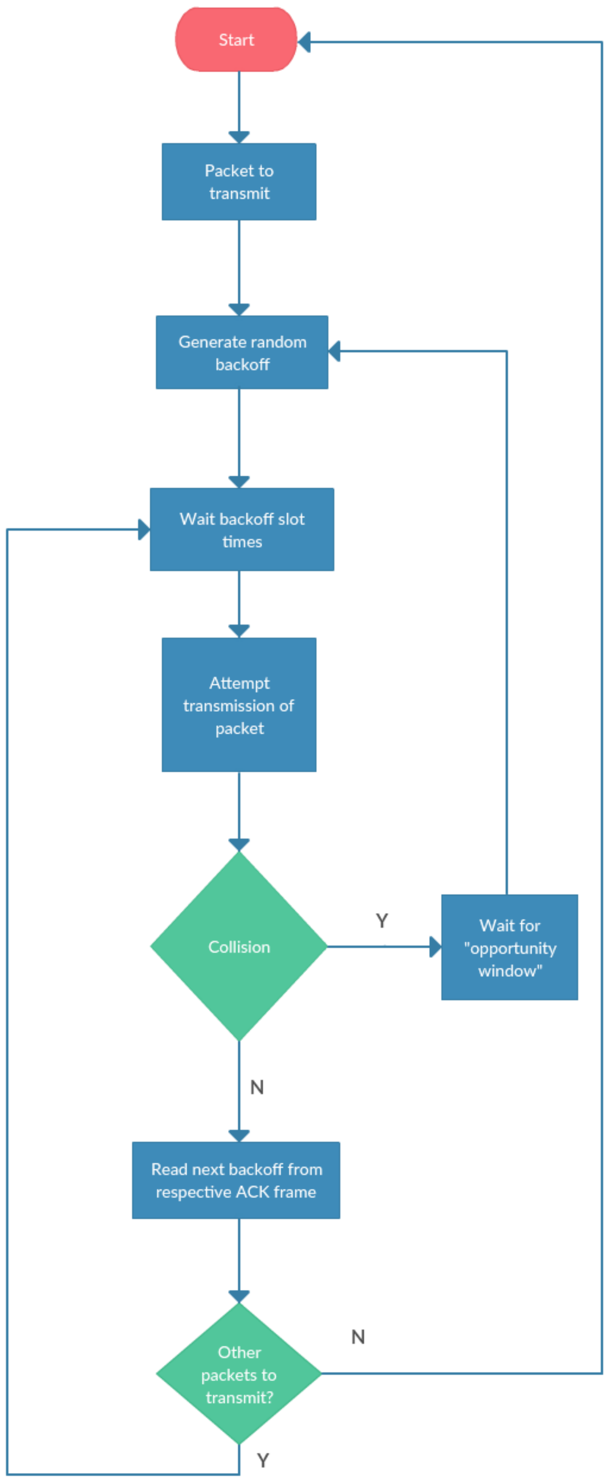

Start
Packet to transmit
Generate random backoff
Wait backoff slot times
Attempt transmission of packet
Collision
Y
Wait for "opportunity window"
N
Read next backoff from respective ACK frame
Other packets to transmit?
N
Y

Figure 3.3: Flowchart for CSMA/CA-DB at STA level

The following algorithm demonstrates how CSMA/CA-DB work at AP level. In the pseudo-algorithm for CSMA/CA-DB for the AP, a few variables were defined, expressing some of the protocol parameters. "cwin" represents the current contention window size. "backoff" is the back-off number that will be sent to the STA on the next ACK. "counter" is a counter to determine how many successful packets have been received by the AP. "max" is the maximum number of oackets received before opening an "opportunity window".

```
while the device is on do
    cwin = 15; //the contention window size ;
    backoff = 0; //the backoff number ;
    counter = 0; //a counter for the number of packets received ;
    max = 100; //the maximum number of packets before opening an opportunity window;
    if packet received then
        counter = counter + 1;
        Read MAC address and place it on the last position of the scheduling table;
        if scheduling table size > cwin/2 then
            cwin = cwin * 2;
        end
        if scheduling table size < cwin/4 then
            cwin = cwin / 2;
        end
        if scheduling table size == 0 and counter == 0 then
            backoff = cwin;
        else
            backoff = last position backoff + 1;
        end
        if counter = max then
            Open opportunity window;
            backoff = cwin;
            counter = 0;
        end
        Send ACK with cwin attached;
    end
    Wait until there is a packet received;
end
```

**Algorithm 2:** CSMA/CA-DB algorithm for AP, with opportunity window being opened after 100 successful packets received

Figure 3.4 shows a flowchart for CSMA/CA-DB at the AP:

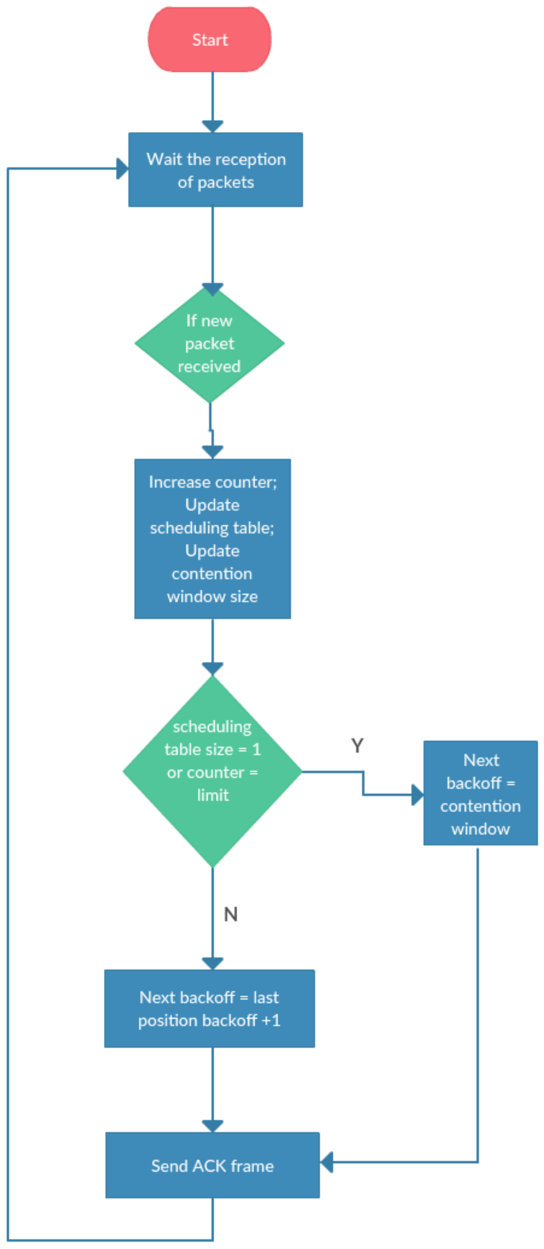

Start
Wait the reception of packets
If new packet received
Increase counter; Update scheduling table; Update contention window size
scheduling table size = 1 or counter = limit
Y
Next backoff = contention window
N
Next backoff = last position backoff +1
Send ACK frame

Figure 3.4: Flowchart for CSMA/CA-DB at AP level

Figure 3.5 shows an example of the medium access mechanism for CSMA/CA while figure 3.6 shows a comparison with the new CSMA/CA-DB MAC protocol. As it can be seen on the two diagrams, CSMA/CA has a randomness associated with it, which carries some disadvantages. In the time sequence diagram, STA E does not have an opportunity to transmit since its DATA frame collided with the DATA frame from STA A, even though STA E had already backoff 11 chosen, nothing hindered STA A from picking the same value. With collisions come a higher backoff contention window for both STAs. The number of time slots for each STA in contention for the channel is also randomly variable. With CSMA/CA-DB, the time while the channel is idle, i.e. the contention period, is deterministic and set to one, while the order of transmission is fair.

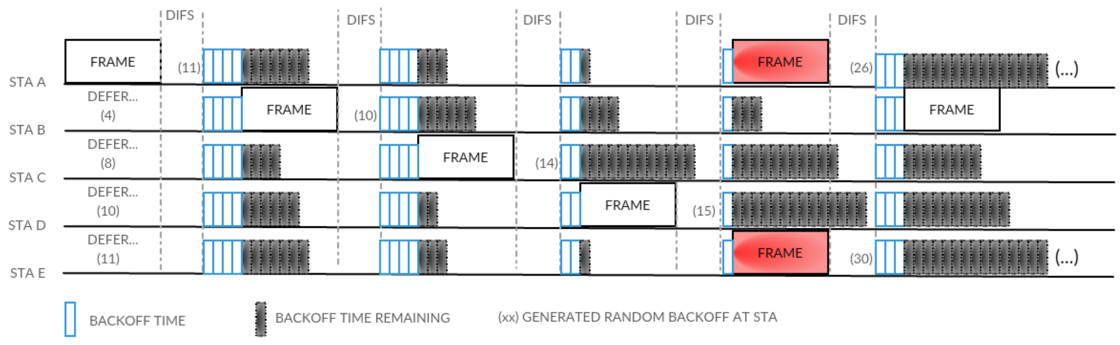


Figure 3.5: CSMA/CA medium access example

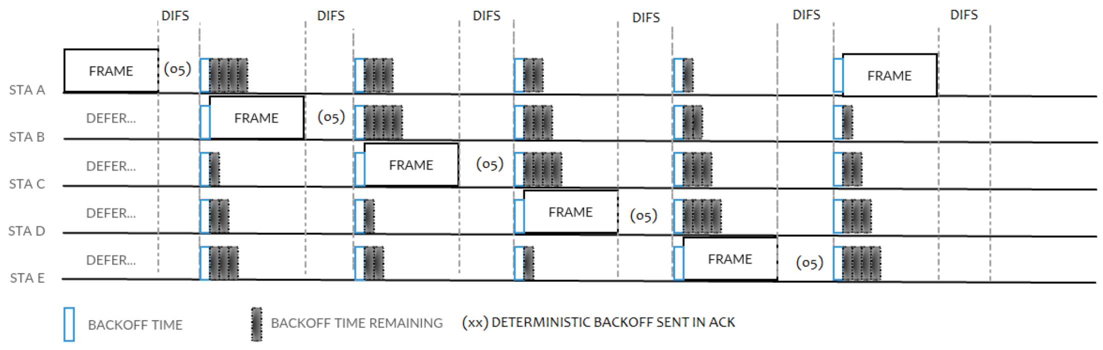


Figure 3.6: CSMA/CA-DB medium access example

## 3.2.1 A Simple Example

As an example of the way the proposed protocol is intended to work, a simple scenario is shown, with one AP and three STAs. STA1 wants to send four data packets, STA2 wants to send three data packets, and STA3 wants to send two data packets. First, an analysis to the way CSMA/CA works will be shown and later compared to the way CSMA/CA-DB works.

Each STA will randomly choose its backoffs for each packet to be transmitted. Table 3.2 shows the packets to be sent by each STA and the random backoffs that will be used hereafter.

For demonstration purposes, all these backoffs were randomly chosen and they are all different. Figure 3.7 shows the first half of the transmission of the packets.

There is a collision between the packet DATA3 of STA1 and the packet DATA3 of STA2. To solve this collision, both STAs will choose a new random backoff, but this time their contention

Table 3.2: Simple example - DATA packets to be sent and the corresponding random backoffs

| STA | Packet | Backoff |
|---|---|---|
| 1 | DATA1 | 6 |
| 2 | DATA1 | 10 |
| 3 | DATA1 | 12 |
| 1 | DATA2 | 9 |
| 2 | DATA2 | 3 |
| 3 | DATA2 | 7 |
| 1 | DATA3 | 2 |
| 2 | DATA3 | 4 |
| 1 | DATA4 | 1 |

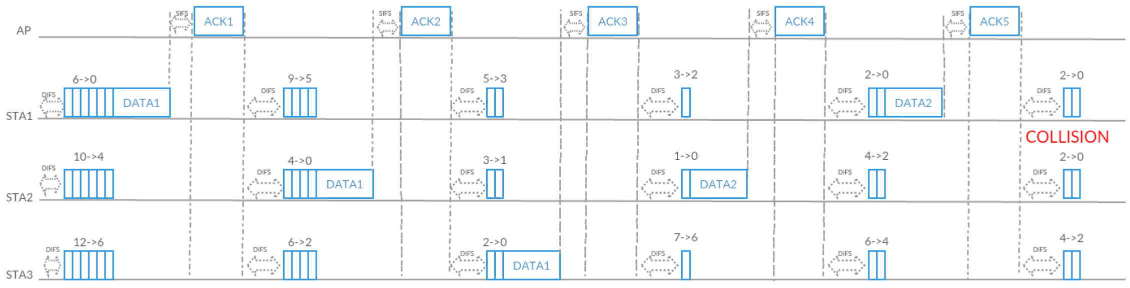


Figure 3.7: Simple example - First part of packet transmission using CSMA/CA

window is 32 in size, instead of 16. So the new backoffs randomly chosen will be 20 for STA1 and 25 for STA2.

Figure 3.8 shows the second half of the transmission of the packets. In the second half, it is possible to see the last packet sent by STA3 and the collision resolution by STA1 and STA2.

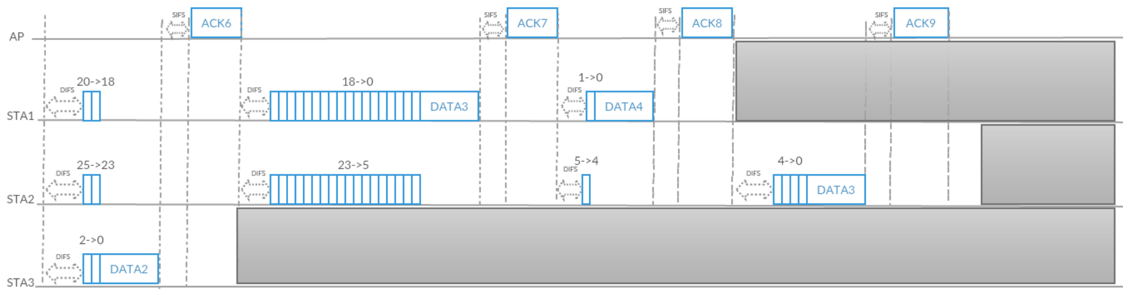


Figure 3.8: Simple example - Second part of packet transmission using CSMA/CA

It is time to compare CSMA/CA-DB approach to packet transmission. In the same scenario described above, with three STAs contending for the medium, with packets to transmit, each STA has randomly chosen its random backoff for the first packet to be sent. Table 3.3 shows the packets to be sent by each STA and the random backoffs. The notation was chosen as "DATAY_X", where Y is the number of the STA and X is the packet number.

Table 3.3: DATA packets to be sent. TBD - To be determined

| STA | Packet | Backoff |
|---|---|---|
| 1 | DATA1_1 | 6 |
| 2 | DATA2_1 | 10 |
| 3 | DATA3_1 | 12 |
| 1 | DATA1_2 | TBD |
| 2 | DATA2_2 | TBD |
| 3 | DATA3_2 | TBD |
| 1 | DATA1_3 | TBD |
| 2 | DATA2_3 | TBD |
| 3 | DATA3_3 | TBD |
| 1 | DATA1_4 | TBD |
| 2 | DATA2_4 | TBD |
| 3 | DATA3_4 | TBD |
| 1 | DATA1_5 | TBD |
| 2 | DATA2_5 | TBD |
| 1 | DATA1_6 | TBD |

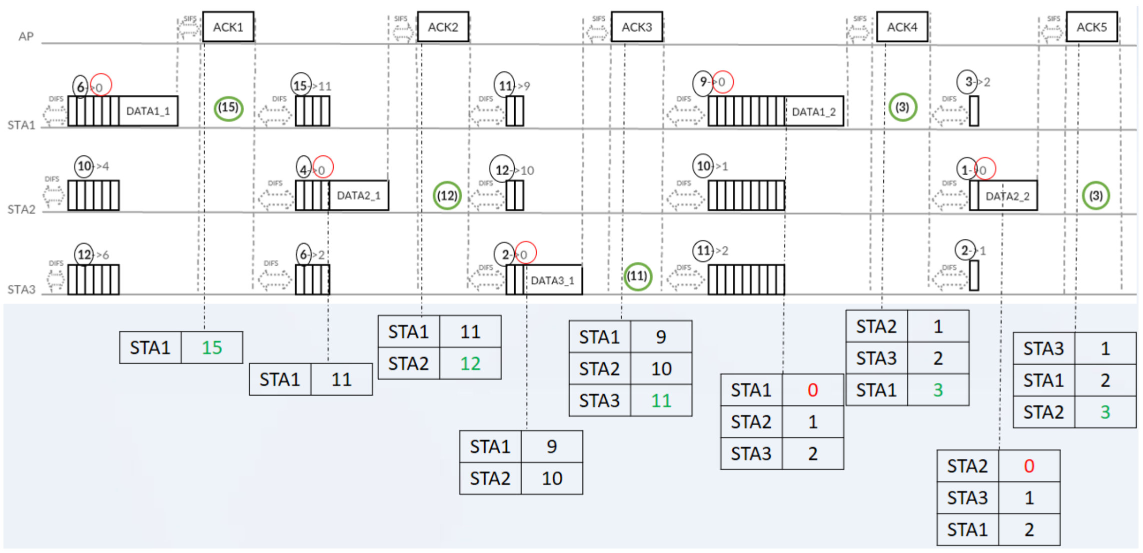


Figure 3.9: Simple example - Initial phase of packet transmission using CSMA/CA-DB

Figure 3.9 shows the first part of the transmission of the packets, the initial phase. This first part of the protocol works exactly as current CSMA/CA. The STAs would have acquired access to the channel and they all would wait DIFS and then sense the channel idle before contending for the channel. Because the smallest random backoff number belongs to packet DATA1_1, after 6 slots of time STA1 sends its first packet. As described before, the AP will send the next random backoff number on the ACK, which is 15 (initial contention window size). After waiting SIFS, the AP sends an ACK after filling the scheduling table with this information, as in Table 3.4.

Table 3.4: Scheduling table when ACK1 is sent after the first packet sent by STA1

| MAC | Backoff |
|---|---|
| STA1 | 15 |

The next packet sent is DATA2_1, which waits 4 time slots before sending the packet. Here, the AP updates its scheduling table accordingly, as in Table 3.5, and sends STA2 its next backoff.

Table 3.5: Scheduling table when ACK2 is sent after the first packet sent by STA2

| MAC | Backoff |
|---|---|
| STA1 | 11 |
| STA2 | 12 |

The backoff given to STA2 will not be 15 this time. Following the protocol, the next backoff is given as the last position of the scheduling table plus one. The next packet is sent by STA3 and the process is repeated with the AP updating the scheduling table and sending ACK3, as seen in Table 3.6.

Table 3.6: Scheduling table when ACK3 is sent after the first packet sent by STA3

| MAC | Backoff |
|---|---|
| STA1 | 9 |
| STA2 | 10 |
| STA3 | 11 |

If any other STA would want to enter in the contention for the medium, there would still be 9 time slots available before STA1 attempts the transmission of the next packet.

Immediately before STA1 sends the second DATA packet, the scheduling has the contents shown in Table 3.7.

Table 3.7: Scheduling table before STA1 sends packet DATA2

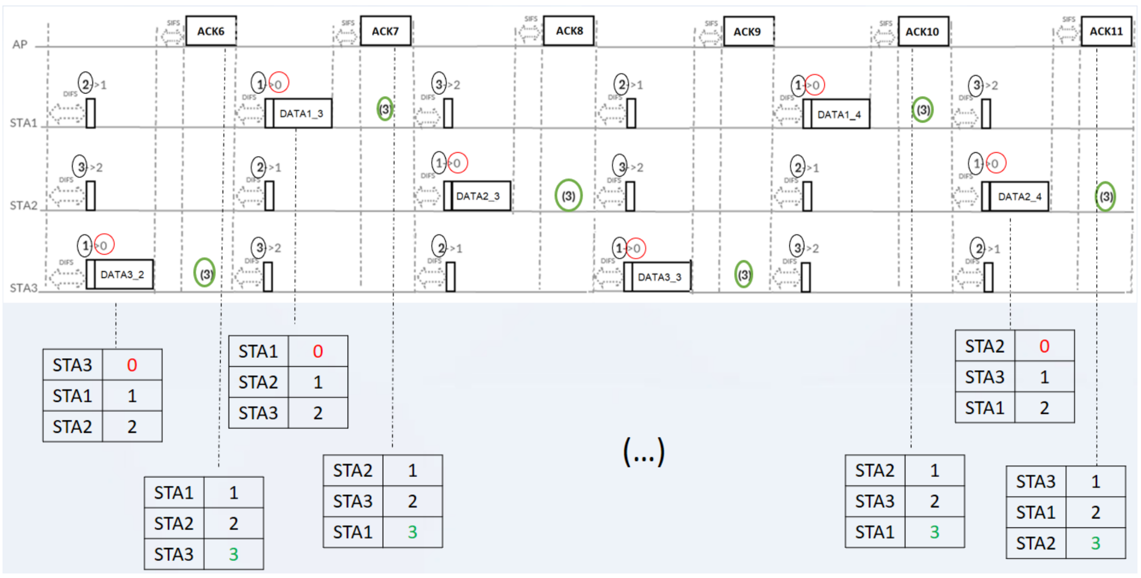
| MAC | Backoff |
|---|---|
| STA1 | 0 |
| STA2 | 1 |
| STA3 | 2 |

The cycle is now complete and when STA1 sends DATA1_2 packet, the AP will give STA1 the next backoff according to the formula "last position on the scheduling table" + 1. After this packet is sent, ACK4 will contain number 3, as shown in Table 3.8.

Table 3.8: Scheduling table after the second packet is sent by STA1

| MAC | Backoff |
|---|---|
| STA2 | 1 |
| STA3 | 2 |
| STA1 | 3 |

Figure 3.10 shows the second part of the transmission of the packets, the normal phase. This was the phase that was simulated using ns-3.



| | |
|---|---|
| STA3 | 0 |
| STA1 | 1 |
| STA2 | 2 |

| | |
|---|---|
| STA1 | 1 |
| STA2 | 2 |
| STA3 | 3 |

| | |
|---|---|
| STA1 | 0 |
| STA2 | 1 |
| STA3 | 2 |

| | |
|---|---|
| STA2 | 1 |
| STA3 | 2 |
| STA1 | 3 |

| | |
|---|---|
| STA2 | 1 |
| STA3 | 2 |
| STA1 | 3 |

| | |
|---|---|
| STA2 | 0 |
| STA3 | 1 |
| STA1 | 2 |

| | |
|---|---|
| STA3 | 1 |
| STA1 | 2 |
| STA2 | 3 |

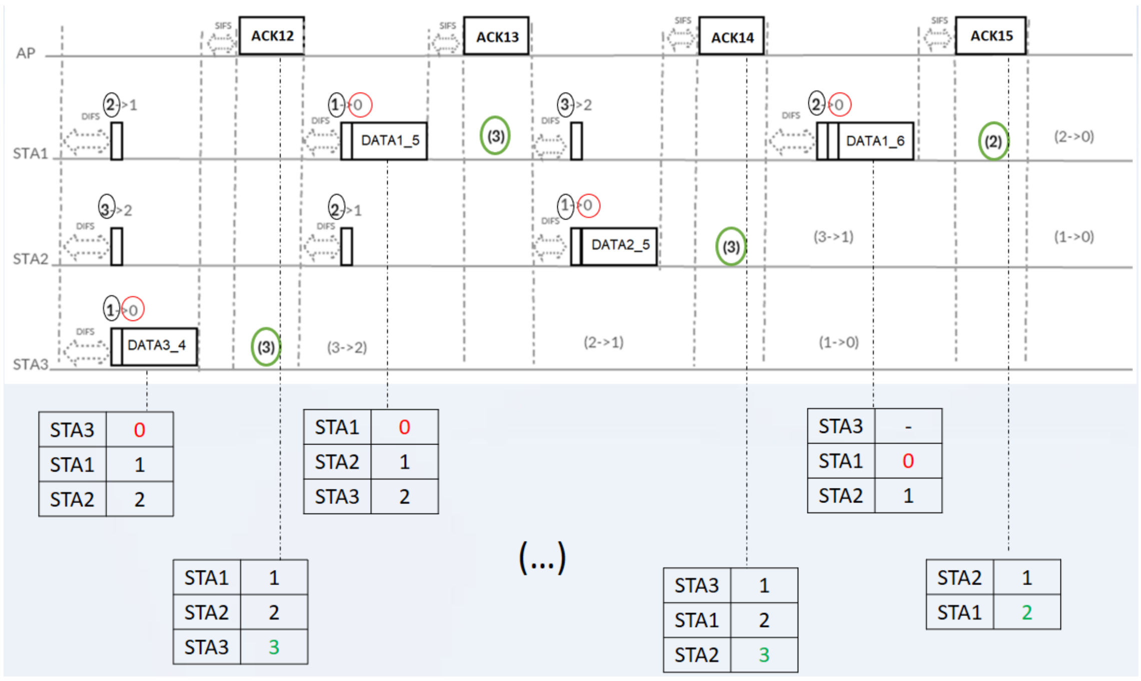
Figure 3.10: Simple example - Second part of packet transmission using CSMA/CA-DB

In the normal phase, it is easy to detect the main advantages of CSMA/CA-DB: the time slots to wait before a packet transmission are kept at a minimum (one time slot), which greatly reduces the idleness of the channel. The order in which the STAs transmit can also be seen as fair.

Figure 3.11 shows the last part of the transmission of the packets, the final phase, where STAs finish their transmissions. After all transmissions, the backoffs given can be summarized on Table 3.9.

After this step, the process is back to the beginning, where some STAs have acquired access to the channel and the AP is ready to receive packets.

It is interesting to check the results of this simple implementation in terms of number of total time slots used compared with current CSMA/CA protocol. In order to transmit the 9 data packets, CSMA/CA would have used 42 time slots, mainly due to a collision and the idleness of the channel. To solve the collision problem, none of the STAs would have transmitted their packets, both



| | |
|---|---|
| STA3 | 0 |
| STA1 | 1 |
| STA2 | 2 |

| | |
|---|---|
| STA1 | 1 |
| STA2 | 2 |
| STA3 | 3 |

| | |
|---|---|
| STA1 | 0 |
| STA2 | 1 |
| STA3 | 2 |

| | |
|---|---|
| STA3 | 1 |
| STA1 | 2 |
| STA2 | 3 |

| | |
|---|---|
| STA3 | - |
| STA1 | 0 |
| STA2 | 1 |

| | |
|---|---|
| STA2 | 1 |
| STA1 | 2 |

Figure 3.11: Simple example - Final phase of packet transmission using CSMA/CA-DB

Table 3.9: Backoffs after all transmissions. * using CSMA/CA-DB

| STA | Packet | Backoff |
|---|---|---|
| 1 | DATA1_1 | 6 |
| 2 | DATA2_1 | 10 |
| 3 | DATA3_1 | 12 |
| 1 | DATA1_2 | 15 * |
| 2 | DATA2_2 | 12 * |
| 3 | DATA3_2 | 11 * |
| 1 | DATA1_3 | 3 * |
| 2 | DATA2_3 | 3 * |
| 3 | DATA3_3 | 3 * |
| 1 | DATA1_4 | 3 * |
| 2 | DATA2_4 | 3 * |
| 3 | DATA3_4 | 3 * |
| 1 | DATA1_5 | 3 * |
| 2 | DATA2_5 | 3 * |
| 1 | DATA1_6 | 2 * |

would have had to choose another random backoff number, plus the contention window would be increased for twice the minimum size for both STAs, which could result in longer time to wait for both STAs and there would still be a possibility of new collisions depending on the random number chosen in both cases. Another interesting detail is the order in which packets are sent to the network, which is random. There is no guarantee that some STA, though it may be the first to contend for the channel will see its packet being the first to be transmitted.

With CSMA/CA-DB, collisions were avoided which, as seen before, is currently one of the main drawbacks in dense wireless scenarios for CSMA/CA and the order of transmission follows a fair first-in, first-out fashion. As seen in this example, in the long run, the time an STA has to wait before being able to transmit the next packet tends to an optimal value, variable and dependent on the number of STAs contending for the medium. With 33 time slots, CSMA/CA-DB was able to transmit 15 DATA packets, while CSMA/CA was only capable of transmitting 9 DATA packet in 42 slots.

It is expected that these theoretical improvements will be proven in the next chapters of this work. With an implementation in ns-3, explained in the next chapter, it is intended to apply the changes in the source code of the network simulator, and, in chapter 5, it is expected that these changes show a practical improvement in terms of throughput.

# Chapter 4

# Model Implementation in ns-3

In this chapter, ns-3 and its building blocks are presented in detail, specially those involving Wifi simulations. In order to test and evaluate the CSMA/CA-DB protocol enhancement, as part of the MAC layer protocol of a wireless channel, a simulation software was chosen to test, first, the current CSMA/CA features. After this initial phase, it was necessary to adapt the software modules in order to include the new protocol enhancements, and, subsequently, create the simulation scenarios, run the tests and confirm, in case of success, the expected improvements. "ns-3" was chosen as the simulation software to be used on this implementation phase, because it is one of the most powerful network simulators, it is open source and INESC TEC, the Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência, has also a group of researchers working with ns-3 and contributing to the ns-3 community.

The chapter contains all the implementation details that pertain to ns-3 classes and code needed to apply CSMA/CA-DB into ns-3 core code.

## 4.1 ns-3 Simulation Software Basics

### 4.1.1 Classes

ns-3 is a discrete-event network simulator written in C++ with diverse modules.

Each one of the available modules includes many classes; some of those are mandatory for a simulation, while others are used to give specific features on the system. Below, some of the available classes are briefly presented:

- Node (network module) - provides methods for managing representations of computing devices in simulations.
- Application - starts/stops at specific times in a simulation and is installed in a node. An application generates traffic with various rates and protocol types. Packets are sent through

sockets to other destinations. One subclass is the "OnOffApplication" which is used for generating Transmisssion Control Protocol (TCP) or User Datagram Protocol (UDP) traffic from one source to a single destination according to an OnOff pattern.

- Channel - models a medium such as wire ("CsmaChannel"), wireless ("WifiChannel"), etc., that is needed to transmit data.
- NetDevice - a Network Interface Card (NIC) installed in a node, in order to communicate with other nodes via a channel. "CsmaNetDevice", "WifiNetDevice", "PointToPointNetDevice" have different features and are used on different systems.

### 4.1.2 Simulation Components

To establish the simulation scenario to be tested, the following components should be defined:

1. Nodes (STAs, APs, etc. on a network)
2. Channel (medium used to share packets between nodes)
   (a) PointToPoint Channel
   (b) Csma Channel
   (c) Wifi Channel
3. NetDevice (hardware installed on each node to support the type of channel)
4. Protocols (what the Nodes use to share data, using "InternetStackHelper")
   (a) TCP
   (b) IP
   (c) UDP
5. IP Addresses (using "Ipv4AddressHelper" or "Ipv6AddressHelper")
6. Applications (to generate packets)
   (a) "UdpEchoClientApplication"
   (b) "UdpEchoServerApplication"
   (c) "onoffapplication"

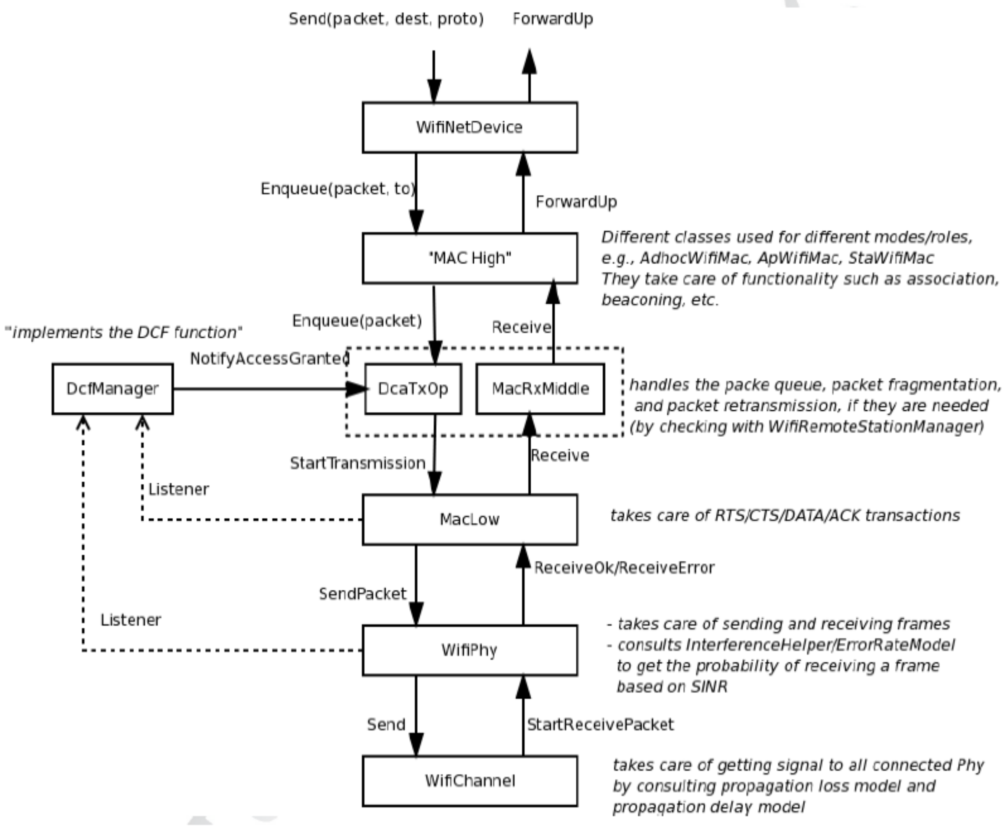

Send(packet, dest, proto)
ForwardUp
WifiNetDevice
Enqueue(packet, to)
ForwardUp
"MAC High"
Different classes used for different modes/roles, e.g., AdhocWifiMac, ApWifiMac, StaWifiMac They take care of functionality such as association, beaconing, etc.
Enqueue(packet)
Receive
"implements the DCF function"
NotifyAccessGranted
DcfManager
DcaTxOp
MacRxMiddle
handles the packe queue, packet fragmentation, and packet retransmission, if they are needed (by checking with WifiRemoteStationManager)
StartTransmission
Receive
Listener
MacLow
takes care of RTS/CTS/DATA/ACK transactions
SendPacket
ReceiveOk/ReceiveError
Listener
WifiPhy
- takes care of sending and receiving frames
- consults InterferenceHelper/ErrorRateModel to get the probability of receiving a frame based on SINR
Send
StartReceivePacket
WifiChannel
takes care of getting signal to all connected Phy by consulting propagation loss model and propagation delay model

Figure 4.1: Wifi module architecture in ns-3 [8]

## 4.2 ns-3 Wifi Module

In ns-3, the IEEE 802.11 standard is implemented using the WiFi module.
This module has the following features:

- DCF implementation (Basic and RTS/CTS)
- 802.11 a/b/g
- QoS support (EDCA only, part of 802.11e)
- Some 802.11n features
- Infrastructure and ad hoc mode
- Rate adaptation algorithms
- Mobility models

The Wifi module architecture, shown in Figure 4.1 contains the PHY layer model, the MAC Low model (which implements DCF and EDCA), the MAC High model and the rate control algorithms.

### 4.2.1 MAC High

The MAC High model contains currently three models: "AdhocWifiMac", "ApWifiMac" (beacon, association by STAs) and "StaWifiMac" (association based on beacons). These models inherit from "RegularWifiMac", which contains QoS and non-QoS support.

### 4.2.2 MAC Low

This layer has three components: "MacLow" (RTS/CTS/DATA/ACK transactions); "DcfManager" (implementing the DCF); "DcaTxop" and "EdcaTxopN" (one for NQoS, and the other for QoS, plus packet queuing and fragmentation/retransmissions).

### 4.2.3 WiFi Rate Adaptation

Supports both low-latency and high-latency. The supported rate adaptation algorithms include a partial list of algorithms used in real devices, such as "ArfWifiManager" (default), "OnoeWifiManager" or "ConstantRateWifiManager", and others presented in the literature, such as "IdealWifiManager", "AarfWifiManager" or "AmrrWifiManager".

### 4.2.4 Wifi PHY

This is the layer working at the packet level. There are different models available such as "PhySimWifi" (symbol-level, too slow for practical use), "YansWifiPhy" (default) and "SpectrumWifiPhy".

### 4.2.5 Error Rate Models

There is one 802.11b model validated with Clear Channel experiments and two OFDM models: the YANS model (somehow optimistic) and the NIST model (which better match with experimentation) [17].

## 4.3 ns-3 Wifi Transmission and Reception

Figure 4.1 showed the different ns-3 classes involved in a Wifi transmission and reception of packets. In order to transmit a packet from one node to another, ns-3 uses several functions implementing the different protocol layers.

Having a good knowledge on these steps is very important, if any changes are to be made on the CSMA/CA protocol.

### 4.3.1 Transmission

The "UpperLayer" (the application layer, which generates a data packet) sends the packet to the "WifiNetDevice::Send()" which is responsible to convert the IP address into MAC and adds the Logical Link Control (LLC) header.

Depending on which "WifiNetDevice" was called, the Mac class is named, either "AdhocWifiMac::Enqueue()", "ApWifiMac::Enqueue()" or "StaWifiMac::Enqueue()". These are responsible to set the Mac header and estimate the PHY modes (i.e. data rates) supported by the destination node.

The next level is at the "DcaTxop::Queue()" which inserts the new packet into a queue through the function "WifiMacQueue::Enqueue()".

Next, "DcaTxop::StartAccessIfNeeded()" requests access to the DCF Manager in case there is no packet pending (if variable "m_accessRequest" = true). If there is another packet pending, "DcfManager::RequestAccess()" will signal a collision through "DcaTxop::NotifyCollision()" which will select a random backoff number and try to access again.

In case of EDCA protocol (IEEE 802.11e), "DcfManager::DoGrantAccess()" will check the internal collision among the 4 different queues provided by the protocol for QoS support.

Once access is granted, "DcfState::NotifyAccessGranted()" sets the variable "m_accessRequest" as false and notifies that channel access is now granted, which will trigger "DcaTxop:: NotifyAccessGranted ()", which will extract one packet from the queue and check if this packet is multicast, if requires fragmentation or RTS.

The next step is at the "MacLow" class, through "MacLow::StartTransmission()" that will call the function "SendRtsForPacket()", in case RTS is required, or otherwise calls "SendDataPacket()", which will estimate the duration of the transmission, manage the ACK reception and/or the next transmission (for block ACK or fragmentation).

"MacLow::ForwardDown()" tosses the packet to the PHY layer, calling "YansWifiPhy:: SendPacket ()", which will change the current PHY state to Tx mode (and this will eventually inform DCF that a transmission is starting) and send the packet to the channel layer, through "YansWifiChannel::Send()". This function estimates the receive power and sets the propagation delay to trigger "YansWifiChannel::Receive()" on the neighbor nodes after a short period.

### 4.3.2 Reception

Function "YansWifiChannel::Receive()" is triggered after the elapsed propagation delay, which was set by the sender through "YansWifiChannel::Send()"

"YansWifiChannel::StartReceivePacket()" calculates the current interface level and it drops the received packet if the state of PHY is not idle or the received power is below than a threshold. It sets the delay to trigger "YansWifiChannel::EndSync()" after the packet transmission.

"YansWifiChannel::EndSync()" estimates the packet error rate (PER) from the SNIR. If a random value is smaller than the PER value, the packet is dropped.

"MacLow::ReceiveOk()" is triggered next and if the packet is an RTS packet, it schedules "MacLow::SendCtsAfterRts()"; if it is CTS, it schedules "MacLow::SendDataAfterCts()"; if it is Data, forwards the packet to the upper layer; if the destination is this node, it schedules "MacLow :: SendAckAfterData ()" and processes the received ACK.

"MacRxMiddle::Receive()" is the next function to be called and it checks if the received packet is a duplicated one with a sequence number.

Depending on which "WifiNetDevice" was called, the Mac class is named, either "AdhocWifiMac::ForwardUp()", "ApWifiMac::ForwardUp()" or "StaWifiMac::ForwardUp()". These are responsible to forward the received packet to the upper layer. "WifiNetDevice::ForwardUp()" is called to forward the packet to the upper layer and, unless the destination is other node, it forwards the received packet to the upper layer.

## 4.4 Simulating with ns-3

### 4.4.1 Scenario replication and random numbers

Simulating any network scenario involves the execution of several simulations keeping the same conditions, such as protocols used, node positions, data rates, bandwidth, in a way that a certain confidence interval can be achieved after a certain amount of simulations, and thus making it possible to replicate the same results, under the same simulation conditions, in future experiments.

As the ns-3 Manual refers regarding random variables [18], ns-3 uses a pseudo-random number generator which means that if there is any randomness in the simulation, each run of the program will yield identical results unless the seed and/or run number is changed. There are two options to obtain randomness across multiple simulation runs, either by setting the seed differently or setting the run number differently. A seed can be set using "ns3::RngSeedManager::SetSeed()" at the beginning of the program; similarly, setting a run number with the same seed can be done using "ns3::RngSeedManager::SetRun()".

To achieve useful replications of the same network scenario, a few measures have to be taken. A network simulation is based on generating pseudo-random numbers, e.g., from the random transmission beginning times for different traffic flows, to the random backoff numbers generated on STAs, before trying to transmit packets. As stated on the ns-3 Manual [18], "users need to be concerned with a few issues, such as the seeding of the random number generator and whether a simulation outcome is deterministic or not, how to acquire different streams of random numbers that are independent from one another, and how long it takes for streams to cycle." The cycle of the random number generator is the amount of numbers it can generate before starting to repeat itself. One way to generate random results is using one different seed for each simulation scenario, and a different run number each time the same simulation is run, in order to compute statistics on the independent simulations. In fact, the only way to guarantee that several simulations for the same scenario have no correlation between their results is setting the same seed and use different runs.

As the Random Variables Manual of ns-3 states [18], ns-3 uses MRG32K3a random number generator from Pierre L'Ecuyer which provides $1.8 \times 10^{19}$ independent streams of random numbers, each of which consists of $2.3 \times 10^{15}$ substreams. Each substream has a period of $7.6 \times 10^{22}$. This way, the largest simulation scenario will have to obey to this substream period to obtain valid results.

### 4.4.2 Simulation Results Statistics

In the simulations presented throughout this work, the statistics were gathered using the tool "Flow Monitor" which can be easily configured on any simulation and can output data, regarding the TCP or UDP flows, on a XML file. Flow Monitor will classify packets in four points, as stated on the ns-3 Models documentation [19]: when a packet is sent, when a packet is forwarded, when a packet is received, and when a packet is dropped. Since the packets are tracked at IP level, any retransmission caused by Layer 4 protocols (e.g., TCP) will be seen by the probe as a new packet.

A few statistic variables are outputed to the XML file [19]:

- "timeFirstTxPacket" (when the first packet in the flow was transmitted);
- "timeLastTxPacket" (when the last packet in the flow was transmitted);
- "timeFirstRxPacket" (when the first packet in the flow was received by an end node);
- "timeLastRxPacket" (when the last packet in the flow was received);
- "delaySum" (the sum of all end-to-end delays for all received packets of the flow);
- "jitterSum" (the sum of all end-to-end delay jitter (delay variation) values for all received packets of the flow, as defined in RFC 3393);
- "txBytes", "txPackets" (total number of transmitted bytes / packets for the flow);
- "rxBytes", "rxPackets" (total number of received bytes / packets for the flow);
- "lostPackets" (total number of packets that are assumed to be lost (not reported over 10 seconds) - this is important to remember when running simulations where the total simulation time is under 10 seconds);
- "timesForwarded" (the number of times a packet has been reportedly forwarded);
- "delayHistogram", "jitterHistogram", "packetSizeHistogram" (histogram versions for the delay, jitter, and packet sizes, respectively);
- "packetsDropped", "bytesDropped" (the number of lost packets and bytes, divided according to the loss reason code).

These variables are used to generate statistics and graphs to help drawing conclusions about the scenarios, current protocol performances and new solution performances.

## 4.5 New Protocol Implementation (CSMA/CA-DB)

The CSMA/CA-DB protocol implementation was developed using ns-3 version 25, changing the current source code to implement the new protocol. Implementing the new protocol was a very important part of the work and consisted of the following six steps:

1. Add an extra field on ACK frames;

2. Read ACK frames meant to other STAs (promiscous mode);
3. Analyze how the random backoff is calculated in the source code of the Wifi module;
4. Change the random backoff into a deterministic backoff;
5. Create a data structure to keep information updated;
6. Coordinate the current backoff values of all STAs in a BSS.

### 4.5.1 Add an extra field on ACK frames

For this feature implementation, at this time, a global variable was created where the AP could place the next random backoff and the STA would read the value from that global variable, instead of sending an actual extra byte on the ACK, for time frame reasons. However, a simple experiment was run where the AP would have a packet to send to the STA. At the "MacLow" layer, using "APWifiMac" class, an integer value was added to the packet before the Mac Trailer and this packet would then be forwarded to the lower layers for sending. At the STA, this value would be read by the STA, using "StaWifiMac" class. This sending and receiving of an extra field was successful. As future work, it remains to be tested the impact of an extra byte on the ACK frame in a dense scenario. One way to avoid this impact would be to increment the data rate at which the ACK frames are sent.

### 4.5.2 Read ACK frames meant to other STAs

Using Class "RegularWifiMac", a simple way to implement this protocol feature, which is important while an STA is trying to access the medium for the first time or is waiting for an "opportunity window", is using the function "Receive (Ptr<Packet> packet, const WifiMacHeader *hdr)":

Listing 4.1: How to read ACK meant to other STAs

```
void
RegularWifiMac::Receive
(Ptr<Packet> packet, const WifiMacHeader *hdr)
{
   NS_LOG_FUNCTION (this << packet << hdr);

    Mac48Address to = hdr->GetAddr1 ();
    Mac48Address from = hdr->GetAddr2 ();


    if (to != GetAddress ())
    {
        //call function to read ACK from AP
        //no other tasks needed at this time
        return;
    }
```

This information would then be used by the STA to change its current backoff value in a way that collisions could be avoided when first trying to gain access to the medium, or to check when the AP is opening an "opportunity window".

### 4.5.3 Detect where the random backoff is calculated

Class "DcaTxop" implements the packet fragmentation and retransmission in ns-3. It uses the classes "MacLow" and "DcfManager" to send packets and decide when to send them. Packets are stored in a "WifiMacQueue" until they can be sent. Every packet is retransmitted until it is either successfully transmitted or it has been retransmitted up until the "ssrc" or "slrc" thresholds.

Every time a random backoff value is needed, "DcaTxop" uses a pseudo-random number generator, with its assigned streams, to generate a new integer value between 0 and the current contention window value:

Listing 4.2: Generation of a random backoff number

```
m_dcf->StartBackoffNow (m_rng->GetNext (0, m_dcf->GetCw ()));
```

This function is called in several places by the "DcfManager" class, in "RequestAccess" where it requests access to the channel and generates a backoff in case there is a collision, i.e. the current number of backoff slots is zero and the medium is busy. "AccessTimeout" does a backoff update in case of a timeout for access waiting. The backoff is updated when a reception or transmission is started, through "NotifyRxStartNow" and "NotifyTxStartNow". "NotifyMaybeCcaBusyStartNow" is another method that updates the backoff value and it is called to notify the DCF that a CCA period has started. "NotifyWakeupNow" updates the backoff value to notify the DCF that the device has resumed from sleep mode. The other two calls are related to the virtual carrier sense, NAV, with "NotifyNavResetNow" and "NotifyNavStartNow". This will allow an STA to postpone the backoff countdown until the current packet sensed on the medium is transmitted.

### 4.5.4 Change the random backoff into a deterministic backoff

The function shown above is called in several places by "DcaTxop":

- "DcaTxop::DoInitialize ()"

- "DcaTxop::NotifyCollision ()"
- "DcaTxop::MissedCts ()"
- "DcaTxop::MissedAck ()"
- "DcaTxop::EndTxNoAck ()"
- "DcaTxop::GotAck ()"

"DcaTxop::DoInitialize ()" is called to initialize the object, and is called once, the same way as "DcaTxop :: DoDispose()" is also called only once. "DcaTxop::NotifyCollision()" is called to notify the DCF that a collision has occured. This will important to manage collisions according to the new protocol. "DcaTxop::MissedCts()" is called when a CTS timeout has occurred. As per this document, CSMA/CA-DB will not be implemented using RTS/CTS protocol. As seen in section 2.2, RTS/CTS protocol, though intended to solve the "hidden node" problem, introduce an overhead in the protocol. "DcaTxop::MissedAck ()" notifies a missed ACK frame. "DcaTxop :: EndTxNoAck ()" is called when a transmission that does not require an ACK has completed. "DcaTxop::GotACK ()" is callen when an ACK is received.

To implement CSMA/CA-DB a new function was developed, named "DcaTxop :: WhatIsMyBackoff (const WifiMacHeader &hdr)":

Listing 4.3: New function to generate deterministic backoff value

```
uint32_t WhatIsMyBackoff (const WifiMacHeader &hdr);
```

This function returns an integer value, the deterministic backoff for the current transmission opportunity, and takes as argument the "WifiMacHeader" for the current packet, as a constant parameter (in order to keep integrity over the different layers of the network simulator). This header contains important information to the protocol, such as the MAC address of the sender, the MAC address of the receiver (which is important in a multi BSS scenario to detect in which BSS this packet is meant to be transmitted), and contains information about what kind of packet this is. With CSMA/CA-DB only DATA and ACK packets are being processed by the new protocol. The other packets use the current CSMA/CA protocol.

Function "DcaTxop::WhatIsMyBackoff (const WifiMacHeader &hdr)" follows certain steps in order to implement CSMA/CA-DB protocol:

1. Check if current packet to send is a DATA packet. If not use default random number generator;
2. Check if scheduling table for this BSS (using the MAC addresses on the "WifiMacHeader") is empty or not;
3. Remove any entries of the scheduling table that have a zero or negative backoff value (packets being sent or already sent);

4. Fill the scheduling table with the information for the current packet (MAC address, table size, and time stamp);

5. Update contention window value accordingly;

6. Give backoff value according to the protocol: if table entry immediately above the current packet has a positive integer has backoff value "x", give "x+1"; if the table has only one entry, give "contention window";

7. Return backoff value.

### 4.5.5 Create a data structure to keep information updated

The scheduling table is a C++ struct that contains a few variables needed for the protocol. A global variable called "m _SchedulingTable" was created to keep the information of each BSS. This variable is a vector of struct, each of which containing the information of a BSS.

The struct was inserted in a header file named "schedulingtable.h", which had to be included in the wscript of the Wifi module of ns-3, and built with the rest of the Wifi module in order to work with the rest of the network simulator and its classes.

Listing 4.4: Scheduling Table contents

```
typedef struct schedulingtable
{
    ns3::Mac48Address iBSS; //MAC address of AP
    std::vector<uint32_t> beb; // vector of backoff values
    //MAC addresses of STAs
    std::vector<ns3::Mac48Address> address;
    uint32_t cwCurrent; // current contention window value
    // time at which a backoff value was given
    std::vector<ns3::Time> backoffStart;
} SchedulingTable;
```

### 4.5.6 How to coordinate the current backoff values of all STAs in a BSS

This was the biggest challenge in the protocol implementation. Whenever a time slot has passed, the scheduling table has to be updated for the algorithm to work, and the current backoff value for all STAs has to see its value decreased accordingly. ns-3 already has mechanisms in place to perform these tasks. Every time a time slot passes, the simulator core will go through every "DcfManager" object to update the backoff value.

Function "DcfState::UpdateBackoffSlotsNow (uint32 _t nSlots, Time backoffUpdateBound)" is responsible for updating the number of time slots left before the STA attempts a packet transmission. The code for this function is shown below:

Listing 4.5: Code of function responsible for updating backoff value

```
void
DcfState::UpdateBackoffSlotsNow (uint32_t nSlots,
    Time backoffUpdateBound)
{
    m_backoffSlots -= nSlots;
    m_backoffStart = backoffUpdateBound;
}
```

This function will decrease the number of time slots that have passed, according to the simulator time by "nSlots" and the time for the last update will be changed to "backoffUpdateBound".

This function will now be responsible for searching the scheduling table and, when the entry is found for the current packet, it will update the scheduling table information, decreasing "nSlots" to the backoff value for this packet and also updating the time when the backoff value was given to "backoffUpdateBound".

# Chapter 5

# Simulation Results and Analysis

In this chapter, the results from simulations are grouped in three categories: channel capacity, simple BSS scenario and building scenario.

## 5.1 Simulation Scenarios For Dense WLANs

In order to simulate the new enhancements proposed by the State of the Art literature, IEEE proposed different scenarios for the HEW, known as the "TGax Simulation Scenarios" [20]. The TGax establishes 4 different scenarios to simulate high efficiency WLANs: a residential building, an enterprise scenario, an indoor small BSS hotspot, and an outdoor large BSS hotspot.

The Residential scenario consists of an apartment building having, for instance, 10m x 10m ($100m^2$) apartments in a multi-floor building, with ten STAs per AP, in an unmanaged WLAN network, intending to replicate a home traffic model.

In the study developed in this work, two simulation scenarios were developed to test the enhancements proposed by the new CSMA/CA-DB MAC layer protocol. The first scenario is called "simple BSS" with one AP and one or more STAs sending a constant flow of packets to this AP. The second scenario is based on the Residential scenario proposed by IEEE HEW group and consists of a one floor building with 20 apartments. Each apartment represents a BSS with one AP at the center and 10 STAs around it, each sending a constant flow of packets to the AP. Three BSSs (or apartments) using the same channel were studied to detect the improvements of the new protocol.

Every scenario was tested using both CSMA/CA and CSMA/CA-DB under the same simulation conditions and several statistics were drawn from the simulation results. It is important to notice that all simulation results represent here the long-term scenario, where no STA uses mobility and, in case of a packet loss, the AP immediately detects it and assigns a new backoff value for that packet.

Simulations were created using 1448 bytes TCP payload size. For the PHY layer, the default YansWifiChannelHelper was used to configure the physical Wifi medium. As ns-3 API reference states in [21], this model is implemented based on the model described in "Yet Another Network Simulator" in [22]. This PHY model was applied using the default configurations, with short guard period enabled, which can achieve better throughput results and is safe to be applied since only IEEE 802.11g is being used. "AarfWifiManager" is the rate control algorithm. STAs MAC layer does not use active probing in these simulations. All scenarios tested use a constant position for both AP and STAs, which leads to "ConstantPositionMobility" being used as a mobility model. All TCP flows are generated using the "OnOffApplication", where all STAs attempting a fixed data rate transmission of packets, with all queues filled at all times. Each of these flows (for scenarios with more than a single STA) has a random start time to avoid pseudo-collisions triggered when two different flows have the same start time.

## 5.2 Preliminary Wifi Channel Capacity Characterization (1 AP / 1 STA)

For these simulations, TCP flows have increasing data rates to test the channel capacity of the wireless medium. Each simulation has a total time of 10 seconds. For these simulations, the random number generator was set using seed 2 and runs 1, 2, 3 and 4.

### 5.2.1 CSMA/CA

The first experience was developed in order to determine the maximum capacity of the medium. The simulation consisted of one AP and one STA, where the STA sent a continuous TCP flow to the AP, using IEEE 802.11g standard. In this simulation, the STA had a continuous flow at different data rates and there were goodput measurements made concerning the data flow from the STA to the AP.

The results statistics were collected using the Flow Monitor tool. Seed 2 was used for the pseudo-random number generator, and each scenario was run 4 times, using stream runs 1, 2, 3 and 4. The graphs were traced using the arithmetic average of the transmitted throughput obtained with Flow Monitor.

This experience reflects the maximum achievable throughput with continuous flow, without contention for channel access, using the current protocols for PHY and MAC layer. Two experiments were ran, the first, having the STA positioned 1 meter away from the AP, and the second, having the STA positioned 30 meters away from the AP. The goodput results are shown in Figure 5.1, and it is possible to conclude that the maximum achievable throughput was around the 24 Mbps.

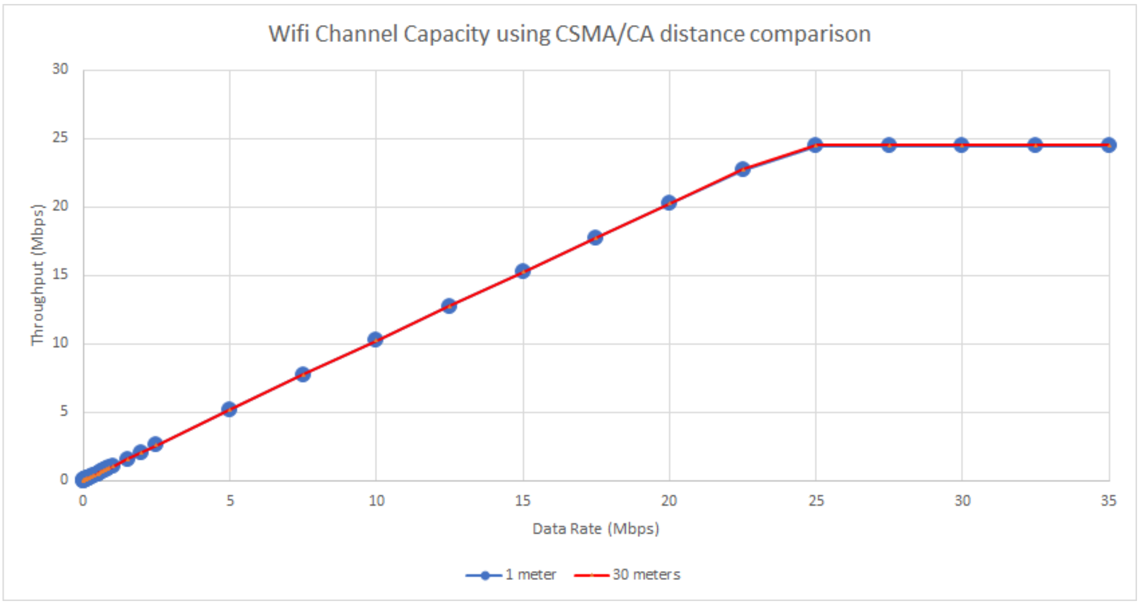


Figure 5.1: Channel capacity using CSMA/CA at 1 meter and 30 meters (with overlapped curves)

Due to IEEE 802.11g characteristics, the achieved goodput is the same at 1 meter and at 30 meters. This will be very important for the first scenario, simple BSS, since STAs are spread in the area surrounding the AP, at a maximum distance of 30 meters from the AP. This way it is safe to conclude that distance will not be a factor to take into account, since these results prove that the maximum throughput remains the same for both distances.

### 5.2.2 CSMA/CA-DB

In a second experiment, the same channel capacity assessment was performed, but using CSMA/CA-DB instead. The results for 30 meters, which was the exact result obtained for 1 meter of distance. These results reflect the same conditions used for CSMA/CA, with one STA and one AP, with IEEE 802.11g standard.

This time, the maximum achievable throughput was close to 30Mbps. This is a very important result. The channel capacity has increased with the change in the MAC layer protocol. Figure 5.2 shows a comparison between CSMA/CA and CSMA/CA-DB for channel capacity at 30 meters of distance.

As Figure 5.2 shows, the channel capacity, under the same simulation conditions, was improved by almost 5Mbps.

## 5.3 Simple BSS scenario (1 AP / 1 to 30 STAs)

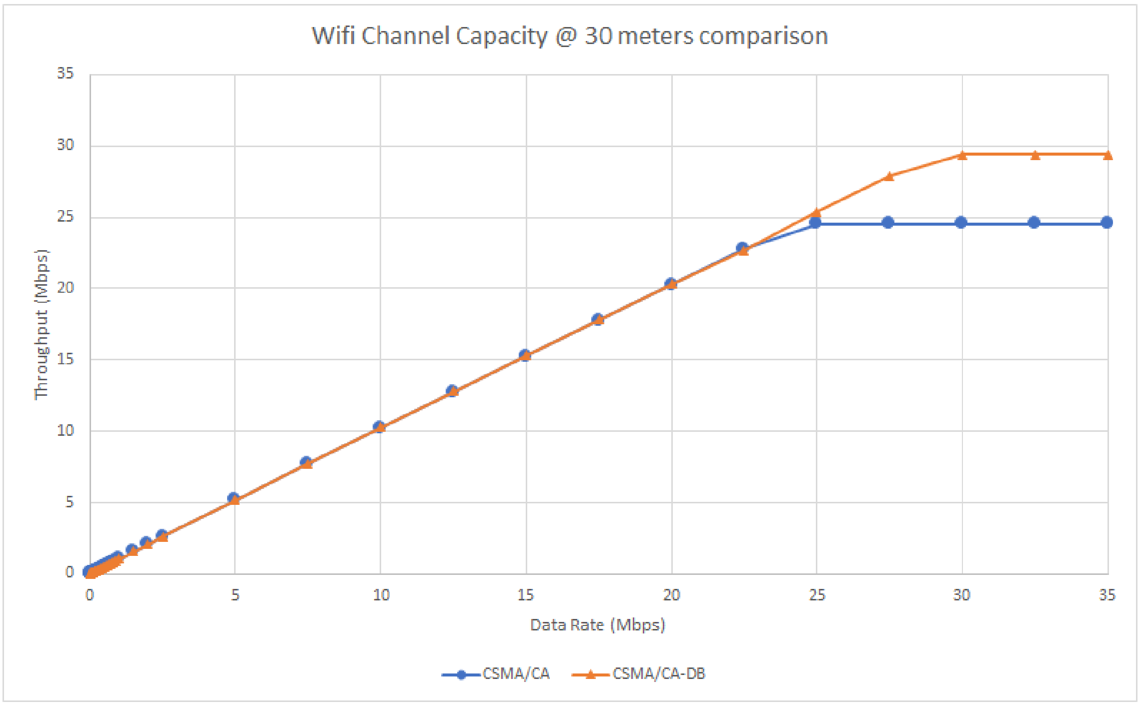


Figure 5.2: Channel capacity comparison at 30 meters of distance

In order to test the current MAC layer protocols, a simple scenario was built, simulating an isolated BSS, formed by an AP with one STA in a first attempt, and then gradually adding more STAs to this BSS, using IEEE 802.11g. A simple drawing is shown below in Figure 5.3 to demonstrate the BSS structure used for this simulation.

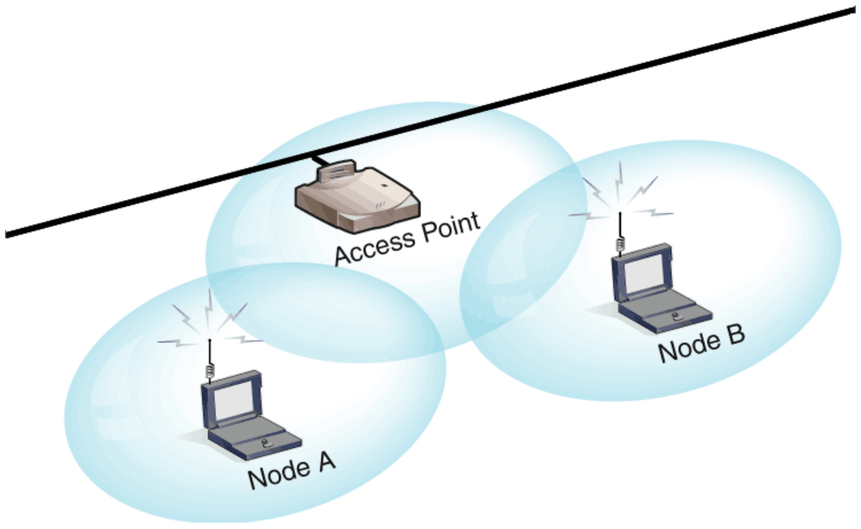


Figure 5.3: Simple BSS scenario [9]

The AP was positioned on the point of coordinates (0,0) and then the first STA was positioned on the point (1,0). Gradually other STAs were included in the BSS until the final simulation containing 30 STAs placed around the AP, in a 30 meters radius. These points were randomly generated and then kept the same for all simulations. Figure 5.4 shows the position of the AP and the STAs.

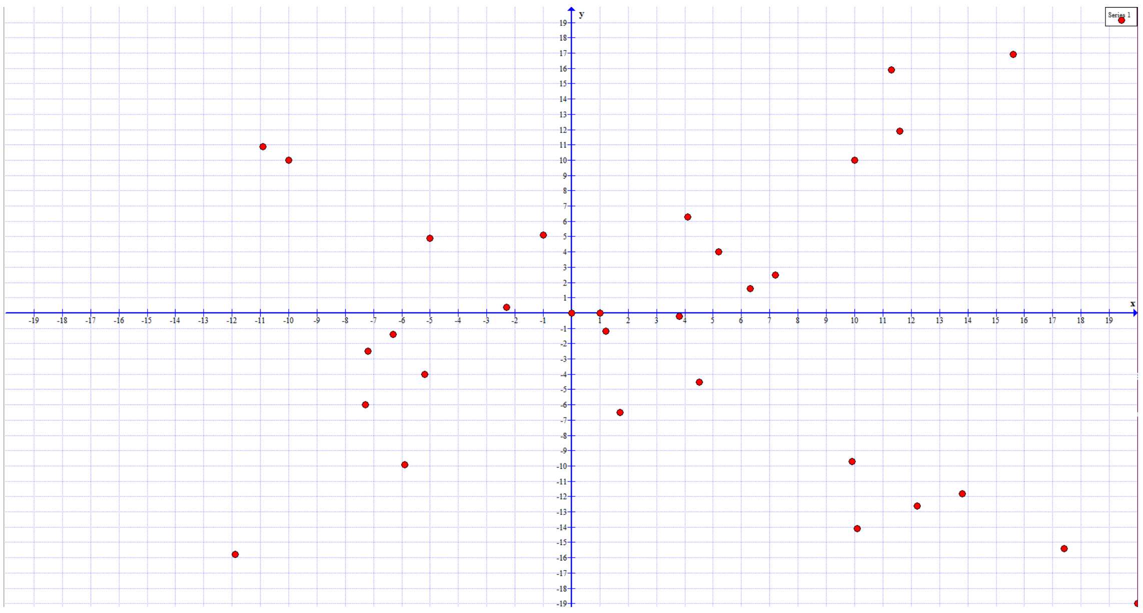


Figure 5.4: Coordinates of AP and STAs in simple BSS scenario

The position of the STAs and AP followed the Constant Position Mobility Model in ns-3, where all nodes of the network are placed in a structure containing 3 dimensional vectors. Packet sinks were added to all STAs and AP in order to capture the flows on the network, so that all the statistic data could be easily processed. The data rate for the TCP flow was purposely chosen to be higher than the channel capacity, at 100Mbps. To capture statistics of all the flows, the "Flow Monitor" tool was used and applied to all nodes of the network and a XML file was generated in the end of the simulation containing data of the flows, IP address of the source and destination of the flows, port, transmitted throughput, received throughput, and number of packets sent and received. "Ipv4GlobalRoutingHelper" was used as the routing protocol. The random number generator was set using the default seed and runs 1 to 8.

Table 5.1 represents the simulation parameters for the Simple BSS scenario:

### 5.3.1 CSMA/CA

Figure 5.5 shows the aggregate throughput obtained through the various simulations, which consists of the sum of all the individual throughput of the flows from the STAs to the AP. It is visible that as the number of STAs increases the aggregate throughput decreases. This is due to

Table 5.1: Simple BSS scenario simulation parameters

| Wifi |
|---|
| Infrastructure network: 192.168.0.0/24 |
| IEEE 802.11g |
| SSID: ns-3-ssid |
| Routing: Ipv4GlobalRouting |
| MAC |
| AarfWifiManager |
| PHY |
| YansWifiPhy |
| Short Guard Interval enabled |
| Mobility |
| ConstantPositionMobilityModel |
| Simulations |
| Seed 2, Runs 1 to 8 |
| Simulation Time: 60s |
| Confidence Interval: 99% |

the fact that the number of collisions is greater on a dense scenario, because of channel contention and the problems already pointed to CSMA/CA on dense scenarios.

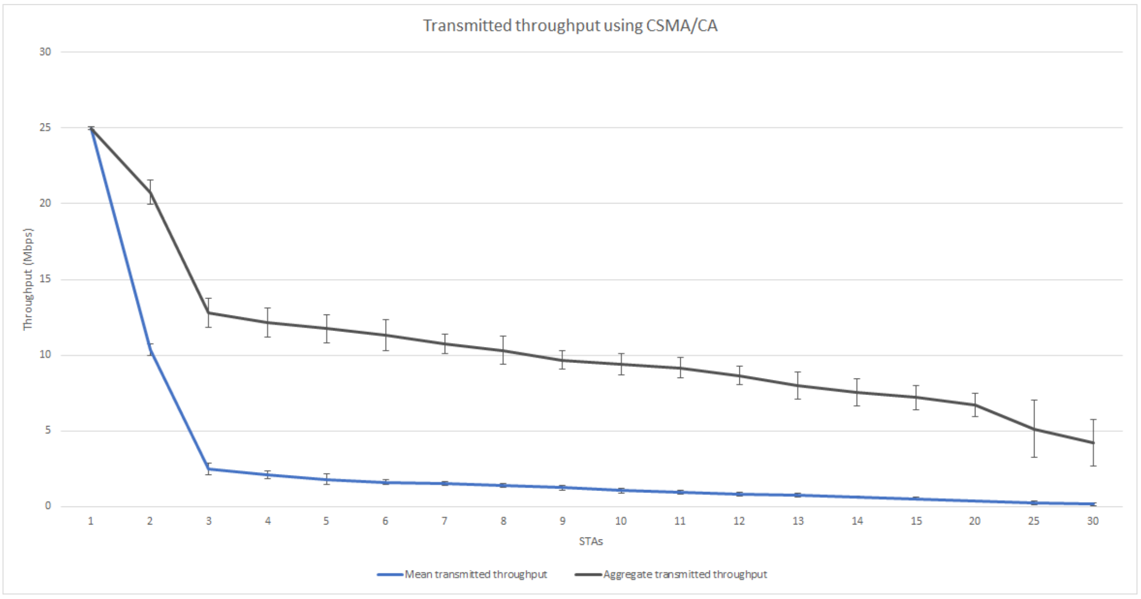


Figure 5.5: Mean transmitted throughput per STA (in blue) and aggregate throughput (in black)

In black, the graph has the aggregate throughput and in blue the mean throughput obtained by each STA. As the number of STAs increase, the channel access is divided by more and more STAs, which results in a decrease of the mean throughput obtained by each STA. However, the aggregate throughput suffers a decrease as well, due to collisions, which grow with the number of STAs.

Another interesting graph to draw at this moment is the number of packets that are lost during the simulation time, which is represented in Figure 5.6.

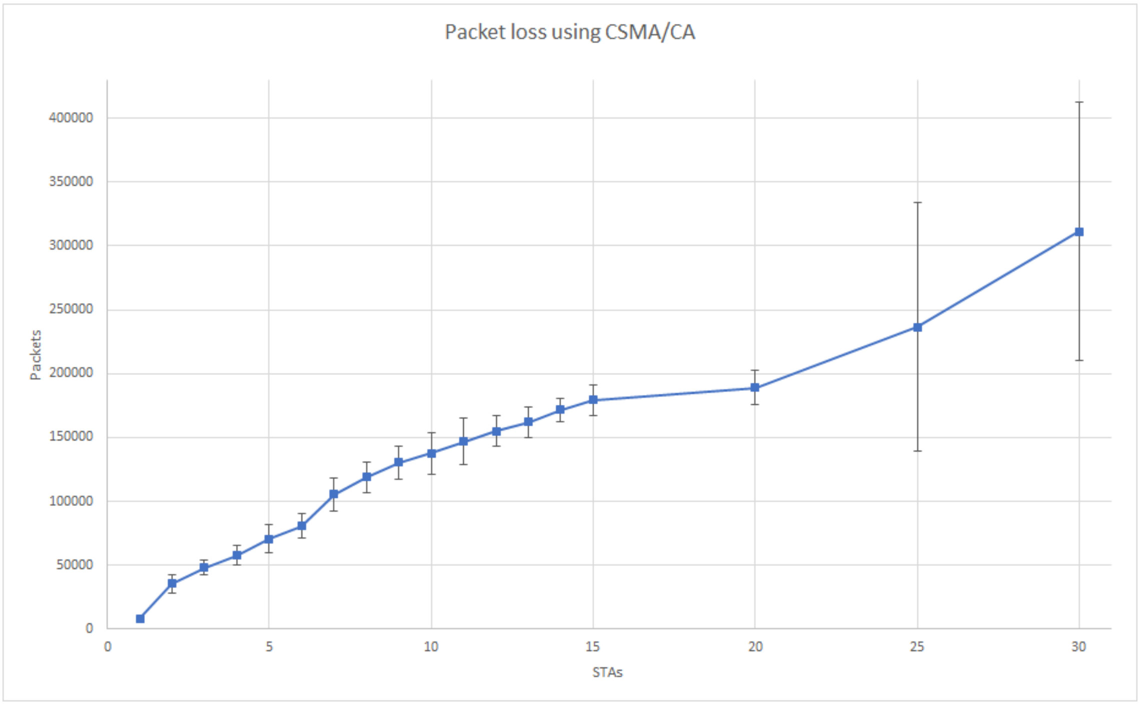


Figure 5.6: Packet loss using CSMA/CA

The number of packets lost on the network increase with the number of STAs trying to access the medium. There is a low percentage of packets that are lost with few STAs contending for the channel, but the more STAs contending for the channel, the longer the time the channel is idle due to the exponential backoff mechanism.

It is important to analyze the reasons for packet drops. There are four different ways a packet can be triggered in the ns-3 simulator as dropped.

A "PhyRxDrop" can be triggered by a channel switching, which in this scenario will never be triggered, the packet can be dropped by being already in reception mode or transmission mode and another packet is sensed by the PHY layer (a collision), the packet can be dropped because the device is in sleep mode, which will not happen in these simulations. Another cause can be an unsupported mode, which will not happen throughout this work, or a Physical Layer Convergence Protocol (PLCP) preamble/header reception failed. The PLCP appends a PHY-specific preamble and header fields to the MPDU that contains information needed by the PHY layer for transmission and reception. A failure in the reception of this field triggers a packet drop. Another way for a packet to be lost is, after reading the PLCP header successfully, the PHY layer will generate a random value and check if this value is greater than the calculated Signal to Interference plus Noise Ratio (SNIR) at the start of the PLCP payload, after being accumulated with all SNIR changes detected in the vector. If the condition fails, and the random value is lesser than this SNIR value, the packet is dropped. There are still cases when packet drops are triggered in case the device is in reception mode and the reception is cancelled or even if the signal power is too small, due to interference, distance, etc.

A "PhyTxDrop" is only called if the device is in sleep mode.

A "MacTxDrop" is only called when an STA is not associated with an AP. This way, any attempts to transmit a packet will fail at the MAC layer.

A "MacRxDrop" can be triggered if a packet is received but the MAC address "TO" does not correspond to the MAC address of the device. In this case, the MAC layer will drop the packet. Even in the case of CSMA/CA-DB, where before contending for the channel some STAs would have to listen to ACKs sent by the AP to other STAs, these packets would still be dropped by the corresponding MAC layers, as there is no processing to be done. Another way that a packet can be dropped is if a DATA packet was received while the STA is not yet associated or if the DATA packet has the flag "FROM DS" set to 0, which means the packet is not intended for the current wireless environment. Another situation occurs when a DATA packet is received but it does not come from the BSS the device is associated with. This is a common case when there are several APs in the same channel and STAs from one BSS can "hear" packets intended to another BSS. Packets can be dropped if the device is an STA and the packet is intended to an AP, for example, if the packet is a probe request or an association request.

Different types of packet drops will occur in the different simulations presented. Looking at the packet tracing for the several simulations performed, it is possible to detect the moments when these drops occurred and the respective TCP retransmissions. Figure 5.7 shows a moment where a TCP retransmission occurred for the simulation with 1 STA.

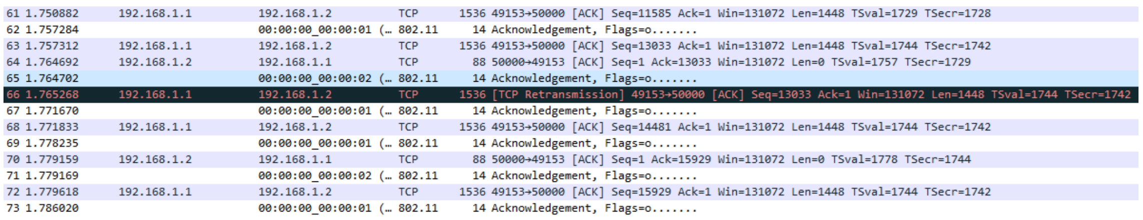

```
61 1.750882   192.168.1.1    192.168.1.2          TCP     1536 49153→50000 [ACK] Seq=11585 Ack=1 Win=131072 Len=1448 TSval=1729 TSecr=1728
62 1.757284                  00:00:00_00:00:01 (… 802.11    14 Acknowledgement, Flags=o.......
63 1.757312   192.168.1.1    192.168.1.2          TCP     1536 49153→50000 [ACK] Seq=13033 Ack=1 Win=131072 Len=1448 TSval=1744 TSecr=1742
64 1.764692   192.168.1.2    192.168.1.1          TCP       88 50000→49153 [ACK] Seq=1 Ack=13033 Win=131072 Len=0 TSval=1757 TSecr=1729
65 1.764702                  00:00:00_00:00:02 (… 802.11    14 Acknowledgement, Flags=o.......
66 1.765268   192.168.1.1    192.168.1.2          TCP     1536 [TCP Retransmission] 49153→50000 [ACK] Seq=13033 Ack=1 Win=131072 Len=1448 TSval=1744 TSecr=1742
67 1.771670                  00:00:00_00:00:01 (… 802.11    14 Acknowledgement, Flags=o.......
68 1.771833   192.168.1.1    192.168.1.2          TCP     1536 49153→50000 [ACK] Seq=14481 Ack=1 Win=131072 Len=1448 TSval=1744 TSecr=1742
69 1.778235                  00:00:00_00:00:01 (… 802.11    14 Acknowledgement, Flags=o.......
70 1.779159   192.168.1.2    192.168.1.1          TCP       88 50000→49153 [ACK] Seq=1 Ack=15929 Win=131072 Len=0 TSval=1778 TSecr=1744
71 1.779169                  00:00:00_00:00:02 (… 802.11    14 Acknowledgement, Flags=o.......
72 1.779618   192.168.1.1    192.168.1.2          TCP     1536 49153→50000 [ACK] Seq=15929 Ack=1 Win=131072 Len=1448 TSval=1744 TSecr=1742
73 1.786020                  00:00:00_00:00:01 (… 802.11    14 Acknowledgement, Flags=o.......
```

Figure 5.7: Packet tracing showing TCP retransmissions

According to the statistics captured through the packet drop callbacks in ns-3, for the case of one STA the only drops occurred due to "PhyRxDrop" because of collisions. The delays involved with retransmissions at the MAC layer level, with the exponential backoff mechanism, led to TCP retransmissions. These retransmissions are triggered when the Round trip time (RTT) for a certain packet expires. In TCP, there is an initial RTT estimation and every time a packet is sent and an ACK received a few previous samples are kept. This RTT estimation is a moving average that is constantly updated according to the following equation:

$$RTT = \alpha.oldRTT + (1 - \alpha).newRTT \tag{5.1}$$

Choosing $\alpha$ for the weighting factor ($0 \leq \alpha < 1$) close to 1 makes the weighted average immune to changes that last a short time (e.g. a single segment that faces a big delay). Choosing $\alpha$ close to 0 makes the weighted average respond to changes in delay very quickly. ns-3 has a default $\alpha$ value and inferring the impact of a change in this default value is beyond the scope of this work. However, if a packet encounters small delays for a number of consecutive frames, these results are kept in the moving average of the RTT estimator; if then a long delay is encountered by the next frame, a TCP retransmission can be triggered.

For the other simulations, with more STAs, the packet drops are due to collisions, failed reception of the PLCP preamble or a fail because of the SNIR value. These collisions reflect how much denser the scenario has become.

Another interesting result to look at, which is related with the previous graphs is the total number of bytes received at the AP, as shown in Figure 5.8.

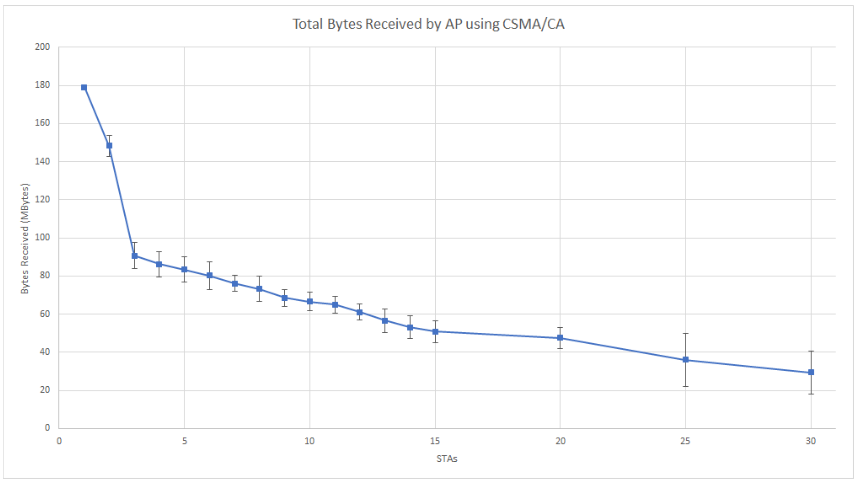


Figure 5.8: Total number of bytes received by the AP using CSMA/CA

This graph shows a curve that follows the same curve as the aggregate throughput obtained.

### 5.3.2 CSMA/CA-DB

Figure 5.9 shows the aggregate throughput obtained through the various simulations, which consists of the sum of all the individual throughput of the flows from the STAs to the AP. It is visible that as the number of STAs increases the aggregate throughput decreases with a slower slope when compared to CSMA/CA. This is due to the fact that with CSMA/CA-DB there are still collisions and packet losses, due to interference and collisions with management frames and other packets that are sent to the medium. As seen previously with the channel capacity measurements, the medium capacity is close to 30Mbps with 1 STA associated with the AP.

In red, the graph has the aggregate transmitted throughput and in orange the mean throughput obtained by each STA. As with CSMA/CA, as the number of STAs increase, the channel access is divided by more and more STAs, which results in a decrease of the mean throughput obtained by each STA.

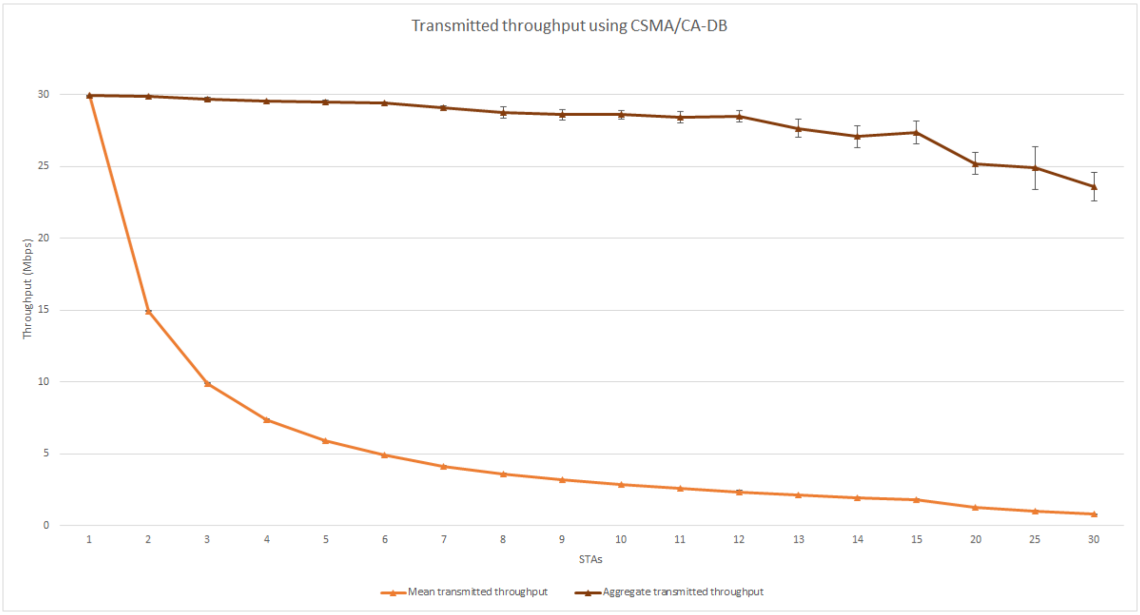


Figure 5.9: Mean and aggregate throughput using CSMA/CA-DB

Another interesting graph to draw at this moment is the number of packets that are lost during the simulation time, which is represented in Figure 5.10.

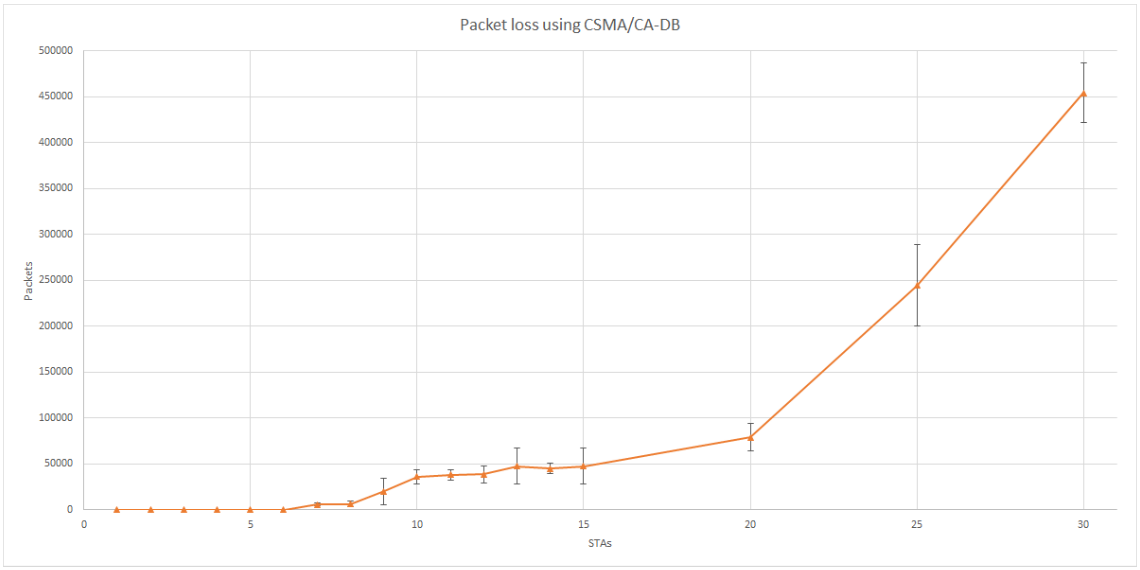
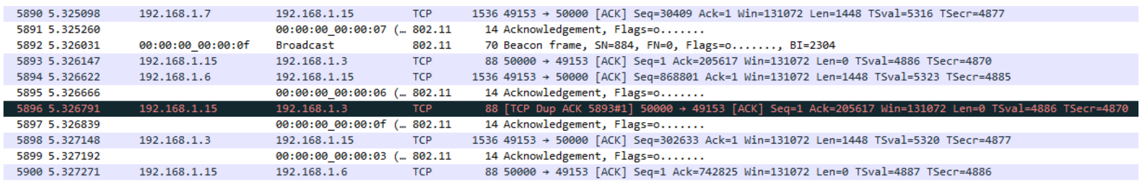


Figure 5.10: Packet loss using CSMA/CA-DB

The number of packets lost on the network increase with the number of STAs trying to access the medium. Up to 8 STAs contending for the channel, the percentage of packets lost on the network is close to zero. However, with an increasing number of STAs contending for the channel access, the number of packets that are lost increases. Figure 5.11 shows some of the packet tracing for the simple BSS experiment using CSMA/CA-DB, captured at one of the STAs, showing DATA packets, ACK frames, periodic beacons and a TCP duplicated ACK.



```
5890 5.325098   192.168.1.7          192.168.1.15         TCP     1536 49153 → 50000 [ACK] Seq=30409 Ack=1 Win=131072 Len=1448 TSval=5316 TSecr=4877
5891 5.325260                        00:00:00_00:00:07 (… 802.11    14 Acknowledgement, Flags=o.......
5892 5.326031   00:00:00_00:00:0f    Broadcast            802.11    70 Beacon frame, SN=884, FN=0, Flags=o......., BI=2304
5893 5.326147   192.168.1.15         192.168.1.3          TCP       88 50000 → 49153 [ACK] Seq=1 Ack=205617 Win=131072 Len=0 TSval=4886 TSecr=4870
5894 5.326622   192.168.1.6          192.168.1.15         TCP     1536 49153 → 50000 [ACK] Seq=868801 Ack=1 Win=131072 Len=1448 TSval=5323 TSecr=4885
5895 5.326666                        00:00:00_00:00:06 (… 802.11    14 Acknowledgement, Flags=o.......
5896 5.326791   192.168.1.15         192.168.1.3          TCP       88 [TCP Dup ACK 5893#1] 50000 → 49153 [ACK] Seq=1 Ack=205617 Win=131072 Len=0 TSval=4886 TSecr=4870
5897 5.326839                        00:00:00_00:00:0f (… 802.11    14 Acknowledgement, Flags=o.......
5898 5.327148   192.168.1.3          192.168.1.15         TCP     1536 49153 → 50000 [ACK] Seq=302633 Ack=1 Win=131072 Len=1448 TSval=5320 TSecr=4877
5899 5.327192                        00:00:00_00:00:03 (… 802.11    14 Acknowledgement, Flags=o.......
5900 5.327271   192.168.1.15         192.168.1.6          TCP       88 50000 → 49153 [ACK] Seq=1 Ack=742825 Win=131072 Len=0 TSval=4887 TSecr=4886
```

Figure 5.11: Packet loss using CSMA/CA-DB

The number of packets lost is relatively high for 25 and 30 STAs. However, with CSMA/CA-DB, the idleness of the channel is considerably lower than CSMA/CA, which leads to a bigger number of packets on the network during the simulation time. This is reflected in higher throughput plus packet loss. With a higher number of STAs, a higher number of management packets are flowing on the network, which associated with the lower idleness of the channel, lead to the number of packet loss and consequent aggregate throughput decrease. One improvement to the protocol would be extending the deterministic backoff generation for management and control frames to avoid the throughput degradation for very dense scenarios. The collisions detected throughout the simple BSS scenario using CSMA/CA-DB were related to packet drops due to collisions (already in Tx/already in Rx) or the SINR frame fail.

A last graph is shown with the results for the total number of bytes received at the AP, which follows the same curve as the aggregate throughput, as shown in Figure 5.12.

The code for the simple BSS simulation is found on Appendix A.1.

### 5.3.3 CSMA/CA and CSMA/CA-DB Comparison for the Simple BSS Scenario

Figure 5.13 shows the aggregate throughput comparison between the two protocols.

As it is visible in the graph, the results for the aggregate throughput show a better performance with the new protocol, CSMA/CA-DB, compared with the standard CSMA/CA protocol, at all times.

The following graph in Figure 5.14 shows the comparison between the two protocol in terms of mean throughput obtained at the STA for each simulation.

As shown in the graph, the results for the mean throughput confirm the benefits of using CSMA/CA-DB. The mean throughput is better for one single STA, since the channel capacity is higher. The results for 20 or more STAs are still visible on the graph, and there is still a good

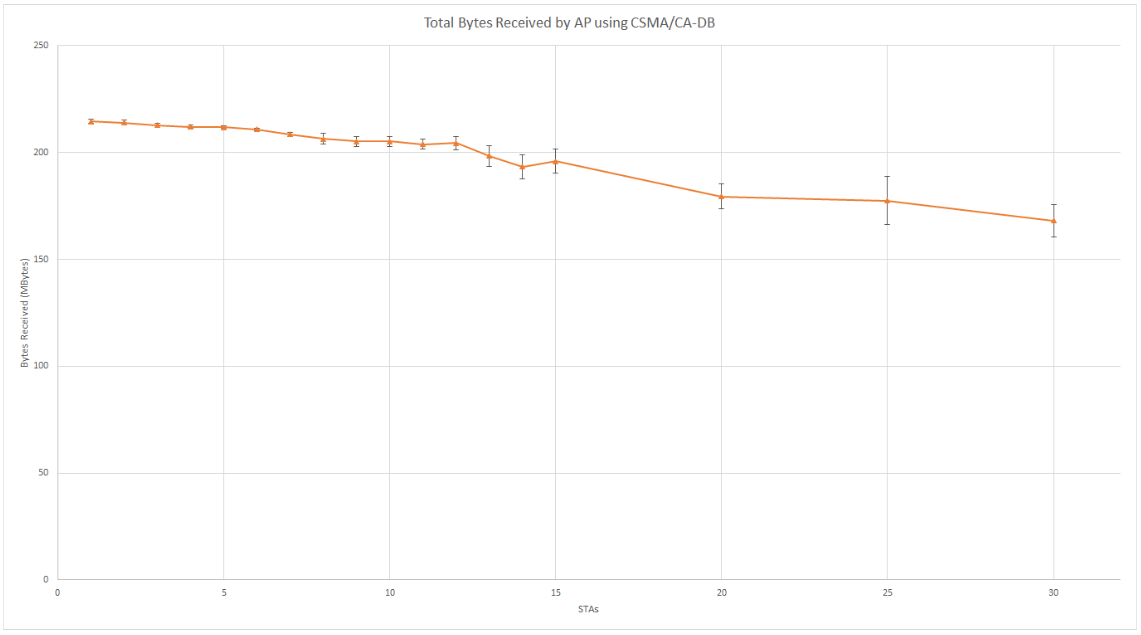


Figure 5.12: Total number of bytes received by the AP using CSMA/CA-DB

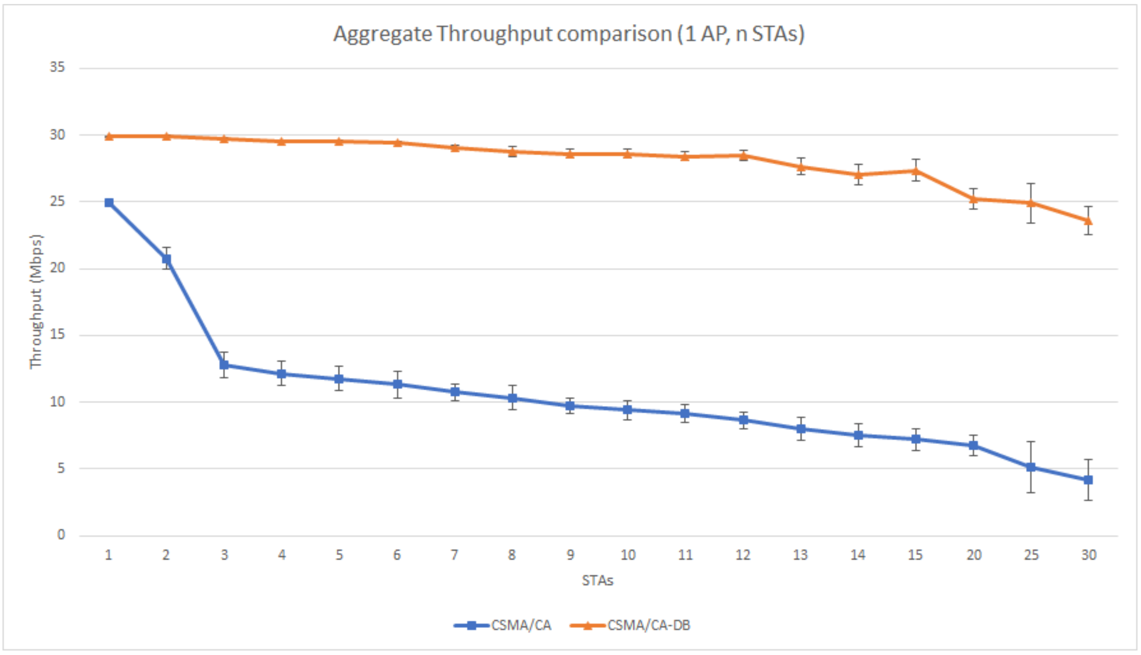


Figure 5.13: Aggregate throughput comparison for simple BSS scenario

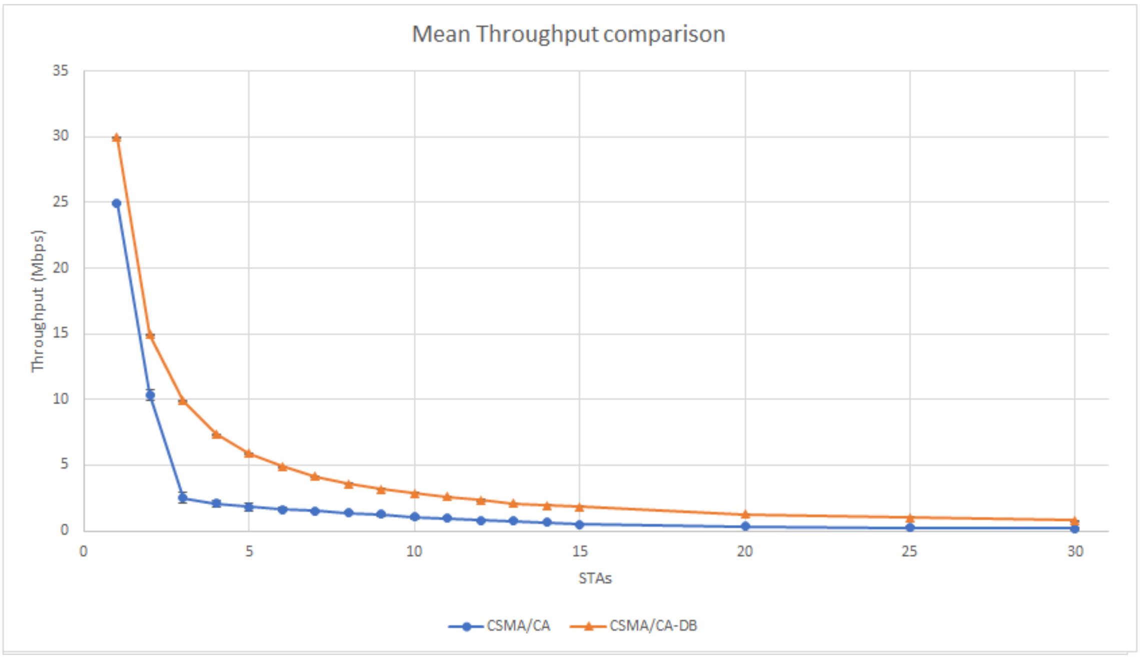


Figure 5.14: Mean throughput comparison

margin between CSMA/CA-DB and CSMA/CA numbers for each individual STA performance on the network.

Figure 5.15 shows the comparison between the two protocols in terms of total bytes received during the simulation time.

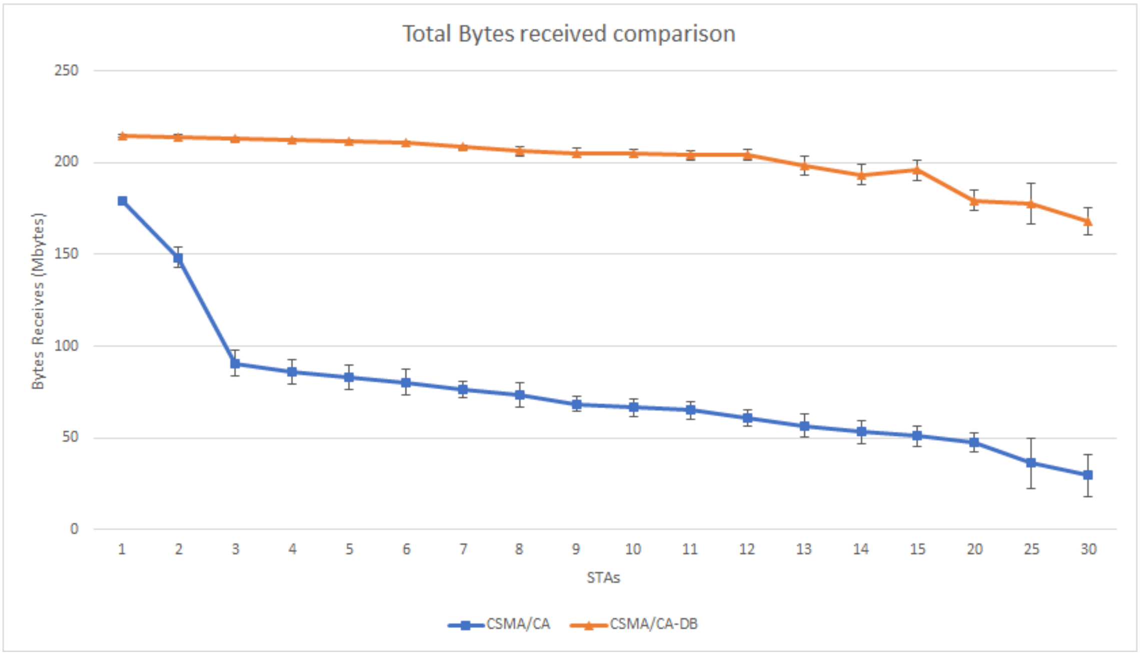


Figure 5.15: Total number of bytes received by the AP

This is just another parameter of the simulation and it only reflects the differences in throughput. It allows to validate the simulation throughput results, and transpose them to goodput results, i.e. the throughput statistics reflect packets that were sent by the STA and were effectively received by the AP.

Figure 5.16 shows the packet loss comparison between both protocols. Up to 25 STAs, CSMA/CA-DB shows a better performance, with less packets lost on the channel. However, for 25 and 30 STAs, there is a higher number of packets that are lost with the new protocol. This means a higher number of retransmissions or packet drops on the wire, but this is due to an inferior idleness of the channel. If the channel is not idle, more packets are being sent, which is reflected in the previous graphs of throughput and bytes sent.

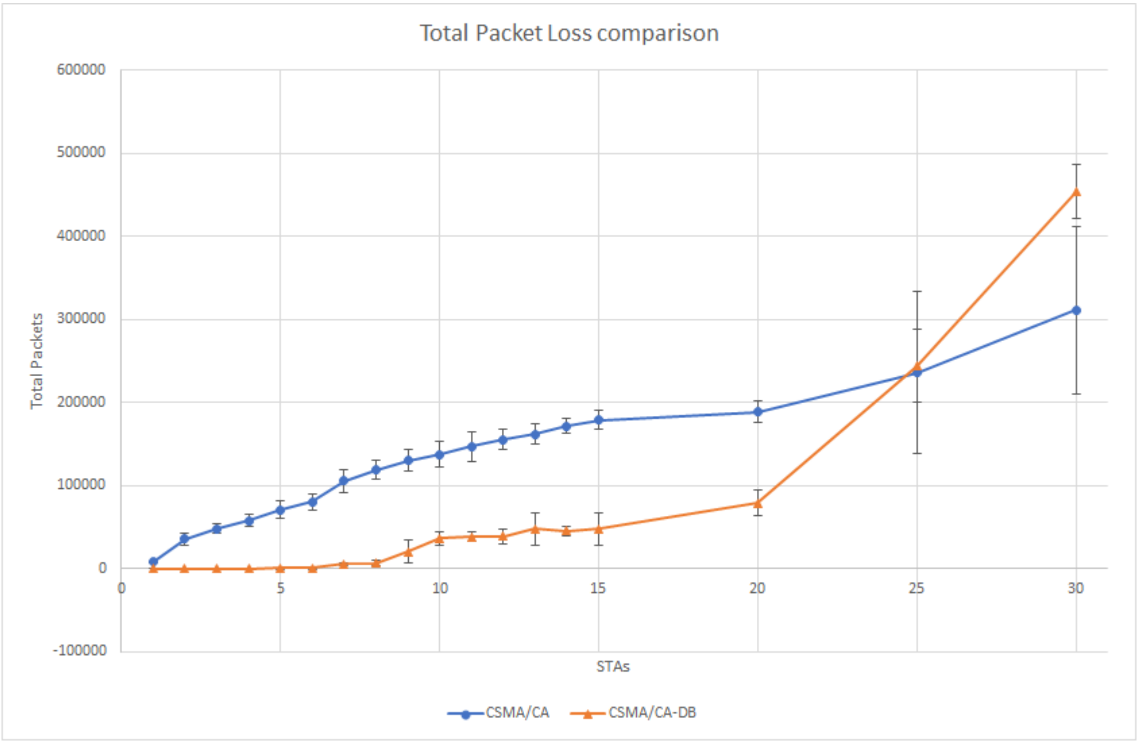


Figure 5.16: Packet loss comparison

Figure 5.17 shows an interesting graph, comparing the aggregate throughput and the number of packets that are lost for CSMA/CA. Actually, the optimal point is found at 10 STAs, i.e. the smaller throughput loss compromise due to an increase in the number of STAs on the network. For CSMA/CA, an increase in STAs reflects a great amount of retransmissions and, consequently, a degradation in the aggregate throughput obtained.

Figure 5.18 shows the same information for CSMA/CA-DB. In this case, the optimal point is located at a point between 25 and 30 STAs. This means that, under the same simulation conditions, CSMA/CA-DB handles more STAs contending for the channel when compared to CSMA/CA.

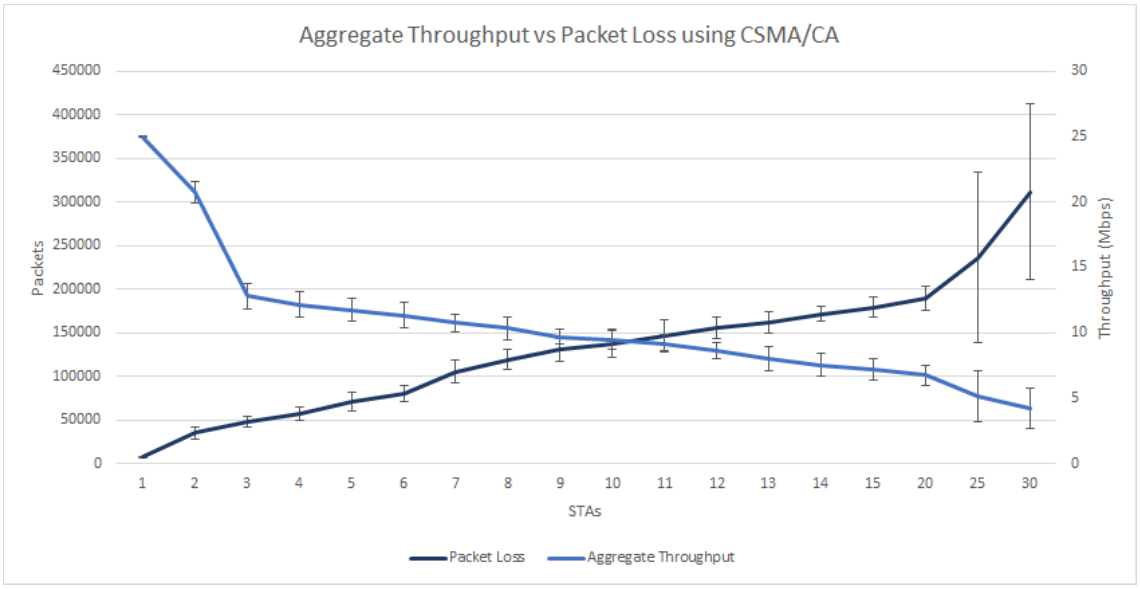


Figure 5.17: Aggregate throughput and packet loss comparison using CSMA/CA

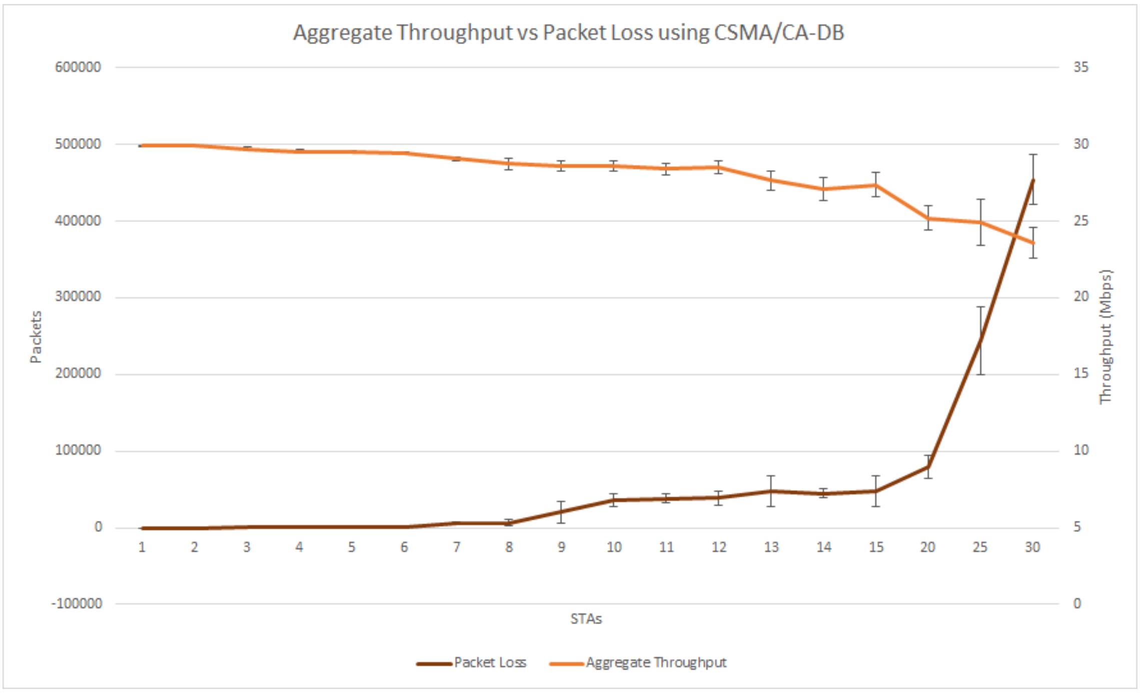


Figure 5.18: Aggregate throughput and packet loss comparison using CSMA/CA-DB

## 5.4 Building Scenario

The second scenario tested was a one floor building with 20 apartments in a 10x2 layout, as it can be seen on Figure 5.19. This scenario was thought to test a multi-channel environment with walls causing blockages and interference on the radio signal. This scenario was based on the residential scenario proposed by the IEEE HEW group as in [20]. The building is one of the floors of the original residential scenario. This simulation is meant to test how the new protocol for the MAC layer will react on a dense scenario with several Wifi networks in close proximity, simulating a real residential scenario with different networks using the same physical channel, hidden nodes and exposed nodes. In order to test the several Wifi channels, each apartment had a certain channel given, either 1, 6 or 11, representing one of the three non-overlapping channels available in the IEEE 802.11g frequency spectrum. For time frame reasons, only three apartments, using the same channel, were studied, which are represented in blue in figure 5.19.

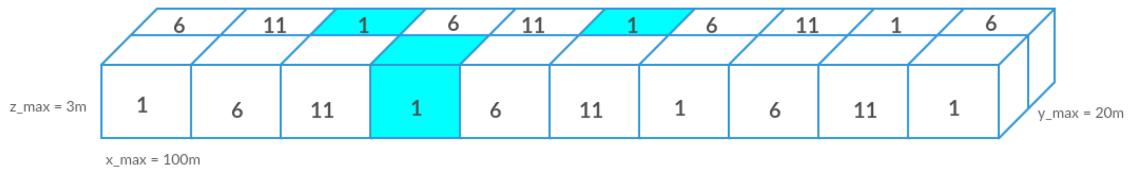


Figure 5.19: Building scenario with three apartments in channel 1 depicted

The dimensions of each apartment are shown in the above figure and this building was simulating in ns-3 using the "Building" class. The simulations ran focused on three of these apartments, which are represented in blue.

Each apartment is a BSS consisting of one AP and 10 STAs. The AP is positioned at the center of the apartment, surrounded by 10 STAs in a fixed position. Figure 5.20 shows the position of the AP and STAs for each BSS.

Simulating a multi-channel scenario in ns-3 is possible using N instances of "YansWifiChannel". To each channel, N APs are associated with it and the respective STAs. ns-3 does not simulate the interference between channels with spectral overlap. This way, APs and STAs in channel 1 are not affected by APs and STAs in channel 2, even if from the specified frequency of each channel there is overlapping. This way, the only way to simulate a real multi-channel scenario is creating three different channel instances. For each BSS a different instance of the MAC layer should also be instantiated.

The position of the STAs and AP followed the Constant Position Mobility Model in ns-3, where all nodes of the network are placed in a structure containing 3 dimensional vectors. Packet sinks were added to all STAs and AP in order to capture the flows on the network, so that all the statistic data could be easily processed. The data rate for the TCP flow was set at 25Mbps. The random number generator was set using seed 6 and runs 1 to 30. The simulation time was set to 60 seconds with 30 seconds of settling time to allow the routing protocol to converge. In order

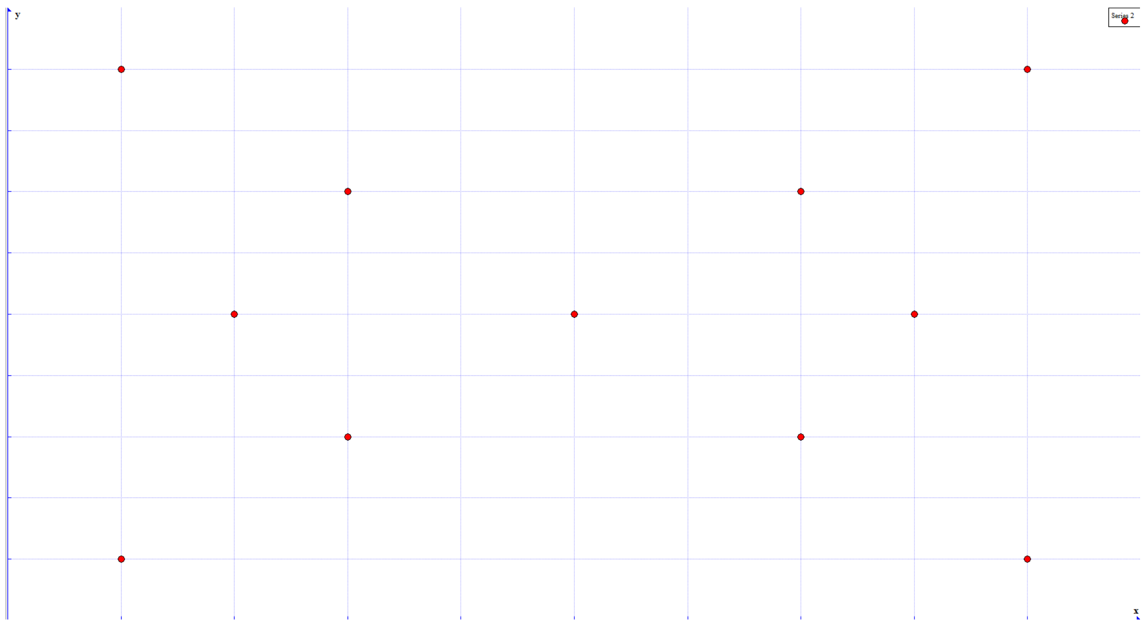


Figure 5.20: Coordinates of AP and STAs in the residential scenario, in a 10x10m apartment

to capture statistics of all the flows, the "Flow Monitor" tool was used and applied to all nodes of the network and a XML file was generated in the end of the simulation containing data of the flows, IP address of the source and destination of the flows, port, transmitted throughput, received throughput, packet loss percentage, number of packets sent and received. AODV was used as the routing protocol, since "Ipv4GlobalRouting" does not work well with several Wifi networks, and the options are OLSR, AODV or DSR.

The code for this simulation is found on Appendix A.2.

Table 5.2 represents the simulation parameters for the Building scenario:

### 5.4.1 CSMA/CA

Figure 5.21 shows the aggregate throughput obtained through the various simulations, which consists of the sum of all the individual throughput of the flows from the STAs to the AP, with each vertical bar representing one of the three apartments under study. The fourth bar represents the aggregate throughput obtained on channel 1 for all BSSs combined.

Figure 5.22 shows the mean transmitted throughput obtained in the simulation results on each of the apartments studied.

Compared with the previous simulations, this time there is another variable into account, the routing protocol. It happened that some simulations, due to the dense scenario involved, had to be re-run since the routing protocol did not have an opportunity to fully transmit its packets throughout the network. The routing protocol used some of the bandwidth available, which was not seen in the simple BSS scenario.

Table 5.2: Building scenario simulation parameters

| Wifi |
|---|
| Infrastructure network: 192.168.x.0/24 |
| IEEE 802.11g |
| SSID: ns-3-ssid-x |
| Routing: AODV |
| MAC |
| AarfWifiManager |
| PHY |
| YansWifiPhy |
| Short Guard Interval enabled |
| Mobility |
| ConstantPositionMobilityModel |
| Simulations |
| Seed 6, Runs 1 to 30 |
| Settling Time: 30s; Simulation Time: 60s |
| Confidence Interval: 95% |

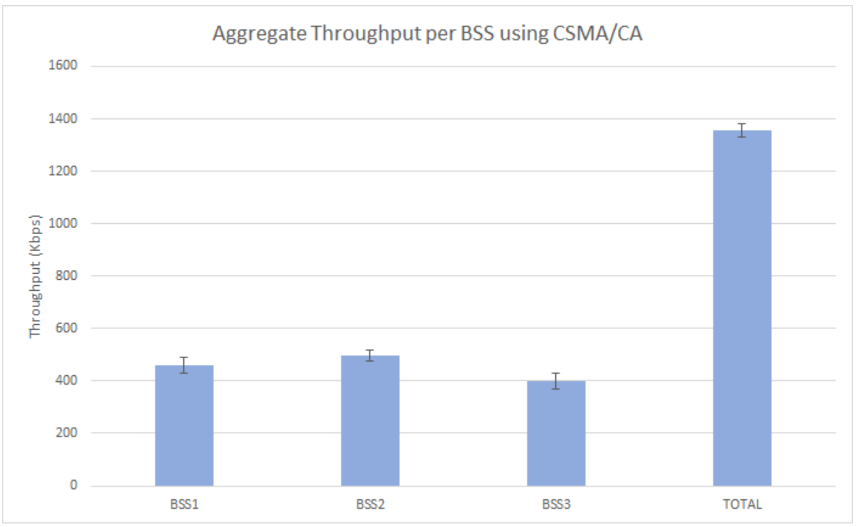


Figure 5.21: Aggregate throughput per BSS using CSMA/CA

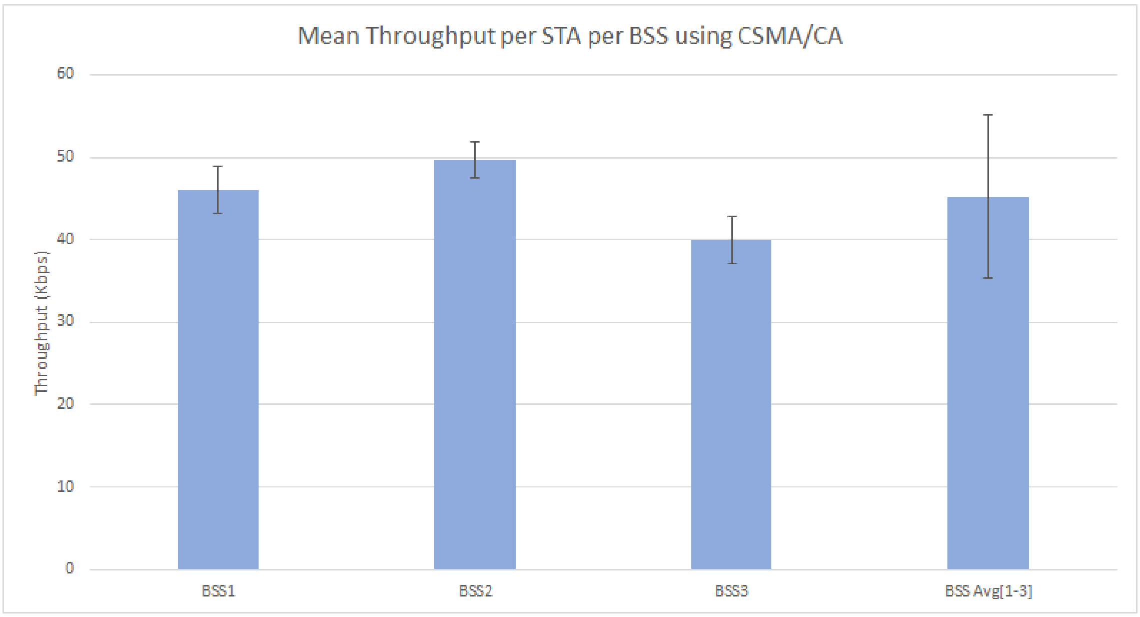


Figure 5.22: Mean transmitted throughput per STA at each BSS using CSMA/CA

Figure 5.23 shows the bytes received at each AP.

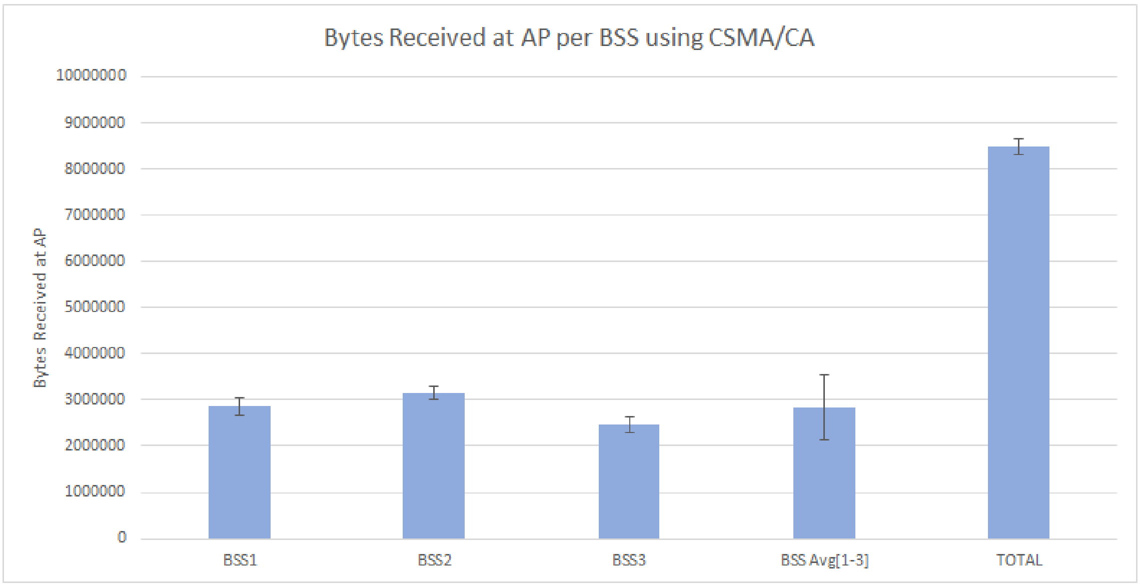


Figure 5.23: Bytes received at each AP using CSMA/CA

As it can be seen in the graph, CSMA/CA has a fair spreading of the bandwidth of the channel. The three BSSs simulated have closely the same number of bytes received at the AP.

Among the detected collisions and packet drops, the same reasons for the PHY layer were detected, as it happened with the simple BSS scenario: SNIR fail, PLCP header reception failed or already in Rx. However, MAC layer drops were identified this time: At the AP level frames not targeted to the AP were dropped, which means the AP could hear frames not meant to its own BSS; at the STA level when receiving frames while the STA was not associated or when receiving a DATA frame not from the BSS the STA is associated with, which means the STA could hear that pertained to another BSS.

Figure 5.24 shows the types of packet loss seen throughout the simulations and their figures.

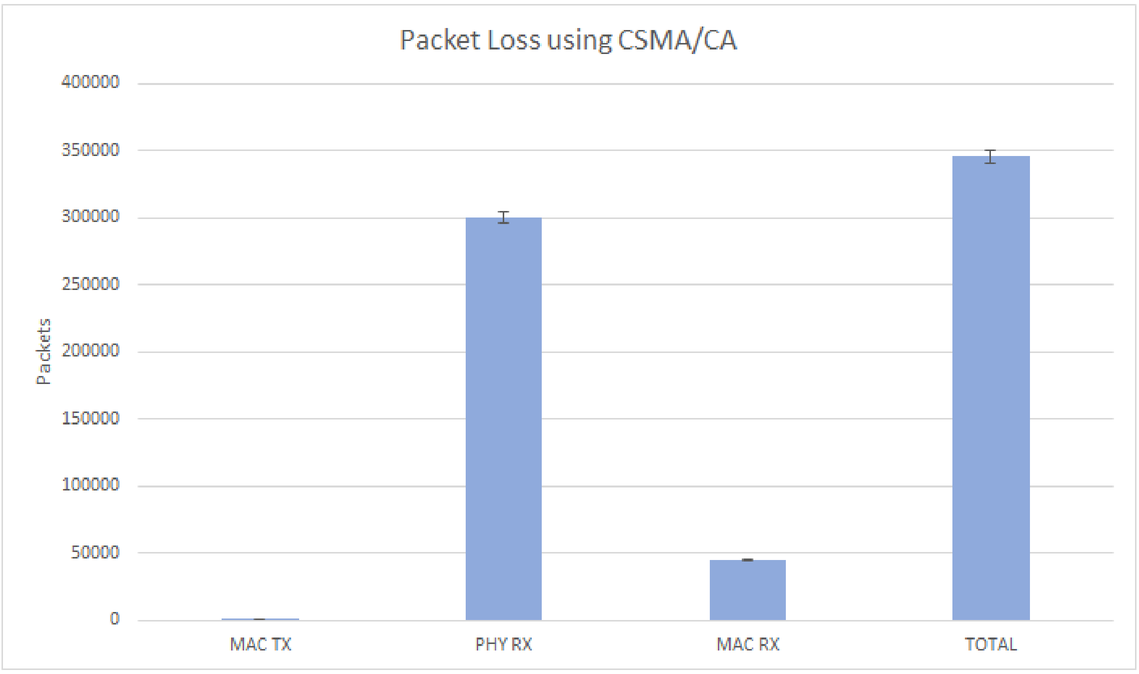


Figure 5.24: Packet loss detected on the network using CSMA/CA

Most of the packets that are loss have to do with collisions or interference. However, in this building scenario, due to the different networks, one for each apartment, there are other types of packet loss, namely MAC TX and MAC RX.

### 5.4.2 CSMA/CA-DB

Figure 5.25 shows the aggregate throughput obtained through the various simulations, which consists of the sum of all the individual throughput of the flows from the STAs to the AP, with each vertical bar representing one of the three apartments under study. The fourth bar represents the total aggregate throughput obtained for the different BSSs combined.

Figure 5.26 shows the mean throughput obtained in the simulation results on each of the apartments studied.

There is another variable into account, as stated previously, corresponding to AODV routing protocol. It happened that some simulations, due to the dense scenario involved, had to be re-run since the routing protocol did not have an opportunity to fully transmit its packets throughout the

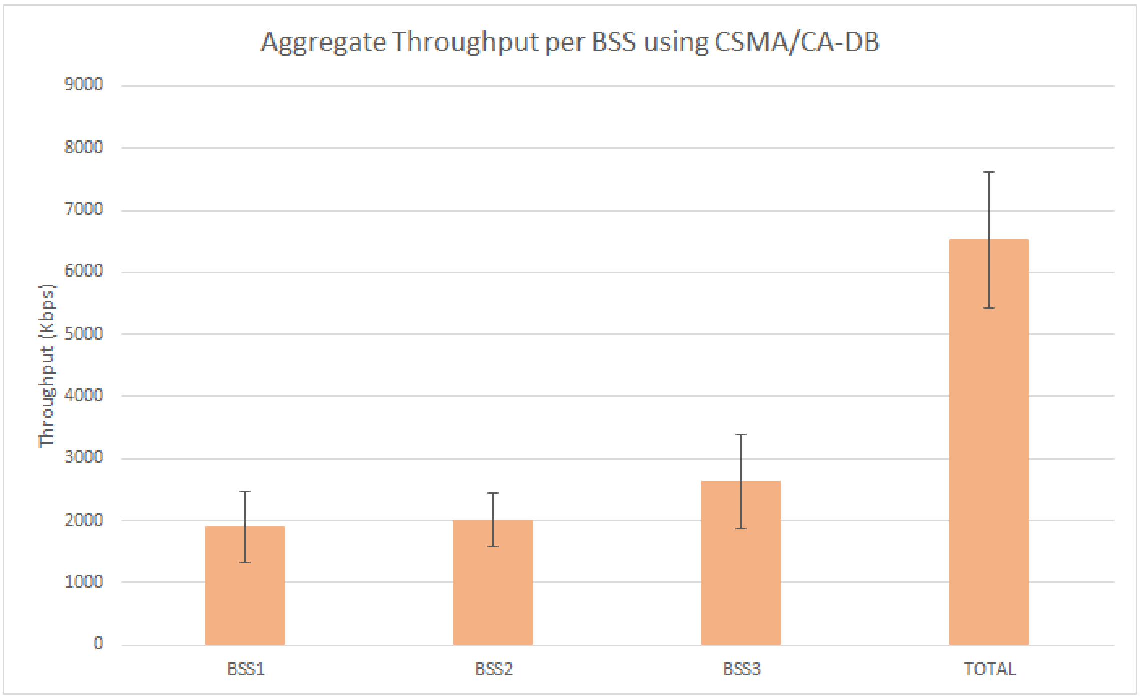


Figure 5.25: Aggregate throughput per BSS using CSMA/CA-DB

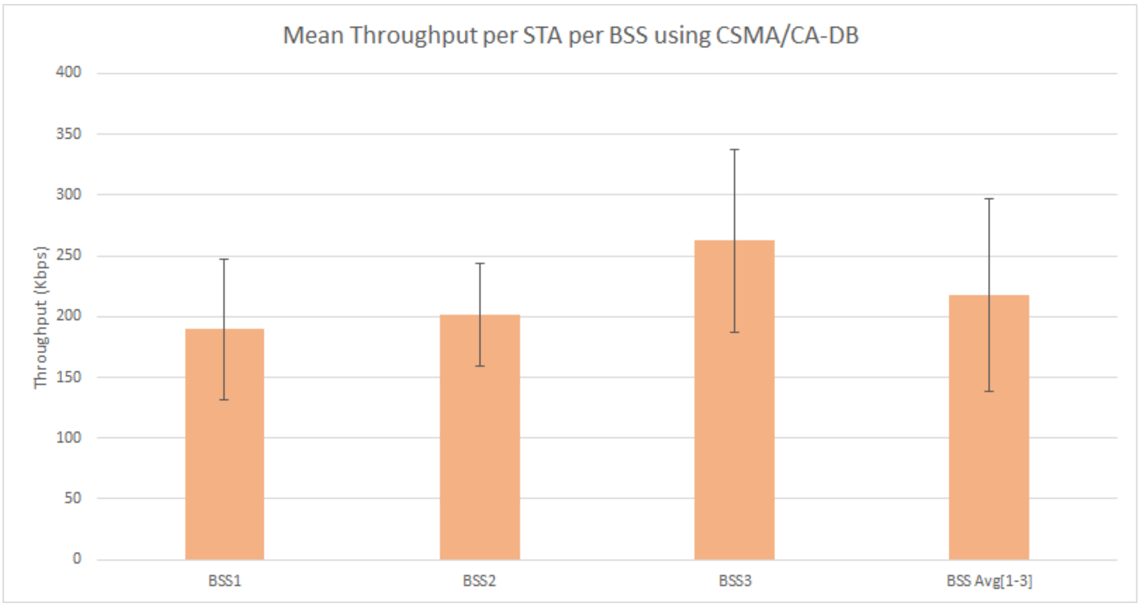


Figure 5.26: Mean throughput per STA at each BSS using CSMA/CA-DB

network. The routing protocol used some of the bandwidth available, which was not seen in the simple BSS scenario.

Figure 5.27 shows the total bytes received at each BSS under study.

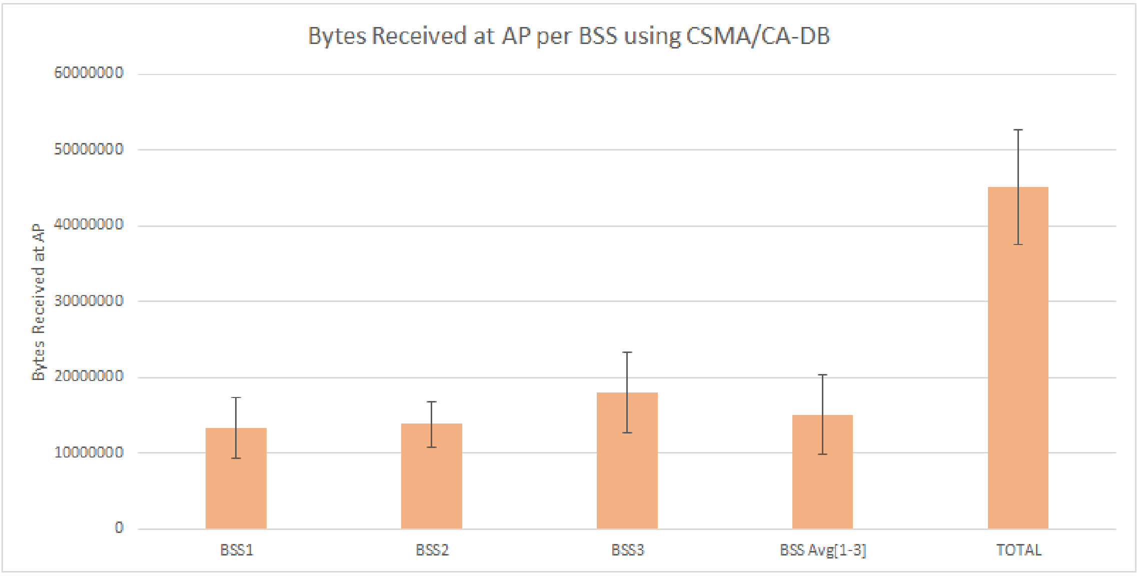


Figure 5.27: Bytes received at AP per BSS using CSMA/CA-DB

Following the results of aggregate and mean throughput obtained previously, it is possible to see that CSMA/CA-DB also shows a fairness between all BSSs in terms of bandwidth share.

Among the detected collisions and packet drops, the same reasons for the PHY layer were detected, as it happened with the simple BSS scenario: SNIR fail, PLCP header reception failed or already in Rx mode or already in Tx mode. However, MAC layer drops were identified this time: At the AP level frames not targeted to the AP were dropped, which means the AP could hear frames not meant to its own BSS; at the STA level when receiving frames while the STA was not associated or when receiving a DATA frame not from the BSS the STA is associated with, which means the STA could hear that pertained to another BSS. Figure 5.28 shows the packet loss results obtained in the simulations on each of the apartments studied.

Again, most packets are lost due to collisions and interference and the PHY layer level. In this simulation, the packets that are lost at the MAC layer level are also very small when compared to those lost at the PHY layer level.

### 5.4.3 CSMA/CA and CSMA/CA-DB Comparison for the Building Scenario

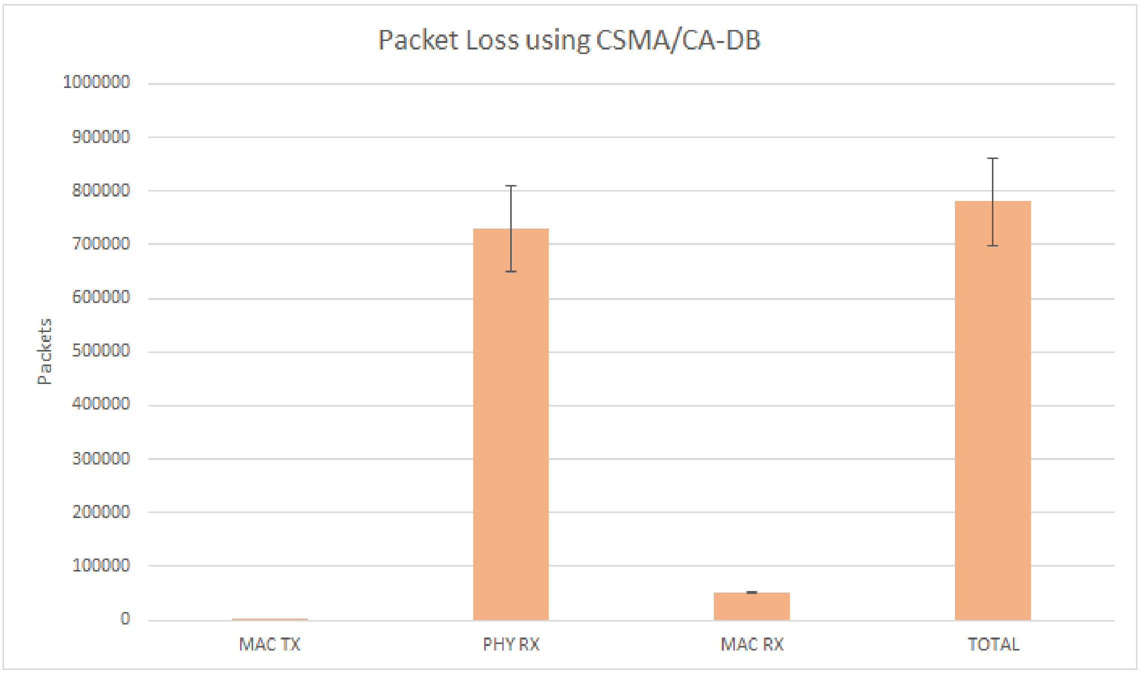


Figure 5.28: Packet loss using CSMA/CA-DB

Figure 5.29 shows the comparison between the two MAC layer protocol for the building scenario in terms of aggregate throughput results obtained at each BSS, plus the total aggregate value.

CSMA/CA-DB outperforms CSMA/CA by some margin in a dense multi apartment scenario in terms of aggregate throughput. Figure 5.30 shows the comparison between the two protocols in terms of mean throughput at each BSS and the fourth column shows the comparison for the average BSS values.

As it was stated for the aggregate throughput results, the mean throughput follows the exact same trend.

Figure 5.31 shows the comparison between the two protocols in terms of bytes received at the APs.

This graph follows the results obtained for the aggregate throughput and the mean throughput.

Figure 5.32 shows the comparison between the two protocols for packet loss results.

The results for MAC TX losses can't be seen on the graph for being too small. MAC RX values are close, though CSMA/CA-DB has some more losses. PHY RX are higher for CSMA/CA-DB

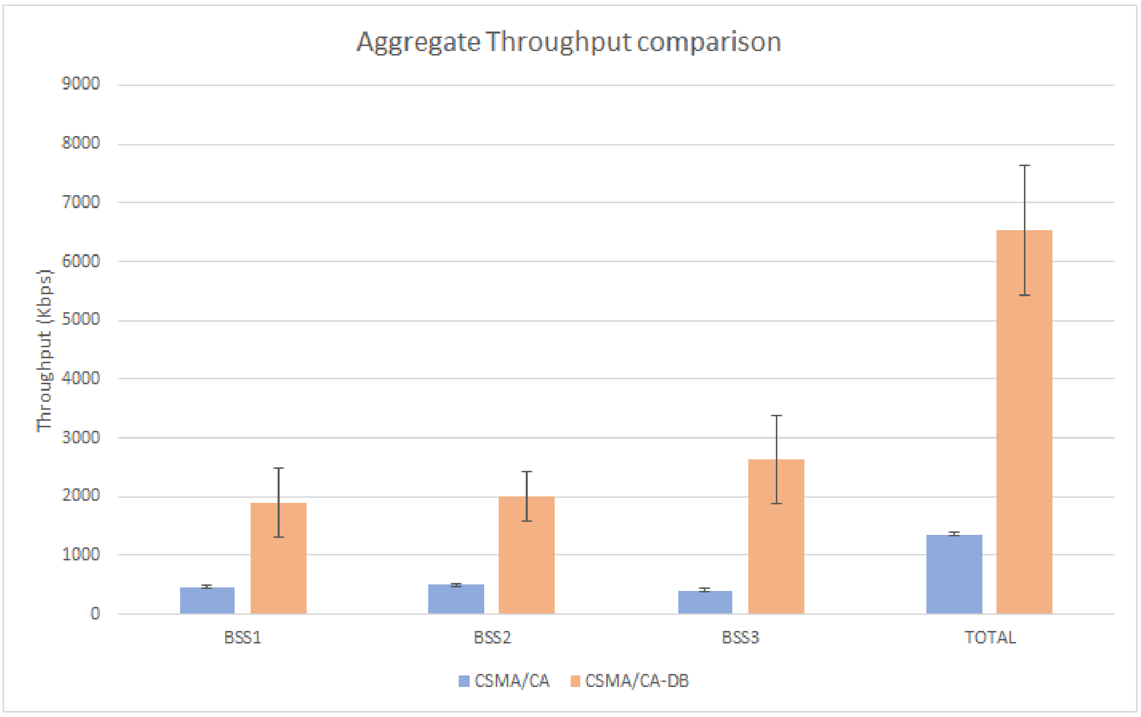


Figure 5.29: Aggregate throughput comparison at each BSS

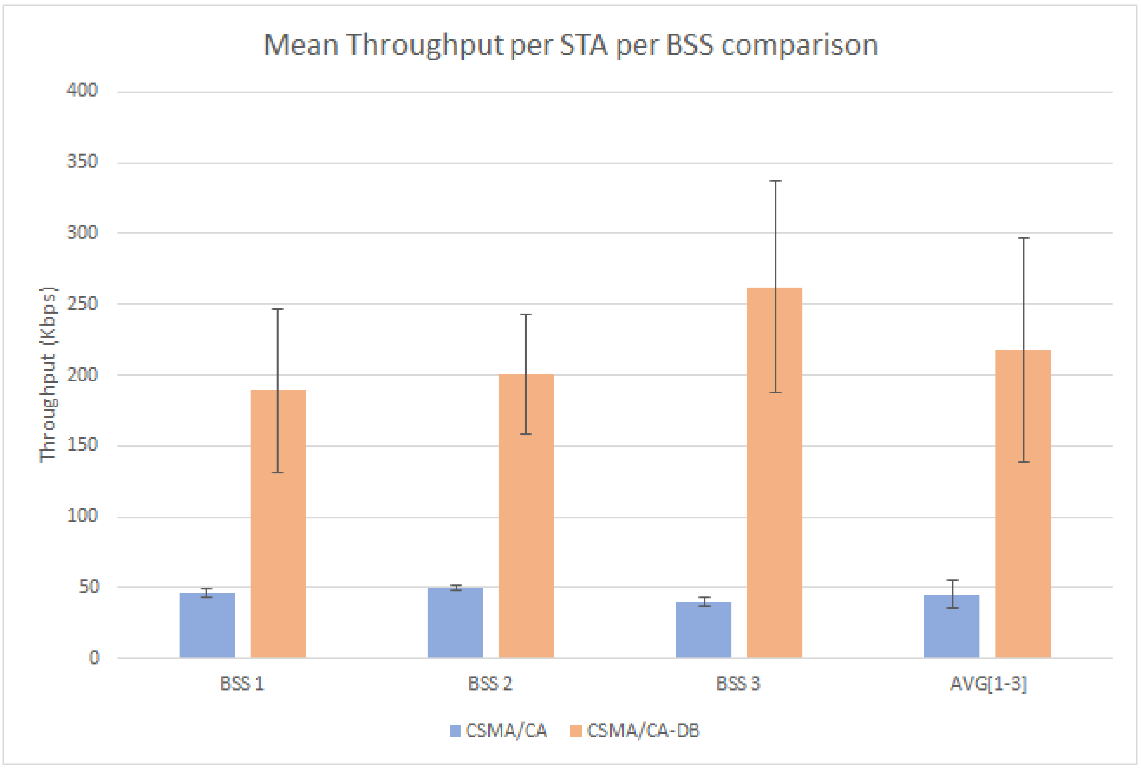


Figure 5.30: Mean throughput per STA comparison at each BSS

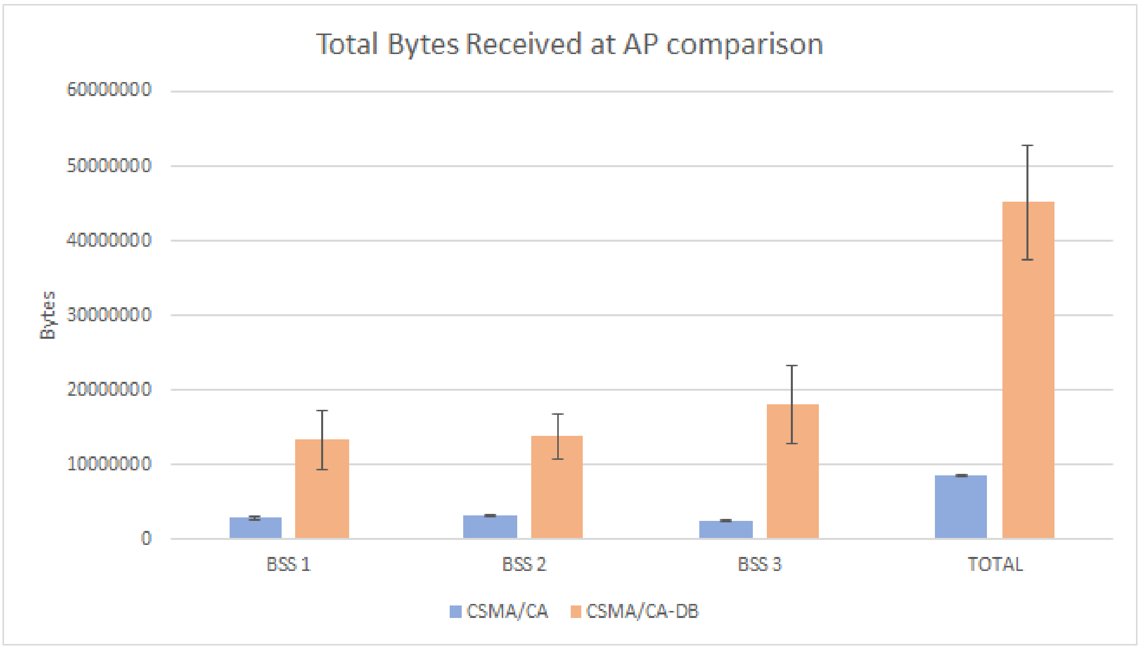


Figure 5.31: Total bytes received comparison at each BSS

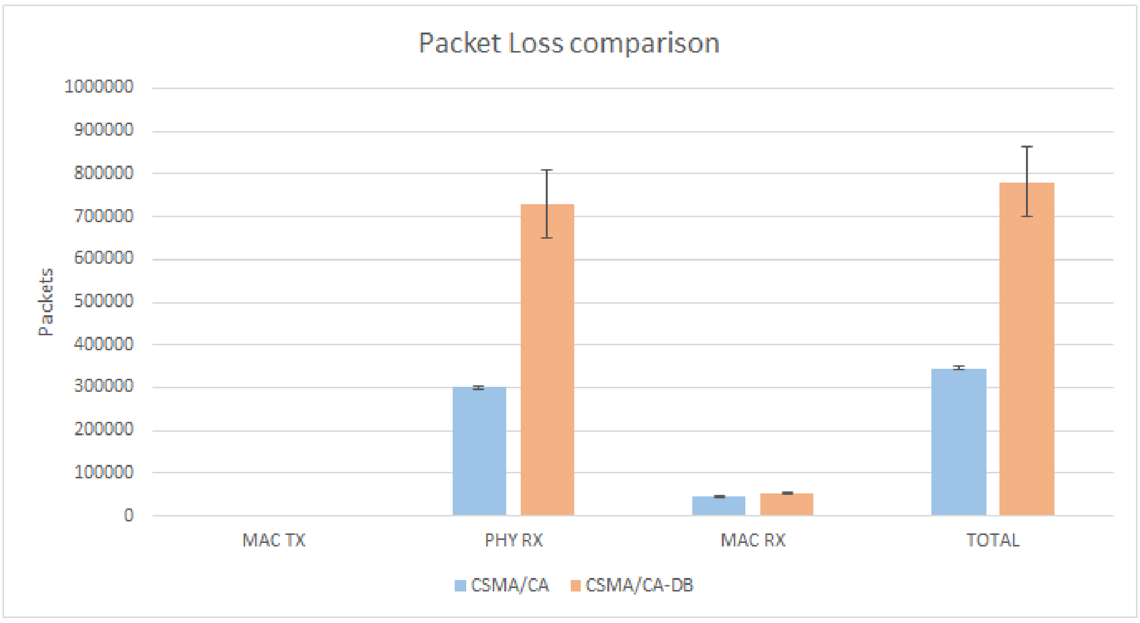


Figure 5.32: Packet loss comparison

than CSMA/CA in this scenario. However, the same occurred for the simple BSS scenario, for 30 STAs (the same number of STAs present at the three apartment on channel 1 under study). The same conclusion can be drawn here: the results in aggregate throughput show a better performance and a lower idleness of the channel for CSMA/CA-DB. The higher number of packet loss is due to the increase of transmission opportunities when compared to CSMA/CA, where the binary exponential backoff mechanism increases the idleness of the channel and reduces the transmission opportunities.

# Chapter 6

# Conclusions

## 6.1 Conclusions obtained from simulation results

The conclusions that can be drawn from the work developed to implement the new MAC layer protocol can be listed as follows:

1. As seen, channel capacity for both protocols is not affected in a 30 meters radius.
2. CSMA/CA-DB has a higher channel capacity than CSMA/CA.
3. For the simple BSS scenario:
   (a) CSMA/CA-DB shows better aggregate throughput from 1 to 30 STAs;
   (b) The mean throughput per STA is higher in CSMA/CA-DB;
   (c) Total bytes received at end node (AP) are higher using CSMA/CA-DB;
   (d) Packet loss is close to zero in CSMA/CA-DB up to 6 STAs. From 9 to 15 STAs, the number of lost packets is close to CSMA/CA. From 20 to 30 STAs, the number of lost packets is higher in CSMA/CA-DB. However, throughput is still higher which means STAs have more transmission opportunities (controlled contention window size rather than exponential);
   (e) The aggregate throughput comparison, for a 99% confidence interval, shows very good improvements for CSMA/CA-DB.
4. For the building scenario:
   (a) The comparison shows better results for CSMA/CA-DB in terms of aggregate throughput and mean throughput per BSS.

## 6.2 Future work

As future work, it would be interesting to study some enhancements, different scenarios, technologies and improve the proposed solution to be able to implement on real devices.

In the ns-3 source code used to implement the protocol, including the next backoff in the ACK frame, instead of using a global variable would be the first improvement, as it would permit to assess the impact of the extra byte on the Wifi channel. Also, finding a way to store a scheduling table individually at each AP device on ns-3 is an important improvement, instead of using a global variable. This would need the use of permanent storage, either by file or database, which would affect the algorithm performance.

Using a deterministic backoff for all types of frames is the next step to try to achieve in terms of the protocol. Several different types of collisions detected on the more dense simulation scenarios could be avoided if other frames were accounted when the AP determines the next backoff for the STAs.

It is important to test all the IEEE HEW scenarios described in [20], including the original "residential" version and take statistics for all the BSSs. The mobility scenarios have also to be tested, where, in this cases, the "opportunity window" periodicity and size is of the utmost importance and would need some studying in order to find the optimal values.

Studying possible improvements at the traffic level, implementing QoS for urgent traffic sources or more demanding sources. This could be achieved by allowing N immediate retransmissions for QoS traffic sources when an ACK timeout would occur, instead of having to wait for an "opportunity window". Using real traffic patterns is another improvement needed, testing not only TCP traffic but also UDP, with variable packet numbers (for instance, testing multimedia or real time traffic or file transfers).

It is important to test the protocol using the most recent Wifi standards, such as IEEE 802.11n and IEEE 802.11ac, which means the protocol would have to be optimized for frame aggregation and block-ACK. This would involved a study on how long can an STA get a hold of the medium, how big of a part can aggregation play in the DATA packets exchange.

Another detail worth of studying, which was pointed during the results chapter, is changing the default $\alpha$ value of the TCP RTT estimator to reduce TCP retransmissions.

Including legacy STAs in the protocol, study the energy impact of the new protocol on both APs and STAs and later implementation and tests on real device are other components for future work.

# Appendix A

# Appendix 1

## A.1 Code for ns-3 simple BSS simulation

Listing A.1: ns-3 simple BSS scenario

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <cstdlib>

#include "ns3/packet.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/global-route-manager.h"
#include "ns3/mobility-module.h"
#include "ns3/netanim-module.h"
#include "ns3/assert.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/wifi-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mac48-address.h"
#include "ns3/nstime.h"
#include "ns3/schedulingtable.h"
#include "ns3/config-store-module.h"

```

```
using namespace std;
using namespace ns3;

uint32_t MacTxDropCount, PhyTxDropCount,
      PhyRxDropCount, MacRxDropCount;

//callbacks to count packet drops at
//MAC layer level
void MacTxDrop(Ptr<const Packet> p)
{
      MacTxDropCount++;
}

void MacRxDrop(Ptr<const Packet> p)
{
      MacRxDropCount++;
}

//callbacks to count packet drops at
//PHY layer level
void PhyTxDrop(Ptr<const Packet> p)
{
      PhyTxDropCount++;
}


void PhyRxDrop(Ptr<const Packet> p)
{
      PhyRxDropCount++;
}

NS_LOG_COMPONENT_DEFINE ("simple_BSS");

int main (int argc, char *argv[])
{
   LogComponentEnable ("simple_BSS", LOG_LEVEL_ERROR);
   //Seed for this scenario was set to 2
   RngSeedManager::SetSeed (2);
   //a different run for each simulation attempt
   RngSeedManager::SetRun(1);
```

```
double simulationTime = 60; //seconds
int k = 1;
//the number of STAs (1 to 30)
int sta = 1;
uint32_t payloadSize;
payloadSize = 1448; //bytes
Config::SetDefault ("ns3::TcpSocket::SegmentSize",
  UintegerValue (payloadSize));
NodeContainer wifiStaNode;
wifiStaNode.Create (sta);
NodeContainer wifiApNode;
wifiApNode.Create (1);
YansWifiChannelHelper channel =
  YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());
// Set guard interval
phy.Set ("ShortGuardEnabled", BooleanValue (k));
WifiMacHelper mac;
//by default, WIFI_PHY_STANDARD_80211a
WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211g);
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
       "Ssid", SsidValue (ssid),
       "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevice;
staDevice = wifi.Install (phy, mac, wifiStaNode);
mac.SetType ("ns3::ApWifiMac",
       "Ssid", SsidValue (ssid));
NetDeviceContainer apDevice;
apDevice = wifi.Install (phy, mac, wifiApNode);
// mobility.
MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc =
       CreateObject<ListPositionAllocator> ();
positionAlloc->Add (Vector (0.0, 0.0, 0.0)); //AP
//STAs positions
positionAlloc->Add (Vector (1.0, 0.0, 0.0)); //1
```

```
/* positionAlloc->Add (Vector (5.2, 4.0, 0.0));
   positionAlloc->Add (Vector (-10, 10, 0.0));
   positionAlloc->Add (Vector (-6.3, -1.4, 0.0));
   positionAlloc->Add (Vector (7.2, 2.5, 0.0)); //5
   positionAlloc->Add (Vector (-1.0, 5.1, 0.0));
   positionAlloc->Add (Vector (-5.0, 4.9, 0.0));
   positionAlloc->Add (Vector (10.0, 10.0, 0.0));
   positionAlloc->Add (Vector (-2.3, 0.4, 0.0));
   positionAlloc->Add (Vector (4.1, 6.3, 0.0)); //10
   positionAlloc->Add (Vector (1.7, -6.5, 0.0));
   positionAlloc->Add (Vector (-5.9, -9.9, 0.0));
   positionAlloc->Add (Vector (-7.3, -6.0, 0.0));
   positionAlloc->Add (Vector (3.8, -0.2, 0.0));
   positionAlloc->Add (Vector (4.5, -4.5, 0.0)); //15
   positionAlloc->Add (Vector (1.2, -1.2, 0.0));
   positionAlloc->Add (Vector (-5.2, -4.0, 0.0));
   positionAlloc->Add (Vector (9.9, -9.7, 0.0));
   positionAlloc->Add (Vector (6.3, 1.6, 0.0));
   positionAlloc->Add (Vector (-7.2, -2.5, 0.0));//20
   positionAlloc->Add (Vector (17.4, -15.4, 0.0));
   positionAlloc->Add (Vector (11.3, 15.9, 0.0));
   positionAlloc->Add (Vector (12.2, -12.6, 0.0));
   positionAlloc->Add (Vector (-10.9, 10.9, 0.0));
   positionAlloc->Add (Vector (11.6, 11.9, 0.0)); //25
   positionAlloc->Add (Vector (10.1, -14.1, 0.0));
   positionAlloc->Add (Vector (15.6, 16.9, 0.0));
   positionAlloc->Add (Vector (20.0, -19.0, 0.0));
   positionAlloc->Add (Vector (13.8, -11.8, 0.0));
   positionAlloc->Add (Vector (-11.9, -15.8, 0.0)); //30 */
  mobility.SetPositionAllocator (positionAlloc);
  mobility.SetMobilityModel
        ("ns3::ConstantPositionMobilityModel");
  mobility.Install (wifiApNode);
  mobility.Install (wifiStaNode);
  /* Internet stack*/
  InternetStackHelper stack;
  stack.Install (wifiApNode);
  stack.Install (wifiStaNode);
  Ipv4AddressHelper address;
  address.SetBase ("192.168.1.0", "255.255.255.0");
```

```
Ipv4InterfaceContainer staNodeInterface;
Ipv4InterfaceContainer apNodeInterface;
staNodeInterface = address.Assign (staDevice);
apNodeInterface = address.Assign (apDevice);
/* Setting applications */
ApplicationContainer serverApp, sinkApp, sinkApp1;
uint16_t port = 50000;
//PACKET SINK ON STAs
for(int l=0; l<sta; l++)
{
  PacketSinkHelper sink ("ns3::TcpSocketFactory",
     InetSocketAddress(Ipv4Address::GetAny(), port));
  //same sink installed in all nodes
  ApplicationContainer apps_sink =
     sink.Install(wifiStaNode.Get(l));
  apps_sink.Start(Seconds(0.0));
  apps_sink.Stop(Seconds (simulationTime+1));
}
//packet sink on AP
PacketSinkHelper sink ("ns3::TcpSocketFactory",
      InetSocketAddress(Ipv4Address::GetAny(), port));
ApplicationContainer apps_sink =
      sink.Install(wifiApNode.Get(0));
apps_sink.Start(Seconds(0.0));
apps_sink.Stop(Seconds (simulationTime+1));
//Setup CBR Traffic Sources
for (int l = 0; l< sta; l++)
{
  Ptr<UniformRandomVariable> x =
     CreateObject<UniformRandomVariable>();
  x -> SetAttribute("Min", DoubleValue(0));
  x -> SetAttribute("Max", DoubleValue(1));
  double rn = x -> GetValue();
  //get AP
  Ptr<Node> n = wifiApNode.Get(0);
  Ptr<Ipv4> ipv4 = n -> GetObject<Ipv4>();
  Ipv4InterfaceAddress ipv4_int_addr =
          ipv4->GetAddress(1,0);
  Ipv4Address ip_addr =
          ipv4_int_addr.GetLocal();
```

```
    OnOffHelper onoff ("ns3::TcpSocketFactory",
        InetSocketAddress(ip_addr, port));
    onoff.SetAttribute ("OnTime",
StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
    onoff.SetAttribute ("OffTime",
StringValue ("ns3::ConstantRandomVariable[Constant=0]"));
onoff.SetAttribute ("PacketSize", UintegerValue (payloadSize));
onoff.SetAttribute ("DataRate", DataRateValue (100000000));
    //traffic flows from node [i] to AP
    ApplicationContainer apps =
        onoff.Install(wifiStaNode.Get(1));
    //traffic sources are installed on all nodes
    apps.Start (Seconds (1.0 + rn));
    apps.Stop (Seconds (simulationTime + 1));
  }
  Ipv4GlobalRoutingHelper::PopulateRoutingTables();
  // Tracing
  /*phy.EnablePcap ("simpleBSS_apDevice", apDevice);
  phy.EnablePcap ("simpleBSS_staDevice", staDevice);*/
  //Add Flow Monitor to each node
  Ptr<FlowMonitor> flowmon;
  FlowMonitorHelper flowmonHelper;
  flowmon = flowmonHelper.InstallAll();

  //Trace Collisions
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*
/$ns3::WifiNetDevice/Mac/MacTxDrop", MakeCallback(&MacTxDrop));
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*
/$ns3::WifiNetDevice/Phy/PhyRxDrop", MakeCallback(&PhyRxDrop));
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*
/$ns3::WifiNetDevice/Phy/PhyTxDrop", MakeCallback(&PhyTxDrop));
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*
/$ns3::WifiNetDevice/Mac/MacRxDrop", MakeCallback(&MacRxDrop));

  Simulator::Stop (Seconds (simulationTime + 1));
  Simulator::Run ();
  //Output FlowMonitor to an XML file
  flowmon->SerializeToXmlFile("simple_BSS.xml", true, true);
  Simulator::Destroy ();
  return 0;
```

```
}
```

## A.2 Code for ns-3 building simulation

Listing A.2: ns-3 building scenario

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <cstdlib>

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/global-route-manager.h"
#include "ns3/mobility-module.h"
#include "ns3/netanim-module.h"
#include "ns3/assert.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/wifi-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/buildings-module.h"
#include "ns3/dsr-module.h"
#include "ns3/dsdv-module.h"
#include "ns3/aodv-module.h"
#include "ns3/schedulingtable.h"

using namespace std;
using namespace ns3;

uint32_t MacTxDropCount, PhyTxDropCount,
      PhyRxDropCount, MacRxDropCount;

void MacTxDrop(Ptr<const Packet> p)
{
      MacTxDropCount++;
```

```
}


void PhyTxDrop(Ptr<const Packet> p)
{
      PhyTxDropCount++;
}


void PhyRxDrop(Ptr<const Packet> p)
{
      PhyRxDropCount++;
}

void MacRxDrop(Ptr<const Packet> p)
{
      MacRxDropCount++;
}


NS_LOG_COMPONENT_DEFINE ("building");

int main (int argc, char *argv[])
{
   CommandLine cmd;
   cmd.Parse (argc, argv);
   RngSeedManager::SetSeed (6);
   RngSeedManager::SetRun(1);
   double simulationTime = 60; //seconds
   double setTime = 30; //seconds
   int k = 1;
   uint32_t payloadSize;
   payloadSize = 1448; //bytes
   Config::SetDefault ("ns3::TcpSocket::SegmentSize",
      UintegerValue (payloadSize));
   //Channel 1
   NodeContainer wifiStaNode1;
   wifiStaNode1.Create (10);
   NodeContainer wifiApNode1;
   wifiApNode1.Create (1);
```

```
NodeContainer wifiStaNode2;
wifiStaNode2.Create (10);
NodeContainer wifiApNode2;
wifiApNode2.Create (1);
NodeContainer wifiStaNode3;
wifiStaNode3.Create (10);
NodeContainer wifiApNode3;
wifiApNode3.Create (1);
//Channel 1
YansWifiChannelHelper channel1 =
  YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy1 =
  YansWifiPhyHelper::Default ();
phy1.SetChannel (channel1.Create ());
// Set guard interval
phy1.Set ("ShortGuardEnabled", BooleanValue (k));
//Create 3 SSids, one for each BSS and 3 mac layers
WifiMacHelper mac1;
WifiHelper wifi1;

Ssid ssid;
wifi1.SetStandard (WIFI_PHY_STANDARD_80211g);
wifi1.SetRemoteStationManager ("ns3::AarfWifiManager");
NetDeviceContainer staDevice1;
NetDeviceContainer apDevice1;
ssid = Ssid ("ns-3-ssid-1");
mac1.SetType ("ns3::StaWifiMac",
  "Ssid", SsidValue (ssid),
  "ActiveProbing", BooleanValue (false));
staDevice1 = wifi1.Install (phy1, mac1, wifiStaNode1);
mac1.SetType ("ns3::ApWifiMac",
  "Ssid", SsidValue (ssid));
apDevice1 = wifi1.Install (phy1, mac1, wifiApNode1);

//Mac 2
WifiMacHelper mac2;
NetDeviceContainer staDevice2;
NetDeviceContainer apDevice2;

ssid = Ssid ("ns-3-ssid-2");
```

```
mac2.SetType ("ns3::StaWifiMac",
  "Ssid", SsidValue (ssid),
  "ActiveProbing", BooleanValue (false));
staDevice2 = wifi1.Install (phy1, mac2, wifiStaNode2);
mac2.SetType ("ns3::ApWifiMac",
  "Ssid", SsidValue (ssid));
apDevice2 = wifi1.Install (phy1, mac2, wifiApNode2);

//Mac 3
WifiMacHelper mac3;
NetDeviceContainer staDevice3;
NetDeviceContainer apDevice3;

ssid = Ssid ("ns-3-ssid-3");
mac3.SetType ("ns3::StaWifiMac",
  "Ssid", SsidValue (ssid),
  "ActiveProbing", BooleanValue (false));
staDevice3 = wifi1.Install (phy1, mac3, wifiStaNode3);
mac3.SetType ("ns3::ApWifiMac",
  "Ssid", SsidValue (ssid));
apDevice3 = wifi1.Install (phy1, mac3, wifiApNode3);

//building scenario
double x_min = 0.0;
double x_max = 100.0;
double y_min = 0.0;
double y_max = 20.0;
double z_min = 0.0;
double z_max = 3.0;
Ptr<Building> b = CreateObject <Building> ();
b->SetBoundaries (Box (x_min, x_max,
  y_min, y_max, z_min, z_max));
b->SetBuildingType (Building::Residential);
b->SetExtWallsType (Building::ConcreteWithWindows);
b->SetNFloors (1);
b->SetNRoomsX (10);
b->SetNRoomsY (2);
double z_shift = 1.5;
// mobility channel 1
MobilityHelper mobility1;
```

```
Ptr<ListPositionAllocator> positionAlloc1 =
  CreateObject<ListPositionAllocator> ();
//Channel 1 APs positions
positionAlloc1->Add (Vector(15.0, 15.0, 1.5));
double x_shift = 10.0;
double y_shift = 10.0;
positionAlloc1->Add (Vector(1.0+ x_shift,
  1.0+y_shift, z_shift));
positionAlloc1->Add (Vector(3.0+ x_shift,
  3.0+y_shift, z_shift));
positionAlloc1->Add (Vector(2.0+ x_shift,
  5.0+y_shift, z_shift));
positionAlloc1->Add (Vector(3.0+ x_shift,
  7.0+y_shift, z_shift));
positionAlloc1->Add (Vector(1.0+ x_shift,
  9.0+y_shift, z_shift));
positionAlloc1->Add (Vector(9.0+ x_shift,
  1.0+y_shift, z_shift));
positionAlloc1->Add (Vector(7.0+ x_shift,
  3.0+y_shift, z_shift));
positionAlloc1->Add (Vector(8.0+ x_shift,
  5.0+y_shift, z_shift));
positionAlloc1->Add (Vector(7.0+ x_shift,
  7.0+y_shift, z_shift));
positionAlloc1->Add (Vector(9.0+ x_shift,
  9.0+y_shift, z_shift));
mobility1.SetPositionAllocator (positionAlloc1);
mobility1.SetMobilityModel
  ("ns3::ConstantPositionMobilityModel");
mobility1.Install (wifiApNode1);
mobility1.Install (wifiStaNode1);
BuildingsHelper::Install (wifiApNode1);
BuildingsHelper::Install (wifiStaNode1);
//same code is applied to other BSSs

//after placing all the nodes in the simulation
//to list all the nodes and determine where each
//user is located, the floor and number inside
//the building
BuildingsHelper::MakeMobilityModelConsistent ();
```

```
//AODV routing protocol
AodvHelper aodv;
/* Internet stack channel 1*/
InternetStackHelper stack1, stack1_1;
stack1.SetRoutingHelper (aodv);
stack1_1.SetRoutingHelper (aodv);
stack1.Install (wifiApNode1);
stack1_1.Install (wifiStaNode1);
//same applied to other BSSs
Ipv4AddressHelper address1, address2, address3;
address1.SetBase ("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer staNodeInterface1;
Ipv4InterfaceContainer apNodeInterface1;

staNodeInterface1 = address1.Assign (staDevice1);
apNodeInterface1 = address1.Assign (apDevice1);
//same code applied to other BSSs
/* Setting applications */
ApplicationContainer serverApp, sinkApp, sinkApp1;
ApplicationContainer apps_sink1, apps_sink2, apps_sink3;
uint16_t port = 50000;
PacketSinkHelper sink ("ns3::TcpSocketFactory",
InetSocketAddress(Ipv4Address::GetAny(), port));
//same sink installed in all nodes
apps_sink1 = sink.Install(wifiStaNode1);
apps_sink1.Start(Seconds(0.0));
apps_sink1.Stop(Seconds (setTime+simulationTime+1));
apps_sink1 = sink.Install(wifiApNode1);
apps_sink1.Start(Seconds(0.0));
apps_sink1.Stop(Seconds (setTime+simulationTime+1));
apps_sink2 = sink.Install(wifiStaNode2);
apps_sink2.Start(Seconds(0.0));
apps_sink2.Stop(Seconds setTime+(simulationTime+1));
apps_sink2 = sink.Install(wifiApNode2);
apps_sink2.Start(Seconds(0.0));
apps_sink2.Stop(Seconds (setTime+simulationTime+1));
apps_sink3 = sink.Install(wifiStaNode3);
apps_sink3.Start(Seconds(0.0));
apps_sink3.Stop(Seconds (setTime+simulationTime+1));
apps_sink3 = sink.Install(wifiApNode3);
```

```
  apps_sink3.Start(Seconds(0.0));
  apps_sink3.Stop(Seconds (setTime+simulationTime+1));
  //Setup CBR Trafic Sources for channel 1
  // get AP
  Ptr<Node> n = wifiApNode1.Get(0); //change to 1
  Ptr<Ipv4> ipv4 = n -> GetObject<Ipv4>();
  Ipv4InterfaceAddress ipv4_int_addr = ipv4->GetAddress(1,0);
  Ipv4Address ip_addr = ipv4_int_addr.GetLocal();
  for(g=0; g<10;g++){ //change to 10
    Ptr<UniformRandomVariable> xm =
        CreateObject<UniformRandomVariable>();
    xm -> SetAttribute("Min", DoubleValue(0));
    xm -> SetAttribute("Max", DoubleValue(1));
    double rm = xm -> GetValue();

    OnOffHelper onoff ("ns3::TcpSocketFactory",
        InetSocketAddress(ip_addr, port));
    onoff.SetAttribute ("OnTime",
StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
    onoff.SetAttribute ("OffTime",
StringValue ("ns3::ConstantRandomVariable[Constant=0]"));
    onoff.SetAttribute ("PacketSize",
        UintegerValue (payloadSize));
    onoff.SetAttribute ("DataRate",
        DataRateValue (25000000));

    //traffic flows from node to AP
    ApplicationContainer apps =
        onoff.Install(wifiStaNode1.Get(g));
    //traffic sources are installed on all nodes
    apps.Start (Seconds (1.0 + rm));
    apps.Stop (Seconds (setTime+simulationTime + 1));
  }
  //same code to other BSSs

  //Add Flow Monitor to each node
  Ptr<FlowMonitor> flowmon;
  FlowMonitorHelper flowmonHelper;
  flowmon = flowmonHelper.InstallAll();

  //Trace Collisions
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*
/$ns3::WifiNetDevice/Mac/MacTxDrop", MakeCallback(&MacTxDrop));
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*
/$ns3::WifiNetDevice/Phy/PhyRxDrop", MakeCallback(&PhyRxDrop));
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*
/$ns3::WifiNetDevice/Phy/PhyTxDrop", MakeCallback(&PhyTxDrop));
  Config::ConnectWithoutContext("/NodeList/*/DeviceList/*
/$ns3::WifiNetDevice/Mac/MacRxDrop", MakeCallback(&MacRxDrop));
  Simulator::Stop (Seconds (setTime+simulationTime + 1));
  Simulator::Run ();
  //Output FlowMonitor to an XML file
  flowmon->SerializeToXmlFile("building.xml", true, true);
  Simulator::Destroy ();
  return 0;
}
```

# References

[1] Boris Bellalta. Ieee 802.11ax: High-efficiency wlans. *IEEE Wireless Communications*, pages 38–46, October 2016.

[2] R. Zhang et al. Chapter 2 mac protocols for high data-rate wireless networks. *Resource Management Services in High Data Rate Wireless Netowrks*, pages 9–42, January 2017.

[3] Purdue University Department of Computer Science. Ieee 802.11 mac, November 2015. `https://www.cs.purdue.edu/homes/park/cs536-wireless-3.pdf`.

[4] Tho Le-Ngoc Quang-Dung Ho, Daniel Tweed. *IEEE 802.11/Wi-Fi Medium Access Control: An Overview*. Springer, First edition, 2017.

[5] Andrew T. Campbell. Mac layer - wifi case study, October 2016. `http://www.cs.dartmouth.edu/~campbell/cs60/mac.pdf`.

[6] Emma Fitzgerald Christian Nyberg Basuki E. Priyanto Andre Jonsson, David Akerman and Kare Agardh. Modeling, implementation and evaluation of ieee 802.11ac in ns-3 for enterprise networks. *Wireless Days*, pages 1–6, November 2016.

[7] Boris Bellalta Jaume Barcelo Miquel Oliver Luis Sanabria-Russo, Azadeh Faridi. Future evolution of csma protocols for the ieee 802.11 standard, March 2013. `https://arxiv.org/pdf/1303.3734.pdf`.

[8] NS-3. Wifi model overview in ns-3, February 2017. Available at `https://www.nsnam.org/docs/models/html/wifi-design.html#overview-of-the-model`.

[9] Savvius. 802.11 wlan packets and protocols, June 2017. `https://www.savvius.com/resources/compendium/wireless_lan/wlan_packets`.

[10] IEEE Computer Science. Wireless lan medium access control (mac) and physical layer (phy) specifications. Technical report, IEEE Standards Association, March 2012.

[11] Boris Bellalta. Ieee 802.11ax: High-efficiency wlans, October 2016. `https://arxiv.org/abs/1501.01496`.

[12] Kwang-Cheng Chen Der-Jiunn Deng and Rung-Shiang Cheng. Ieee 802.11ax: Next generation wireless local area networks. *2014 10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE)*, pages 77–82, January 2014.

[13] Jim Lansford Raja Banerjea and Naveen Kakani. A simplified simultaneous transmit and receive mac proposal, March 2014. `https://www.slideideas.net/a-simplified-simultaneous-transmit-and-receive-mac-proposal`.

[14] Nicolò Facchi Francesco Gringoli Luis Sanabria-Russo, Boris Bellalta. Collision-free operation in high density wlan deployments, July 2016. `https://arxiv.org/pdf/1607.08138.pdf`.

[15] Chung-Ming Huang Rung-Shiang Cheng. Collision detect and avoidance media access mechanism for next generation 802.11ax networks, November 2015. `http://ieeexplore.ieee.org/document/7332566/`.

[16] NS-3 Community. Current development, 2017. `https://www.nsnam.org/wiki/Current_Development#Current_development_for_main_trunk_of_ns-3`.

[17] Daniel Lertpratchya. Wi-fi module in ns-3, February 2017. `https://www.nsnam.org/tutorials/consortium14/ns-3-training-session-5.pdf`.

[18] ns 3 Manual. Random variables - manual, April 2017. Last access on 2017/04/06. URL: `https://www.nsnam.org/docs/manual/html/random-variables.html`.

[19] ns 3 Models. Flow monitor - model description, April 2017. Last access on 2017/04/06. URL: `https://www.nsnam.org/docs/models/html/flow-monitor.html`.

[20] IEEE 802.11 TGax. Tgax simulation scenarios, November 2015. `https://mentor.ieee.org/802.11/dcn/14/11-14-0980-16-00ax-simulation-scenarios.docx`.

[21] ns 3 API. ns3::yanswifiphy class reference, June 2017. Last access on 2017/06/06. URL: `https://www.nsnam.org/doxygen/classns3_1_1_yans_wifi_phy.html#details`.

[22] Thomas R. Henderson Mathieu Lacage. Yet another network simulator, June 2017. `http://cutebugs.net/files/wns2-yans.pdf`.