Improving User Association in High-Density WLANs using Machine Learning

Carrascosa Zamacois, Marc

Curs 2017-2018

Director: Boris Bellalta Jimenez

GRAU EN ENGINYERIA TELEMÀTICA

upf. Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

Improving User Association in High-Density WLANs using Machine Learning

Marc Carrascosa Zamacois

Bachelor's thesis

Telematics engineering

Escola superior politècnica UPF

2018

Thesis director

Boris Bellalta Jimenez

Acknowledgements

I would like to thank my tutor Boris Bellalta for all the help and encouragement he has given me.

A special thank you to Arnau for making class that much more enjoyable.

And last but not least, to my parents, whose help, encouragement and patience managed to get me through this. I can never thank you enough.

This work has been partially supported by a Gift from the Cisco University Research Program (CG#890107, Towards Deterministic Channel Access in High-Density WLANs) Fund, a corporate advised fund of Silicon Valley Community Foundation.

Abstract

When deploying a Wi-Fi network, it is common to have the coverage area of multiple Access Points (APs) overlapping as to ensure that no area is left without service. This and the ever increasing amount of wireless devices that users carry can create a lot of additional interference to the network. As a result, the optimal AP - end user device association has become quite a challenge. Our aim is to study the current problems in 802.11 networks, such as associating an end user's Wi-Fi station (STA) to an AP only using the received signal strength, which can overcrowd an AP and leave another underused. We will also investigate current efforts to mitigate these issues, as well as design new association schemes that take into consideration the information acquired by the APs and end user STAs to improve user satisfaction, which we will measure in number of successful transmissions, as well as amount of bandwidth obtained by each STA. We will attempt to do this by using machine learning algorithms that will dynamically improve the association of STAs over time.

Resum

És habitual al fer un desplegament de xarxes Wi-Fi que les àrees de cobertura de diversos Punts d'Accés (APs) coincideixin per tal d'assegurar que es pugui aconseguir una connexió en qualsevol punt del desplegament. Si a això li sumem que el nombre de dispositius sense fils per persona augmenta cada vegada més, trobem que la quantitat d'interferència que hi ha en una xarxa és molt alta. Com a conseqüència, trobar una associació òptima entre l'usuari i l'AP és una tasca complicada. El nostre objectiu és estudiar els problemes actuals en l'associació del IEEE 802.11, un dels quals és fer servir la potència de senyal rebuda com a criteri d'associació, cosa que pot sobrecarregar un AP i deixar-ne un altre amb molt pocs usuaris. També estudiarem el treball que s'està portant a terme actualment per a mitigar aquests problemes i intentarem dissenyar nous mecanismes d'associació que considerin la informació obtinguda pels APs i els terminals finals per a millorar la satisfacció dels usuaris. Pretenem utilitzar algorismes de Machine Learning que milloraran l'associació de la xarxa de manera progressiva.

Resumen

Es habitual al hacer un despliegue de redes Wi-Fi que las áreas de cobertura de varios Puntos de Acceso (APs) coincidan para asegurar que se pueda conseguir una conexión en cualquier punto del despliegue. Si a esto le sumamos que el número de dispositivos inalámbricos por persona aumenta cada vez más, encontramos que la cantidad de interferencia que hay en una red es muy alta. Como consecuencia, encontrar una asociación óptima entre el usuario y el AP es una tarea complicada. Nuestro objetivo es estudiar los problemas actuales en la asociación del IEEE 802.11, uno de los cuales es usar la potencia de señal recibida como criterio de asociación, cosa que puede sobrecargar un AP y dejar otro con muy pocos usuarios. También estudiaremos el trabajo que se está llevando a cabo actualmente para mitigar estos problemas e intentaremos diseñar nuevos mecanismos de asociación que consideren la información obtenida por los APs y los terminales finales para mejorar la satisfacción de los usuarios. Pretendemos utilizar algoritmos de Machine Learning que mejorarán la asociación de la red de manera progresiva.

Contents

Acknowledgements **iii**

List of Figures **x**

List of Tables **xi**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Objectives . . . 2
1.3 Document structure . . . 2

2 Related Work **3**
2.1 User association . . . 3
2.2 HD WLANs . . . 4
2.3 Machine Learning in WLANs . . . 6

3 User association in IEEE 802.11 WLANs **9**
3.1 Introduction . . . 9
3.2 Basic procedure . . . 9
3.3 IEEE 802.11 amendments . . . 11
3.3.a IEEE 802.11i . . . 11
3.3.b IEEE 802.11r . . . 14
3.3.c 802.11ai . . . 15
3.3.d 802.11k . . . 16
3.3.e 802.11ax . . . 16
3.4 AP selection in IEEE 802.11 . . . 16
3.5 Conclusions . . . 17

4 Performance Evaluation of High Density WLANs **19**
4.1 Introduction . . . 19
4.2 System model . . . 19
4.2.a Scenario . . . 19
4.2.b Metrics used . . . 21
4.2.c Simulation approach . . . 22
4.3 Results . . . 24
4.3.a Multiple Overlapping WLANs . . . 24
4.3.b Increasing nodes in the network . . . 26

4.3.c Capture effect . . . 30
4.3.d CCA and power adjustment . . . 35
4.4 Conclusions . . . 44

5 Decentralized AP selection using RL **45**
5.1 Introduction . . . 45
5.2 System Model . . . 45
5.2.a Scenario . . . 45
5.2.b Network throughput model . . . 46
5.2.c Simulation approach . . . 49
5.3 Reinforcement Learning . . . 50
5.3.a Multi armed bandits . . . 50
5.3.b ϵ-greedy . . . 51
5.4 SSF, ϵ-greedy and STA placement . . . 51
5.5 Our modifications to the algorithm . . . 54
5.6 Initial simulation . . . 54
5.7 Different network topologies . . . 59
5.8 Stickiness vs. regular algorithm . . . 61
5.9 Tweaking the algorithm . . . 62
5.9.a Non-Binary rewards . . . 62
5.9.b Exploit vs. inaction . . . 63
5.9.c Comparison . . . 64
5.10 A second look at fixed STA positions . . . 66
5.11 Variable bandwidth . . . 67
5.12 Conclusions . . . 71

6 Conclusions **73**

List of Figures

3.1 Standard 802.11 frame [1] . . . 10
3.2 Message flow for each association type . . . 10
3.3 802.11 time line . . . 11
3.4 802.11i frame flows . . . 13
3.5 802.11r re-association . . . 14
3.6 802.11i re-association with ERP . . . 15
3.7 802.11ai re-association with FILS . . . 15

4.1 Example of a hidden node when using a threshold of 10 dBs . . . 22
4.2 Transmission examples. Successful transmissions in blue and unsuccessful ones in red. . . . 23
4.3 AP configurations . . . 24
4.4 2 APs share the channel . . . 25
4.5 3 APs share the channel . . . 25
4.6 4 APs share the channel . . . 25
4.7 3 APs with different channels . . . 26
4.8 Increasing the amount of STAs in the network . . . 26
4.9 Increasing the amount of APs in the network . . . 27
4.10 Successful vs. unsuccessful transmissions . . . 27
4.11 Hidden and exposed nodes with increasing number of channels . . . 28
4.12 Hidden and exposed nodes with increasing number of channels . . . 28
4.13 Transmissions with 4 APs and increasing number of STAs and channels 29
4.14 Transmissions with 100 STAs and increasing number of APs and channels . . . 29
4.15 Hidden and exposed nodes with 2 APs and 10 STAs . . . 31
4.16 Transmissions with 2 APs and 10 STAs . . . 31
4.17 Hidden and exposed nodes with 3 APs and 10 STAs . . . 31
4.18 Transmissions with 3 APs and 10 STAs . . . 32
4.19 Hidden and exposed nodes with 4 APs and 10 STAs . . . 32
4.20 Transmissions with 4 APs and 10 STAs . . . 32
4.21 Boxplots of the average hidden and exposed nodes with 4 APs and increasing STAs . . . 33
4.22 Transmissions with 4 APs when the threshold and STAs increase . . . 33
4.23 Boxplots of the average hidden nodes with 100 STAs and increasing APs . . . 34
4.24 Boxplots of the average exposed nodes with 100 STAs and increasing APs . . . 34

4.25 Transmissions with 100 STAs when the threshold and APs increase . 35
4.26 Results for 2 APs and 10 STAs 37
4.27 Results for 3 APs and 10 STAs 38
4.28 Hidden and exposed nodes for 4 APs and 10 STAs 38
4.29 Transmissions for 4 APs and 10 STAs 39
4.30 Transmission power for each configuration 39
4.31 Results with 4 APs when we increase the amount of STAs in the network 40
4.32 Transmission power selected when STAs increase 41
4.33 Results with 100 STAs when we increase the amount of APs in the network 42
4.34 Average STA CCA with 100 STAs and increasing APs 43
4.35 Transmission power used with 100 STAs and increasing APs 43

5.1 Structure of the simulation 50
5.2 Node configurations 52
5.3 Percentage of bandwidth obtained 53
5.4 Convergence 56
5.5 Bandwidth and satisfaction achieved 57
5.6 States of the algorithm and STA stickiness vs. bandwidth 58
5.7 Bandwidth achieved with different requirements 59
5.8 Changing the epsilon value 61
5.9 Bandwidth achieved with each association 63
5.10 Bandwidth achieved with each association 63
5.11 Comparison of bandwidth achieved 64
5.12 Jain's fairness 65
5.13 Percentage of bandwidth obtained 66
5.14 Performance when the demands change 68
5.15 Performance of two STAs in a single simulation 69

List of Tables

4.1 Glossary 20
4.2 Simulation parameters 20

5.1 Glossary 46
5.2 Simulation parameters 46
5.3 Notation used 47
5.4 IEEE 802.11ax rates for 20 MHz and one spatial stream 48
5.5 Statistics for each configuration 54
5.6 Statistics for the simulation of 3 APs and 10 STAs 56
5.7 Statistics when changing the amount of APs 60
5.8 Statistics as the ϵ value changes 62
5.9 Statistics for each algorithm 65
5.10 Statistics for each configuration 67
5.11 Statistics for each approach 68
5.12 Convergence intervals for each algorithm 69

Chapter 1

Introduction

1.1 Motivation

High density Wireless Local Area Networks (HD-WLANs) are networks with a high number of Access Points (APs) in a given area. Shopping malls, universities and even entire cities have massive multi-AP deployments giving coverage to hundreds of users, which means that these networks have a different set of challenges than single AP deployments, such as user association.

In the context of IEEE 802.11 WLANs, user association is based on the Strongest Signal First (SSF) mechanism, in which the end user stations (STAs) find the strongest signal available based on their Signal to Noise Ratio (SNR) and associate to whichever AP sent it. This method works properly in single AP networks, but has been shown to be inefficient in multiple-AP deployments [2, 3] by ignoring the AP load, the topology of the network and the other users in it.

Another challenge with HD-WLANs is channel interference. Ideally, each WLAN will be in a different channel and it will only sense the users that belong to that network. In reality, there are few orthogonal channels available, especially in the 2.4 GHz band, which has only 3. A limited number of channels means that interference between APs will not always be avoidable.

Overlapping creates two problems in HD-WLANs:

- **Hidden nodes**: These appear when two nodes on the same channel do not sense each other. Since they are unaware of the other's existence, they will end up transmitting at the same time, creating collisions. Hidden nodes are already present in single AP networks, but in HD-WLANs these collisions can be created by nodes from different networks that are on the same channel, aggravating the problem.
- **Exposed nodes**: If two networks are close to each other and on the same channel, a node from the first network can sense a node from the second one. Then, if the first node senses that the second one is transmitting, it will wait to transmit to not create a collision, but since they are from different networks

their receivers are likely in different places, and it is also likely that those receivers are outside of the range of the interfering node, which means that they could receive simultaneously without suffering from interference. In this case the problem is not a collision, it is a loss in potential throughput.

We will be studying the effects that these problems have on the network performance and possible ways to mitigate them.

For the association problem, we will use Machine Learning (ML) algorithms to make the entire network learn and adapt to its environment over time. We will specifically look at Reinforcement Learning (RL), which has an algorithm learn by taking actions which receive a reward or punishment that serve as knowledge to inform the next decision. Usually the issue is to find a balance between exploring all options to find new information and exploiting the knowledge that has already been accumulated.

1.2 Objectives

The main purpose of this project is to improve the IEEE 802.11 performance in HD-WLANs by intelligently associating STAs to APs. To that end we will:

1. Study the current version of IEEE 802.11, as well as the drafts of future revisions (11ai,11ax), and find what has been done in regards to improving user association.
2. Study Machine Learning techniques to create an algorithm that is able to find the optimal association solution.
3. Create a simulation tool using MATLAB where we can test and evaluate the performance of HD-WLANs.
4. Design and evaluate Machine Learning algorithms for improving user association in HD-WLANs.

1.3 Document structure

This document is organized as follows: in Chapter 2 the state of the art for current association improvement efforts and channel interference is presented. Chapter 3 summarizes the current state of the IEEE 802.11 standard, as well as future amendments. Chapter 4 presents the model used for our simulations and an analysis of the hidden and exposed node problem, to see if they are a problem that can be ignored or needs to be taken into consideration when designing new associations mechanisms. In Chapter 5 the reinforcement learning algorithm used and the model in which it was tested can be found. Finally, our conclusions are discussed in Chapter 6.

Chapter 2

Related Work

2.1 User association

Strongest Signal First (SSF) is the de facto method for Access Point (AP) association in WLANs. However, it has been shown to lead to uneven load and client distribution among APs [3, 4], which leads to poor performance and unfairness situations. Such user behavior analysis shows that the number of users associated to an AP is related to its location. However, peak throughput is not achieved with full users load, since most peaks in the system depend on the needs of individual users. The results in [4] show that 10% of users are responsible for 40 % of the load in most APs.

In order to extend the current SSF-based approach, some works use an extended set of measurements for decision-making. In [5], the hand-off of re-associations is improved by letting the stations (STAs) keep a list of APs found during scanning organized by their Received Signal Strength Indicator (RSSI). With that, AP re-association can be done while skipping scanning. The authors in [6] use probe messages to measure the AP's available bandwidth, in order to decide to which one they want to associate with. Once associated, the STAs periodically check if their individual service requirements are still met. Otherwise, probing is done with the goal of finding a better AP that enhances the individual performance. Similarly, the method presented in [7] uses the beacon frames received at STAs to measure the signal strength of each AP that is in range, thus allowing to estimate the throughput they could achieve after association.

As an alternative to SNR-based methods, works based on performance estimation are gaining attention. In particular, we highlight potential bandwidth [8] and proportional fairness [9, 10] estimation methods to perform AP association. These add some computational time which can be undertaken in the STAs themselves or on a central hub that communicates the decisions to the STAs later. For these cases, the IEEE 802.11r standard could be really useful, since it provides the neighbor report, in which an AP obtains information from other APs and relays it to its STAs, so that they can make more intelligent decisions regarding possible re-associations.

Regarding commercial equipment, there are systems that implement some sort of load balancing and association control. In particular, Cisco's APs implement an aggressive load balancing option in which the AP accepts all association requests until a certain number of nodes are associated [11]. Afterwards, any probe request is replied with a response status code 17, which the STA is expected to honor by attempting association with other APs. A maximum number of code 17 responses per STA can be set, with a maximum of 20, after which the STA is allowed to associate with the original AP. To solve that, Cisco's Lightweight Access Point Protocol (LWAPP) implements load balancing in a more elegant way. Through LWAPP, APs notify their load to STAs, and combining that with their RSSI, they decide which AP is better for association [12].

Notwithstanding, estimation procedures can result very challenging in next generation scenarios, due to large amount of coexisting devices and novel channel access techniques. For the later, and to fit AP association into 802.11ac MU-MIMO systems, the authors of [13] disregard the RSSI values and concentrate on the channel correlation and MIMO-grouping possibilities. On the other hand, the recent popularity of Machine Learning (ML) has opened the door to the appearance of novel AP association methods in complex environments, which we will discuss in Section 2.3.

2.2 HD WLANs

The density of wireless deployments is only increasing as time goes on. With more and more portable devices and demand for bandwidth, the amount of APs in an area has been increasing. Dense WLANs are ubiquitous now, and improving their performance is a popular subject [14, 15, 16].

The work in [14] defines several situations where interference can be found between networks and separates them into two major groups: overlapping cells, where the APs see each other, and co-located cells, where only the STAs see nodes from other networks. A solution is proposed in using different contention windows (CW) for the clients to fix those situations in which Request To Send/Clear To Send (RTS/CTS) is unsuccessful. In [16], another classification of the interference types is made based on the distance between the APs, three classes are defined according to which nodes perceive interference and the throughput is analyzed for each case. One of the conclusions of this work is the fact that when two APs see each other there will be no possibility for parallel transmissions, greatly reducing the possible throughput, and RTS/CTS will offer worse results than basic access. A similar conclusion is found by the authors in [15], who also classify all the interference possibilities into three groups based on the interference that can be created depending on the type of node involved. They study how RTS/CTS will behave in these scenarios showing that it is not enough for solving these problems in multi-AP scenarios, as well as the effect of the distance between APs and its effect on the throughput, and how increasing the distance between APs increases overall performance.

The authors in [17] study the effect of path-loss in high density WLANs, showing that a network that is considered over-congested in an open environment can be non-congested when walls are present, arriving at the conclusion that unplanned AP deployments in moderate to high path-loss areas are far less critical than they are in open environments.

In [18], the authors concentrated on the current methods for dealing with channel interference, they study how Auto Channel Selection(ACS) does not use only orthogonal channels and uses sub-optimal channels that create inter-channel interference, as well as how Auto Rate Fallback (ARF) worsens the throughput for every node in the network in an attempt to fix one single node's performance. They propose a solution for channel assignment based on number of nodes in a WLAN. In [19] another dynamic channel assignment scheme is proposed, in which the APs broadcast their channel and the amount of nodes associated to them, so that each AP can decide which channel to use asynchronously. The authors in [20] propose a method to detect overlapping networks by using a differential of the RSSI at a given time and checking if it exceeds certain thresholds. They also propose calculating the duration of packets with the size and rate defined in the header and checking if the RSSI after that point is still higher than the noise level to detect interference. After the detection, they rely on the AP auto-selecting a different channel to avoid further interference.

The authors in [21] propose methods to optimally design a dense network, considering both 11g and 11a protocols, and centering their study on finding the right cell radius as well as optimal Clear Channel Assessment (CCA) tuning. They also find that the higher data rates are not desirable due to their high SINR requirements. The work in [22] also attempts to modify the CCA dynamically to improve throughput of the system by making it proportional to the number of contending STAs of each node. It also attempts to improve overall fairness by tuning the transmission opportunity (TXOP) according to a pre-set threshold defined by the network manager so that the links with lower access probability can send multiple consecutive frames for a short period of time.

The work in [23] extends the idea of Enhanced Distributed Channel Access (EDCA) by giving different access priorities to the STAs or even the APs. By setting different priority groups it proposes a system in which the values for the CWmin, CWmax, Arbitration InterFrame Space Number (AIFSN) and TXOP are changed during a period of time on each AP of a network, giving priority to one Basic Service Set so that it can work unhindered, improving the overall network performance over time. This system is based around the idea of every node seeing each other.

In [24], the authors propose a modification of the current Network Allocation Vector (NAV) in which the current vector will be called Self BSS NAV(SBNAV) and a new one will be created called Overlapping BSS NAV (OBNAV), this new vector will be set according to beacons and packets received from a different network, and the node will only transmit once both NAVs allow it.

2.3 Machine Learning in WLANs

While Machine Learning is not a new subject, its use as a way to improve wireless networks is fairly recent. There are several types of Machine Learning algorithms and they can be classified in two groups: Supervised Learning, in which the algorithm is trained with data that has been labeled following certain rules, and then it is supposed to find the function, pattern or model that best defines the entire data set, and Unsupervised Learning, in which the data fed to the algorithm is not labeled at all, meaning that the algorithm is supposed to sort the data by finding patterns in it. Supervised Learning has several subcategories, like semi-supervised learning, which uses a combination of labeled and unlabeled data, Active Learning, which allows the algorithm to ask questions to an information source, and Reinforcement Learning, which uses rewards and punishments as a way to help the algorithm predict the action with a better reward through trial and error.

In [25], the authors present a summary of the future of wireless networks and their use of Machine Learning centered on 5G. Their Reinforcement Learning section gives a good introduction on Q-Learning and its use in conjunction with the Markov decision process, as well as mentioning the Multi Armed bandits problem and its applications. The authors in [26] use Reinforcement Learning so that a Wireless Sensor Network may organize itself. The nodes treat sending a broadcast query in a particular channel as an action, and keep trying different channels to update the rewards for each one based on the neighbors discovered with the end goals to find the best possible path to the sink.

The work in [27] aims to improve the coexistence of LTE and Wi-Fi networks by using Q-learning and an ϵ-greedy algorithm to dynamically select the amount of time an LTE device should transmit in its duty cycle so that interference can be minimal for both networks. In [28], coexistence between LTE and Wi-Fi is also attempted by having the base station offload traffic from applications that can handle delays to the Wi-Fi bands while keeping the stricter applications in the licensed bands. They use Reinforcement Learning to find an optimal configuration of transmission across both Wi-Fi and cellular bands.

In [29], the authors use a feed-forward Neural Network (NN) to improve STAs' ability to pick the best AP at any time. A STA implementing such a NN must take into consideration several metrics for estimating the performance achieved by each AP. Once AP association is done, the model is validated by comparing the experienced performance with the predictions done. Such procedure is carried out until enough iterations have passed so as to guarantee convergence.

In [30], an AP association scheme is proposed in which the STAs keep estimating the utilization of the APs they use, and based on a threshold, switch to a new AP if they become unsatisfied. They use Reinforcement Learning to stop the AP switching to become too frequent.

The authors in [31] explore several Reinforcement Learning algorithms as a way to solve resource allocation in wireless networks. Actions chosen by the agents are the

configuration of both channel and transmission power; and they try to find the one that gives the higher throughput.

Chapter 3

User association in IEEE 802.11 WLANs

3.1 Introduction

In this Chapter we will explain the standard association procedure for IEEE 802.11, from its initial release where there was only a single method of encryption to the current amendment and future ones. We will see that several improvements try to reduce the association time so that switching to a different AP can be imperceptible to the end user, while others are designed with the intention to give information to the STA and improve the AP selection.

3.2 Basic procedure

IEEE 802.11 uses three different types of frames:

- **Management**: These frames carry information about the network, they are used by the APs and STAs to find each other, negotiate association, terminate it, or just ask for reports on the quality of the connection.
- **Control**: These frames are used once the connection has been established between APs and STAs, they are used for signaling between all the nodes in the network to better control the flow of transmissions (like RTS & CTS frames).
- **Data**: Frames containing data, meaning that they are used to encapsulate other protocols, like TCP or UDP.

Figure 3.1 shows an IEEE 802.11 frame. The frame control field on the header tells us the type of frame in its third and fourth bits. If these bits are '00' it is a management frame, control frames use '01' and data frames use '11'.

For the entirety of this Chapter, we will concentrate on Management frames, since they are the ones that take care of user association.

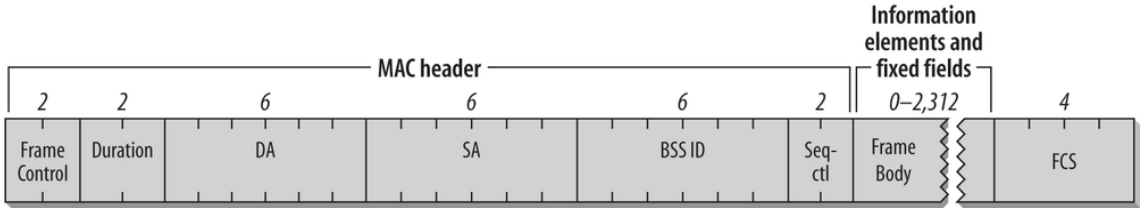
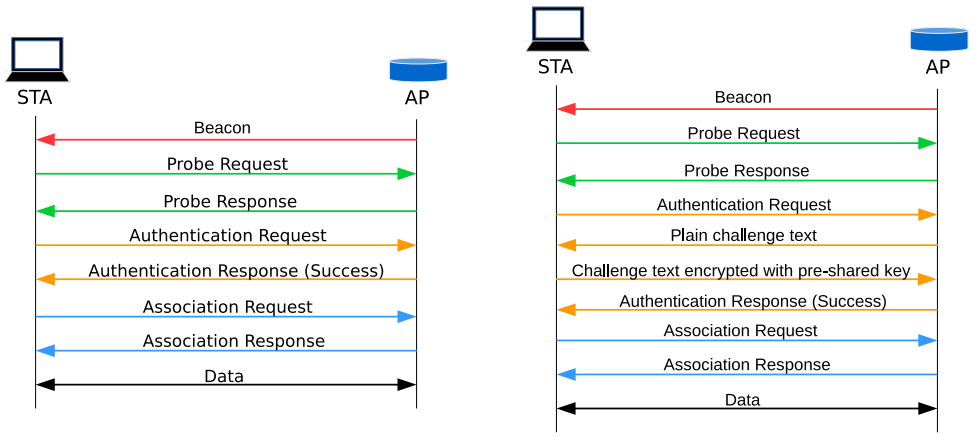


Figure 3.1: Standard 802.11 frame [1]

There are three possible states in the user association process, which is shown in Figure 3.2 :

- **Unauthenticated and unassociated**: The initial state in which the STA and AP exchange information about their data rates and encryption type with probes, and then the AP checks if the STA should be associated to the network with the authentication frames. In its initial release, the 802.11 protocol only allowed open system or Wired Equivalent Privacy (WEP) encryption. In an open system the authentication frames are sent even if they should not fail.
- **Authenticated and unassociated**: Once authenticated, the STA sends an association request with the information negotiated previously (rates, channels, etc). If the information matches the one from the AP, the STA receives an association response with a success code and is given access to the network. Otherwise, the association response is still sent but with a code that explains why the association failed.
- **Authenticated and associated**: State in which the STA and AP can exchange data frames.



(a) Open System (b) WEP encryption

Figure 3.2: Message flow for each association type

3.3 IEEE 802.11 amendments

There have been several improvements to the IEEE 802.11 standard over the years. These improvements are amendments that get added to the standard, and are denoted with a letter behind the standard's name. We will look at those that relate to user association in this section. Figure 3.3 shows a bit of the history of the standard and how the amendments get added over time. As we can see it can take a while before a new version of the standard is implemented, which means that usually a bunch of amendments get added at once. The current standard is the 2016 version, and there already are multiple amendments being drafted for the next version.

We will focus on amendments centered around AP-STA association. We have seen in the previous section that initially, association was a fast and simple process. However because of security needs, this process got more complicated with time. We will mention the steps taken from the initial amendment to the future 802.11ai, which aims to greatly shorten the association time.

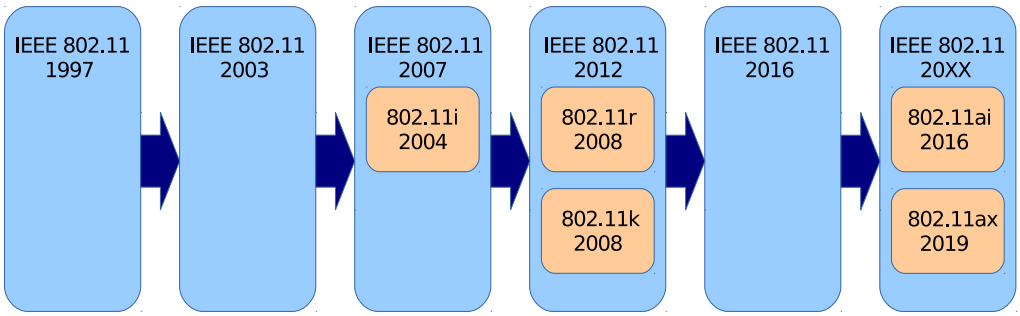


Figure 3.3: 802.11 time line

3.3.a IEEE 802.11i

We mentioned previously that WEP encryption was the only option in the original standard. This mechanism was created in 1997 and uses a stream cipher called RC4. This creates a semi random key and the key is combined with a plain text using Exclusive or (XOR).

WEP was quite vulnerable to an attacker that captured the challenge frames, and after a few years the key was easily decipherable in minutes. The first solution to this problem was Wi-Fi Protected Access (WPA), which was a patch that could be implemented with firmware updates to improve on the WEP mechanism before a new, better mechanism was devised. WPA improved the cipher construction and

used a new key for each packet, dealing with the main vulnerability of the WEP mechanism, its static key. It also included a Message Integrity Check (MIC).

Finally, Wi-Fi Protected Access 2 was implemented in the 802.11i amendment. It further improved security by using the Robust Secure Network from 801.x. This amendment allowed the use of WPA2 encryption based on the 4-way handshake, depicted in Figure 3.4a.

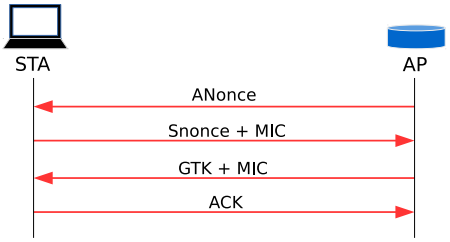
The 4 way handshake begins with the assumption that both STA and AP know the Master Key which can be either negotiated through another protocol (Extensible Authentication Protocol) or is previously known (like the password that is written below the router in residential deployments). The first message is sent by the AP with the Authenticator Nonce (Number used only Once) and their address. The STA responds with their own Nonce and the Message Integrity Check (MIC) so that the AP can authenticate the information and respond with the Group Temporal Key (GTK) that it will use for broadcast messages. Finally, an Acknowledgment (ACK) terminates the handshake and lets the AP know that the information was received by the STA.

The 4 way handshake changed how the association process worked. The STA and AP would send the same messages they would if the system was open, and after the association response the handshake would begin. With this, the amendment had a couple of consequences: the first one is the creation of networks that were more secure than ever before, the second one is that association became much more cumbersome, as it required two to four more messages before giving access to the network. This new association is shown in Figure 3.4b.

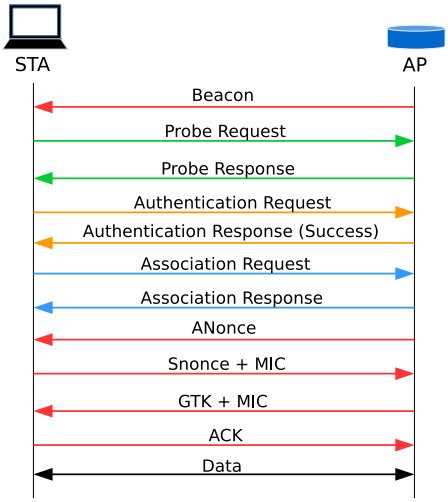
We mentioned previously that in some deployments the master key is negotiated through another protocol. The Extensible Authentication Protocol (EAP) was introduced the same year as the 802.11i amendment, and its purpose is the negotiation of the master key through the exchange of certificates that authenticate the STA and the AP. This method is used in most non-residential environments, like universities, and much like the 802.11i amendment, it brought security at the cost of a higher number of frames. The EAP association process is shown in Figure 3.4c.

After the initial Association Response, the EAP protocol starts. The EAPoL start is just a trigger for the communication to begin. The AP asks for the identity of the user so that it can be relayed to the server. If the user exists, the certificate exchange proceeds, and the master key is derived. At this point the AP just relays the messages between STA and server, changing the 802.11 header and encapsulating the frames into a RADIUS format for the server. Once the EAP authentication is validated, the four way handshake begins between the AP and STA.

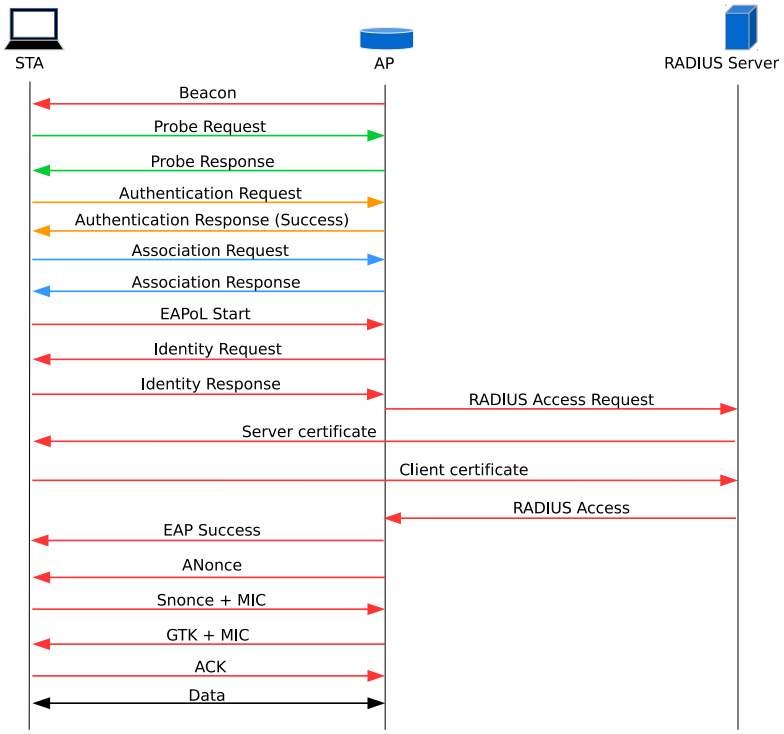
These new methods of association had a big impact on roaming, the 4 way handshake by itself introduced 4 more packets to the association process, but the biggest impact came from the addition of an authenticating server. Since the transmissions between AP and server are not restrained by a wireless signal, their proximity was unnecessary, which meant that transmissions between them could reach values of 200 ms or more, and this doubled the association time at the very least.



(a) 4-way handshake



(b) 802.11i association



(c) 802.11i association with EAP

Figure 3.4: 802.11i frame flows

3.3.b IEEE 802.11r

This amendment was designed to reduce the amount of frames needed to roam between APs, reducing the time needed to switch to another AP. The solution proposed was called Fast BSS transition, and it worked for all Basic Service Sets (BSS) in an Extended Service Set (ESS), meaning that it worked for all the APs in the same network. Fast BSS transition does three things: the first one is encapsulating the 4-way handshake in the authentication and re-association frames; the second is skipping the certificate exchange if it has been done once already; and the third one is allowing the current AP to relay the authentication frames to the new one, allowing the STA to remain connected instead of having to disassociate before sending them.

The same year that this amendment was published, a revision of EAP was proposed, called the Extensible Re-association Protocol (ERP). Much like 802.11r, it was designed for roaming, and tries to reduce the amount of frames needed for a STA that returns to a network. It is built on top of EAP, meaning that if a STA sends an ERP message and the AP does not use the protocol, after a certain time the STA will initiate a full EAP communication. When available, it halves the amount of frames needed. The ERP exchange can be seen in Figure 3.6.

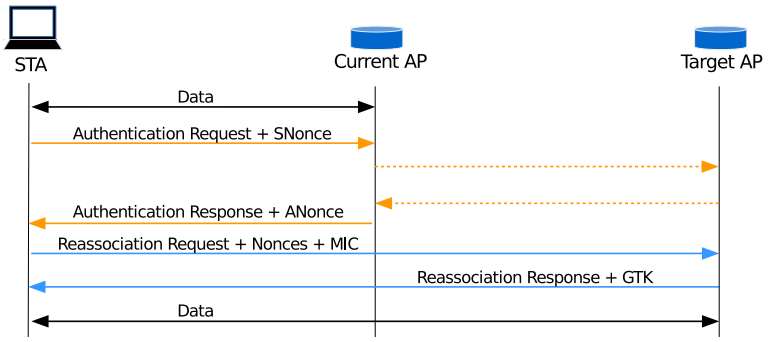


Figure 3.5: 802.11r re-association

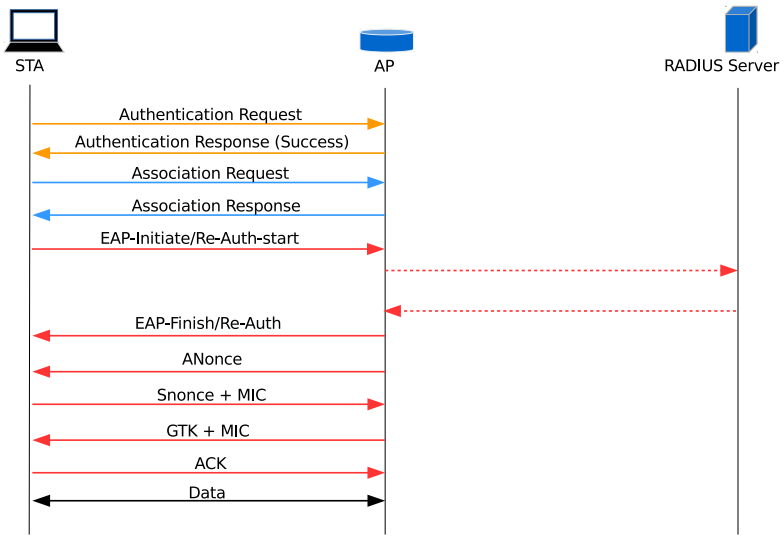


Figure 3.6: 802.11i re-association with ERP

3.3.c 802.11ai

The 802.11ai amendment aims to bring together all these concepts by encapsulating ERP messages in the authentication frames, and the 4-way handshake in the association frames. The process is shown in Figure 3.7. The authentication frames encapsulate the ERP frames as well as the Nonces of the 4-way handshake, and then the remaining two messages of the 4-way handshake are encapsulated into the association responses, which brings the number of frames needed for a re-association to the same number needed in an open system.

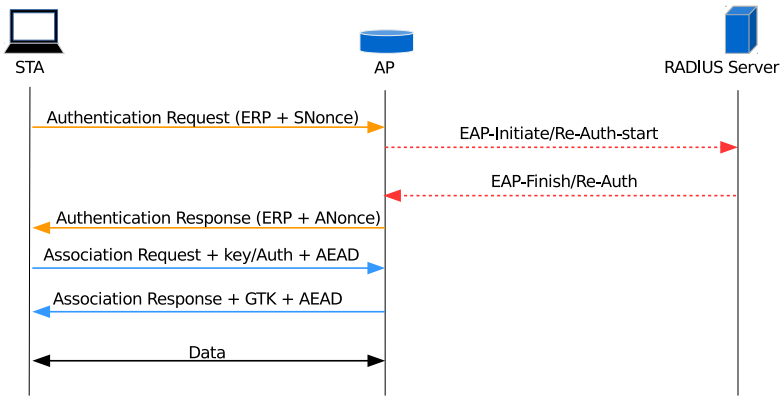


Figure 3.7: 802.11ai re-association with FILS

3.3.d 802.11k

The 802.11k is an amendment that is usually associated with 802.11r, not only because they were published the same year, but because they complement each other. The 802.11r amendment allowed for faster re-association, but a STA does not send probe requests once it is associated, and unless other APs happened to be on the same channel as their current one, they would not receive their beacons. That is what 802.11k fixes with the neighbour report, which is an action frame that a STA sends to its current AP to acquire information about nearby APs. This report is a list of all the APs that the current AP can sense, with all the information that would usually be found in a beacon, such as security type and channel used. This report is designed to allow the STAs to skip the disassociation and scanning that roaming would usually require.

3.3.e 802.11ax

This amendment is not focused on association, but on improving the performance of the network in terms of data rates and overheads. To do this it uses Orthogonal Frequency-Division Multiple Access (OFDMA), which will replace the current Orthogonal Frequency-Division Multiplexing (OFDM). OFDM encodes information by splitting it into several carrier frequencies that travel at low data rates, and once received the information is put together again. Its advantage over a single carrier at high speed is the use of guard intervals that allow it to evade Inter Symbol Interference (ISI). OFDMA improves upon it with a multi user version in which a subset of carrier frequencies is assigned to one user, and another subset to another user. Further, OFDMA increases the number of subcarriers available, and also brings the 1024-QAM modulation, which will allow us to reach higher rates than the previous maximum of 256-QAM.

As we mentioned, this amendment has no bearing on user association, but we will be using its features when simulating networks in future chapters.

3.4 AP selection in IEEE 802.11

In this Section we want to take a deeper look at the current AP selection method that STAs use: the Strongest Signal First (SSF) mechanism.

The first step in this mechanism is the scanning of channels. STAs scan all available channels looking for nearby APs, and they can do so in two ways:

- **Passive Scanning:** A STA starts by setting its antenna to a specific channel and waits a certain amount of time to receive beacons from APs in that channel. The amount of time spent in the channel depends on the implementation, but it should take into account that most APs send beacons every 100 ms by default. If the STA receives a beacon, it stores its information, and once the

STA has waited long enough, it switches to a new channel. This is repeated until the STA has tried every channel in its channel list, which can vary with the implementation, and could contain every channel or only a few.

- **Active Scanning:** The STA scans all the channels in its list much like the passive scanning, but instead of waiting for beacons, it sends probe requests, which will be answered by probe responses of nearby APs with the same information that would be in their beacons. This type of scanning should ideally take less time than the passive scanning because the waiting time before switching channels is based on the time that it would take for an AP to answer its request, instead of the beacon frequency. This wait time is also implementation dependent, but usually is based around the amount of time that the AP would need to answer the request if it were not overloaded.

While active scanning seems the better option of the two because of its lower scanning time, it has the disadvantage of adding overhead to the network, which can be a problem if there are a lot of STAs around. Most systems nowadays still use active scanning as a default in spite of this.

After the scanning is completed, the STA looks at the Basic Service Set Identifiers (BSSID) from the beacons or probes received to try and find one that it recognizes. If there are multiple BSSIDs that are known to the STA, it selects the one with the strongest Received Signal Strength Indicator (RSSI) and starts the association process as we described in Section 3.2.

3.5 Conclusions

From what we have seen, the majority of the association amendments proposed for the 802.11 protocol have been dedicated to improving security or making the exchange of frames more efficient. This is helpful for our purposes, considering that our aim is to allow a STA to find its most optimal AP, and that will require re-association to multiple APs to gather information, which requires short association times. In order to obtain information to decide which AP to use for re-association, 802.11k seems the most interesting proposal, as it gives the STAs the information required without interrupting the service for the end user. A conjunction of 802.11k and 802.11ai would allow the STA to find the optimal AP with a minimal amount of re-associations.

Chapter 4

Performance Evaluation of High Density WLANs

4.1 Introduction

In this Chapter we will create a model that accurately portrays a High-Density WLAN to study the more common problems that can be found when there are multiple APs in the same area. We will study the effect of channel interference, the effects of increasing the amount of nodes in a network and how we can use power saving and CCA adjustment to mitigate these issues.

Our aim in this Chapter is to find how much of a problem the hidden and exposed nodes are, to see if we already have the tools at our disposal to lessen their effect on the network throughput or if new mechanisms need to be devised to discover and control them.

4.2 System model

In this Section we will describe the parameters, assumptions and metrics we used to create our simulation of HD-WLANs and evaluate their performance.

4.2.a Scenario

Our simulation creates a 2D plane and sets a number of APs and STAs in random positions distributed uniformly. After that, each STA gets assigned to an AP following the Strongest Signal First (SSF) mechanism. The signal received by each node is calculated using the log distance path-loss model, which we show in Equation 4.1. The parameters used in the equation are described in Tables 4.1 and 4.2.

Parameter	Meaning
RP(dBm)	Received Power
TP (dBm)	Transmission Power
PL (dB)	Path Loss
BPL	Break-Point Loss
PL_0(dB)	Path Loss at 1 meter
Gs(dB)	Shadowing interference
fc (GHz)	Frequency band used
d_walls (m)	Distance between walls

Table 4.1: Glossary

Parameter	Value
Area(m)	50x50
PL_0(dB)	40.05
Gs(dB)	Normally distributed with std. dev. 5
Channels	8
fc (GHz)	5
d_walls(m)	10

Table 4.2: Simulation parameters

Log distance path-loss model according to the enterprise simulation scenario by the 802.11ax task force [32]:

$$\begin{aligned} \text{RP} = \text{TP-PL} = \text{TP-PL}_0 + 20\log_{10}\left(\frac{\text{fc}}{2.4}\right) + 20\log_{10}(\min(\text{dist}, 10)) + \\ + \text{BPL} \cdot 35\log_{10}\left(\frac{\text{dist}}{10}\right) + 7 \cdot \left(\frac{\text{dist}}{d\_\text{walls}}\right) + \text{Gs} \end{aligned} \tag{4.1}$$

We calculate the received power by subtracting the path-loss from the transmission power of the node, which will be 20 dBm unless specified. PL_0 is the loss in dBm after the signal travels one meter. The fc parameter refers to the band used, it can either be 2.4 Ghz or 5 GHz. The next two elements in the equation give the loss based on the distance between STA and AP. First we check the losses when the distance is less than 10, and the next element checks for those cases in which the distance is equal or higher than 10 with the break-point loss (BPL). If BPL is 1 then we add further loss to the equation.

$$\text{BPL} = \begin{cases} 1, & \text{if dist} \geq 10 \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

The penultimate element gives us the loss due to the traversed walls in the signal's path. The $d_$walls is the separation between walls in our simulation: 10 metres. Finally, the Gs is the shadowing, or interference due to objects affecting the propagation of the signal, each pair of AP-STA obtains the shadowing using random normally distributed values with a standard deviation of 5.

4.2.b Metrics used

To evaluate the performance of an HD-WLAN we will use several metrics, and the most important ones are the successful and unsuccessful transmissions. We want our network to have a higher number of successful transmissions than unsuccessful ones, and we will use the following metrics to analyze the interference that causes transmissions to fail or succeed. These metrics are mainly a classification of the types of nodes in the network.

We consider three types of nodes:

1. **Hidden nodes:** If two nodes A and B are in the same channel and cannot sense each other (they receive less than -82 dBms from the other one's signal), and the difference in their signal strength at A's receiver is lower than a certain threshold, B is a hidden node of A. This relationship can be reciprocal, but it does not need to be.

2. **Contending nodes:** If two nodes sense each other they are contending nodes, and like before, this can be a reciprocal relationship but it does not need to be.

3. **Exposed nodes:** If two nodes A and B are in different networks that share the same channel and can sense each other, if the difference between the power perceived by their transmissions at A's receiver is higher than a certain threshold, B is an exposed node of A. Once again, this does not need to be reciprocal. It bears mentioning that exposed nodes are a subset of contending nodes.

Let us explain this threshold that we have mentioned: when a node receives a signal, this signal needs to be above a certain SNR ratio to be decoded. The ratio depends on the hardware and the data rate used, but when two signals are received, one needs to have a higher transmit power than the other for the node to be able to filter the weak one out.

The example in figure 4.1 shows the detection of a hidden node: STA 2 receives -91 dBms from STA 1. Since STA 2's CCA is -82 dBm, STA 1 is ignored, but if the two STAs transmit at the same time, the AP receives a signal of -48 and -59 dBms from STA 1 and STA 2 respectively. STA 1 has a strong enough signal that the AP

can decode it even if they transmit simultaneously, but STA 2 does not, therefore, STA 1 is a hidden node of STA 2.

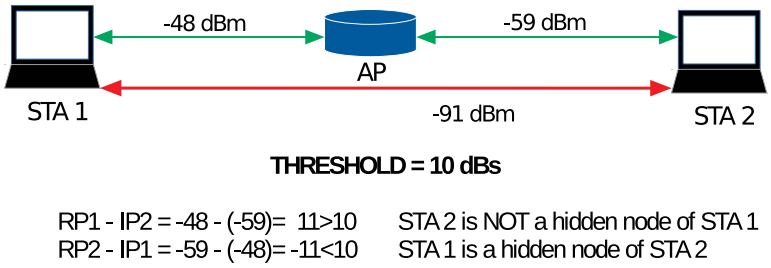


Figure 4.1: Example of a hidden node when using a threshold of 10 dBs

With this we can now explain how the types of nodes have an effect on the amount of successful transmissions we can find:

If STA 1 is a hidden node of STA 2, that means that STA 1 and STA 2 do not sense each other, and that sometimes they will try to transmit at the same time, which will create a collision: an unsuccessful transmission.

If STA 1 is an exposed node of STA 2, STA 2 will hear STA 1 transmitting and hold its transmission to not create a collision in a situation when both STAs could transmit at the same time without issues. These nodes do not create an unsuccessful transmission but stop a successful one from happening.

The contending nodes are just part of the CSMA mechanism, two contending nodes respect each others transmissions and evade collisions. This works as intended on a single AP network, but when there are multiple APs it leads to exposed nodes.

An ideal network should have no hidden or exposed nodes, and every node should be a contending node for every other node.

4.2.c Simulation approach

We will now explain how the transmissions are simulated[1]: First, at the beginning of the simulation we define which node will be the receiver for each node. In the case of the STAs they all transmit to their respective APs, and the APs choose one random STA associated to them as their receiver. Now that every node has a receiver we can check which transmissions will create problems with each other. Following what we showed in Figure 4.1 we now check which nodes are hidden, exposed or contending nodes for every STA and AP in the network and store this information in a table.

Finally we simulate transmission slots in a simplification of the CSMA mechanism. For every slot, every node in the network tries to transmit. We assign a random

[1]The code for the simulation can be found in: https://github.com/MCarrascosaZ/Improving-user-association-in-HD-WLANs-using-Machine-Learning

number to every node in the network and then check them one by one. The first one we check gets to transmit by default, the second one gets to transmit as long as the first node is not its contender, the third one transmits if the first and second one are not its contenders, and so on.

Once we go through all the nodes we have obtained a list of nodes that will try to transmit in the slot, and now we check if they are hidden nodes of another transmitting node. If a transmitting node has a hidden node in this list, then its transmission is considered unsuccessful, if it does not, then the transmission will be successful.

Figure 4.2 shows an example of the simulation. In the first slot STA 4 is the only STA transmitting, this could be because every other STA has it as a contender. The second slot shows that two STAs can transmit at the same time as long as they are not contenders or hidden nodes of each other. Slot 3 shows a collision, which is created by STA 2 and STA 3 being hidden nodes of each other. Slot 4 shows a case in which there are two concurrent transmissions and an unsuccessful one, in this case STA 1 could be a hidden node of STA 3, and if STA 1 has a strong enough signal, the AP can still decode its transmission. Finally, slot 5 shows that APs can also transmit. We consider a slot as a unit of time, and each transmission lasts a slot.

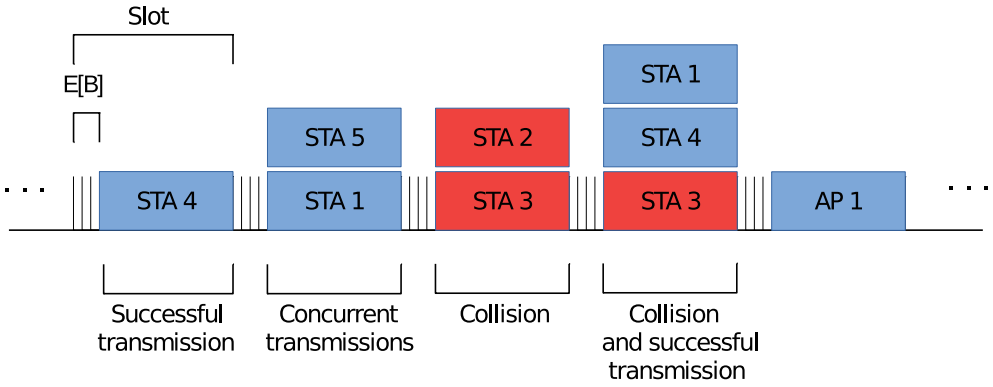


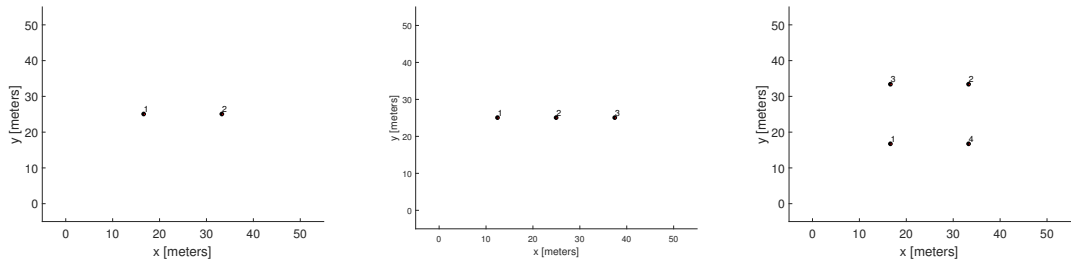
Figure 4.2: Transmission examples. Successful transmissions in blue and unsuccessful ones in red.

4.3 Results

Every simulation uses random values for the placement of the STAs. For the APs we will use mostly fixed placements that will be specified in each section. Each node has a contention window of 16, a CCA of -82 dBm and a transmission power of 20 dBm. Every simulation is repeated 100 times with a different seed each time.

4.3.a Multiple Overlapping WLANs

In this first simulation we show the effects of having multiple networks in range of each other and on the same channel. We will use three configurations in which the location of 10 STAs will be randomized, but the position of the APs will be fixed. In the first configuration we have 2 APs side by side, in the second one we have 3 APs side by side, and on the third one we set 4 APs in a square formation. We show these configurations in Figure 4.3

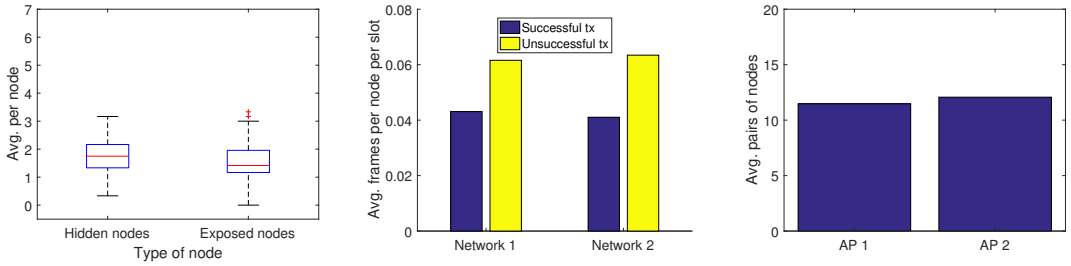


(a) 2 APs in a line (b) 3 APs in a line (c) 4 APs in a grid

Figure 4.3: AP configurations

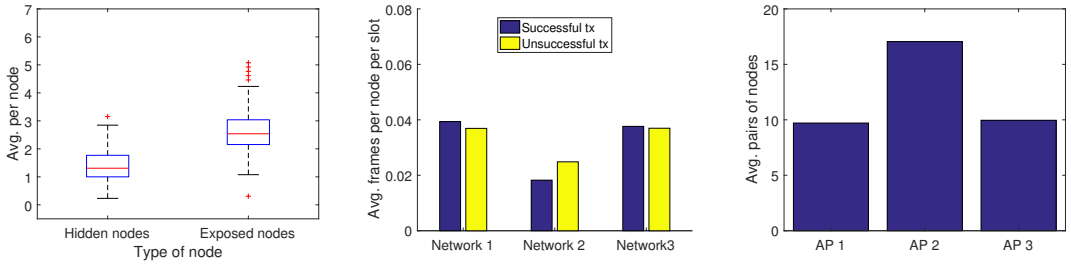
Figure 4.4a shows the average amount of hidden and exposed nodes for each node in the network. We can compare with figure 4.5a to see that as the amount of hidden nodes is a bit smaller when using 3 APs, and even smaller when using 4. This is due to the increasing distances between APs. In the case of the 3 APs, even if the middle one is close to the other two, the first and third AP are at a distance of 25 metres, and this allows some of their nodes to not create interference.

Figures 4.4c,4.5c and 4.6c show how many pairs of nodes each AP can sense in which the nodes of that pair cannot sense each other (i.e: AP 1 senses STA 1 and 2, but STA 1 does not sense STA 2). This serves as an approximation of how much interference each AP will receive. We can see that in the cases of 2 and 4 APs, since they are balanced scenarios, the amount of pairs is similar for each AP. However, in the configuration that uses 3 APs we can see the imbalance created in the middle AP, which sees 7 pairs more than the other APs. The consequence of this can be seen in Figure 4.5b, where the middle network has half the amount of transmissions as the other APs, and is the only one with more unsuccessful than successful transmissions.



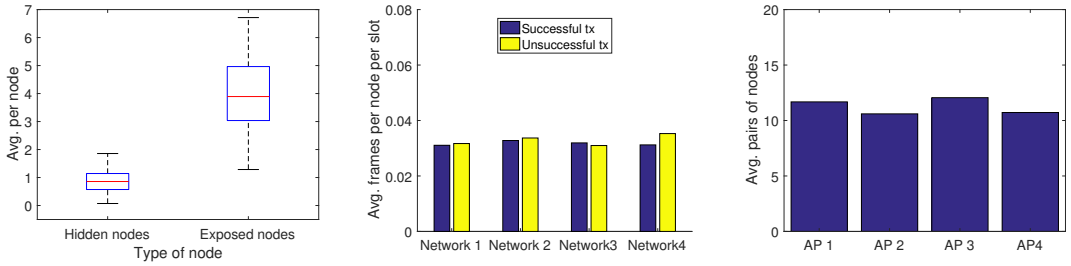
(a) Hidden/exposed nodes (b) Transmissions (c) Pairs sensed by the APs

Figure 4.4: 2 APs share the channel



(a) Hidden/exposed nodes (b) Transmissions (c) Pairs sensed by the APs

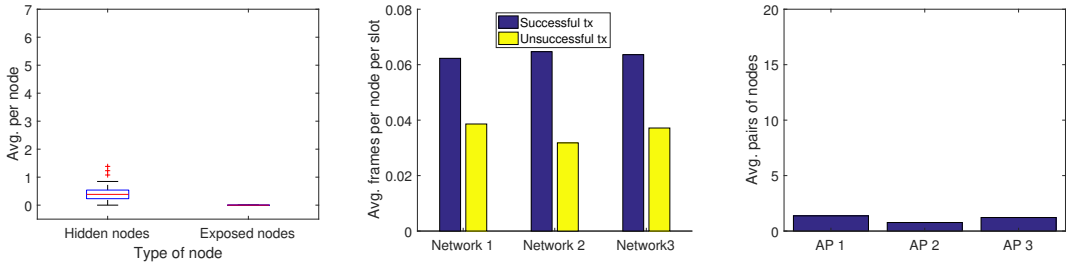
Figure 4.5: 3 APs share the channel



(a) Hidden/exposed nodes (b) Transmissions (c) Pairs sensed by the APs

Figure 4.6: 4 APs share the channel

Let us consider the 3 AP configuration. We will now use a different channel for each AP and rerun the simulation. The results can be seen in 4.7. The first thing to realize is that now there are no exposed nodes. Further, the amount of hidden nodes has decreased from 1.5 to 0.4 because now the only hidden nodes possible are from nodes within the same network instead of every node in the entire area. We can also see that the middle AP has the same amount of transmissions as the others, since it now does not receive any interference. Also, the pairs of nodes sensed by each AP has significantly decreased, the highest value in Figure 4.5c was 17 and in Figure 4.7c we can see that the biggest value is more than ten times lower.



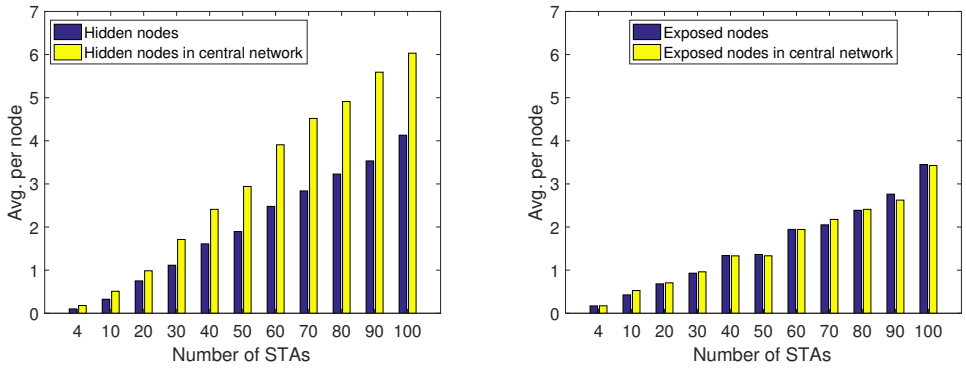
(a) Hidden/exposed nodes (b) Transmissions (c) Pairs sensed by the APs

Figure 4.7: 3 APs with different channels

4.3.b Increasing nodes in the network

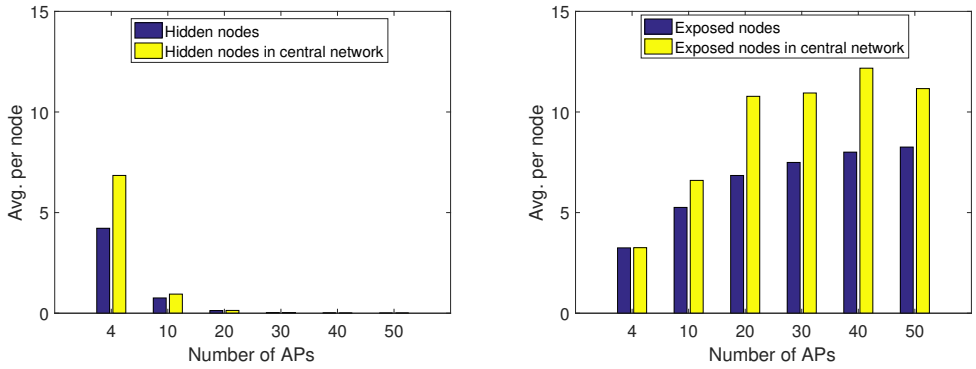
In this section we will check the effects that adding more nodes will have on the network. We will use two configurations: one in which we fix the amount of APs to 4 and increase the amount of STAs from 4 to 100, and another in which we fix the number of STAs to 100 and increase the amount of APs from 4 to 50. In both cases the first AP will be located at the center of the area, and every other node will have a random location. The APs pick one of eight channels at random. We run 100 iterations every time.

We start by checking the number of hidden and exposed nodes. In Figures 4.8 and 4.9 we show the average amount of hidden and exposed nodes for each configuration and we compare them to the central network, which is in the busiest location of the area. We can see in Figures 4.8a and 4.8b that as the amount of STAs increase, so do both the hidden and the exposed nodes. This behaviour is replicated in Figure 4.9b, where we see the exposed nodes also increase when we increase the number of APs. Figure 4.9a however, shows that by adding more APs we can greatly reduce the amount of hidden nodes in our network. The hidden and exposed nodes perceived by the central network serve once again to show that the position of the nodes is an important part of avoiding interference.



(a) Hidden nodes with 4 APs and increasing STAs (b) Exposed nodes with 4 APs and increasing STAs

Figure 4.8: Increasing the amount of STAs in the network

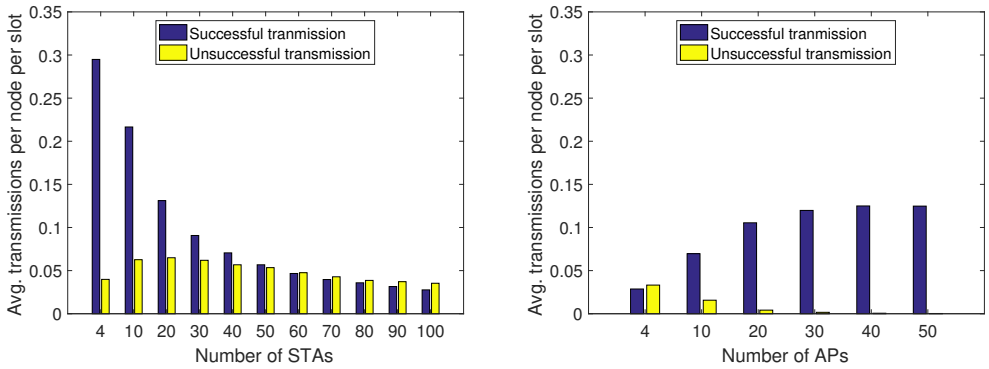


(a) Hidden nodes with 100 STAs and increasing APs

(b) Exposed nodes with 100 STAs and increasing APs

Figure 4.9: Increasing the amount of APs in the network

Next, in Figure 4.10 we show how the average number of successful and unsuccessful transmissions for each node varies when increasing the number of nodes. In Figure 4.10a we see that with few STAs the amount of successful transmissions is far higher than the unsuccessful ones, but this reverses around 60 STAs, when each AP has an average of more than 10 STAs associated to it. In Figure 4.10b however, we see that increasing the amount of APs increases the successful transmissions while decreasing the unsuccessful ones. This is mainly due to the fact that the unsuccessful transmissions are caused by hidden nodes. The more hidden nodes there are, the more nodes that will transmit at the same time creating a collision. Since exposed nodes are virtually the same as contenders, they stop transmissions from happening, but do not create collisions. This means that increasing the number of APs will have a positive effect on the network.



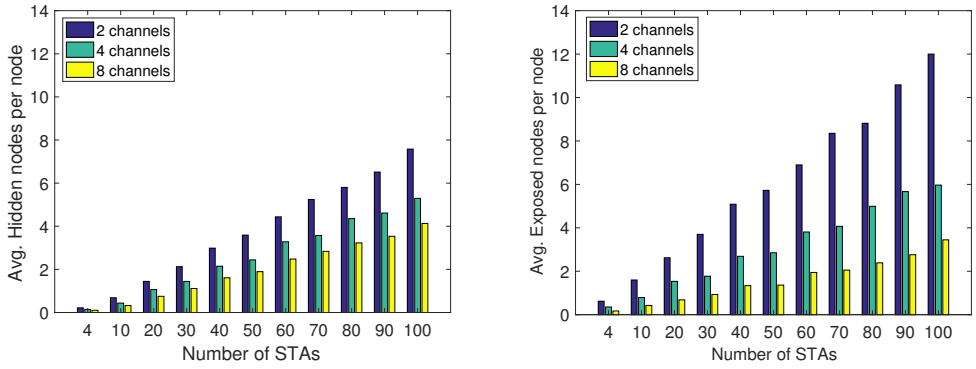
(a) Transmissions with 4 APs and increasing STAs

(b) Transmissions with 100 STAs and increasing APs

Figure 4.10: Successful vs. unsuccessful transmissions

Next we will use the same scenarios but with a varying amount of channels available to the APs. To begin, we can see how the number of hidden and exposed nodes

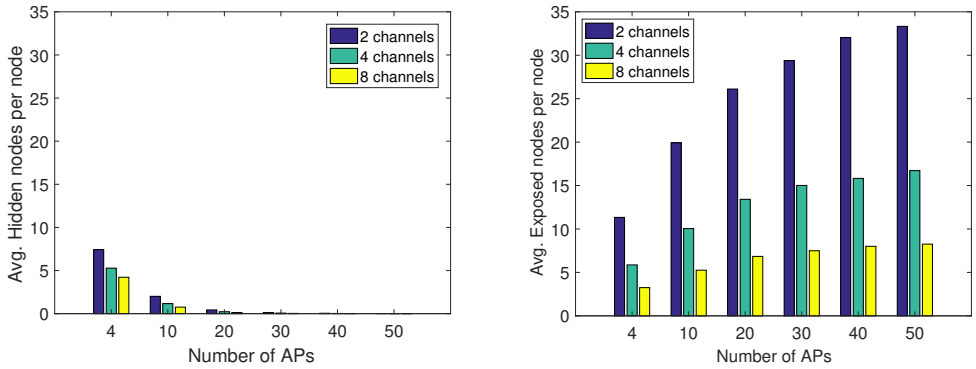
react in Figure 4.11 and 4.12. We see that predictably, both hidden and exposed nodes decrease once we have more channels available, and that this is the same no matter what type of node we add to the network.



(a) Hidden nodes with 4 APs and increasing STAs

(b) Exposed nodes with 4 APs and increasing STAs

Figure 4.11: Hidden and exposed nodes with increasing number of channels



(a) Hidden nodes with 100 STAs and increasing APs

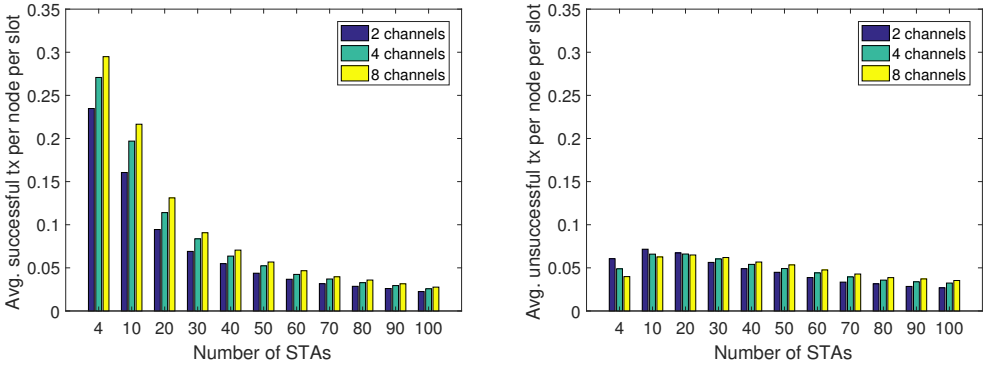
(b) Exposed nodes with 100 STAs and increasing APs

Figure 4.12: Hidden and exposed nodes with increasing number of channels

When we look at how the successful transmissions change when we use more channels we find the expected result in Figures 4.13a and 4.14a, where we see that the more channels we have, the more successful transmissions we will get. In the case of the unsuccessful transmissions however, we see that more channels can actually have them increase instead of decrease. If we look at Figure 4.13b we see that up until there are 20 STAs, there are more unsuccessful transmissions with less channels, and after that, there are more. In the case of Figure 4.14b, with 4 APs less channels work better, but as the number of APs increases the difference between using 2,4 or 8 channels starts to become smaller.

When increasing STAs, the increase in unsuccessful transmissions is due to the higher amount of contending nodes. When using 2 channels the amount of exposed

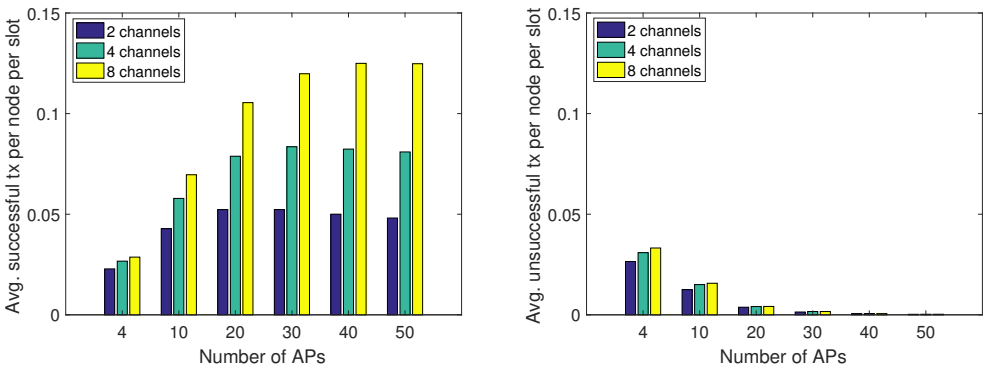
nodes and contender nodes on the same channel is higher than when using 8. This means that every single node has more nodes that stop it from transmitting. When using 8 channels there are less contenders on the same channel, which means the nodes get to transmit more often, creating collisions.



(a) Successful transmissions (b) Unsuccessful transmissions

Figure 4.13: Transmissions with 4 APs and increasing number of STAs and channels

For the case in which we increase the amount of APs, we always have the same number of STAs (100). We start by using 4 APs, and this means that each node has an average of 25 contenders or possible hidden nodes with 8 channels, but an average of 50 when using 2 channels. If we look back to Figures 4.12a and 4.12b we can see that the amount of exposed nodes is far higher than the amount of hidden nodes. What happens in this case is that when we use less channels, the high amount of exposed nodes prevents most nodes from transmitting, and when they get to transmit, the amount of hidden nodes is small enough that they cause few collisions. However, when we use more channels the amount of hidden nodes slightly decreases while the number of exposed nodes does so heavily, meaning that nodes get more chances to transmit with the same chance of a collision when they do so.



(a) Successful transmissions (b) Unsuccessful transmissions

Figure 4.14: Transmissions with 100 STAs and increasing number of APs and channels

4.3.c Capture effect

The capture effect refers to the fact that when a node receives two signals at the same time, it can only decode one or none of them, depending on their difference in signal strength. The stronger signal is decoded, and the weakest is treated as noise.

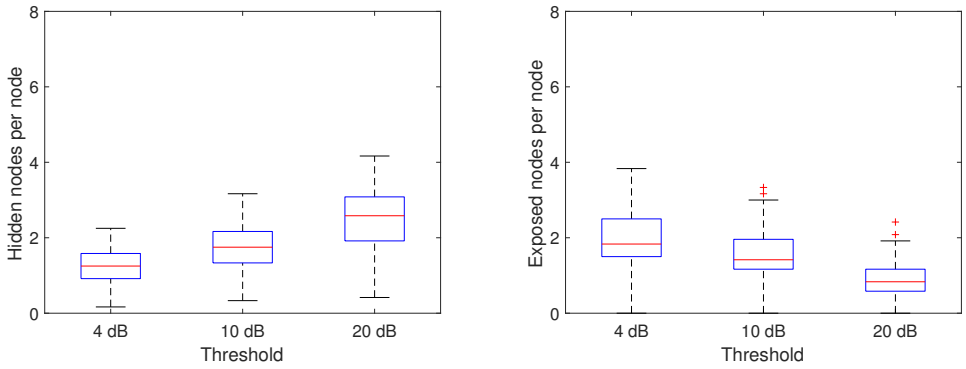
In our simulation we have considered the following: two nodes that do not sense each other are not hidden nodes as long as the receiver can decode one of the signals. When there are two transmissions at the same time because of two nodes not sensing each other, the receiver may be able to decode it depending on their SNR. If it can decode it, we do not consider them hidden nodes because the slot, at least for two out of the three nodes in this conversation, has proceeded without an issue. For the exposed nodes we consider something similar: if the difference in the SNR of two signals from nodes on different networks is big enough to not create a collision if the nodes transmitted, then we consider them exposed nodes, otherwise, they should be called contenders since they would create a collision if they transmitted.

The SNR needed to decode a signal depends on two things: the data rate and the hardware used. The higher the data rate used the higher the SNR needed will be, this is studied in [33], where they propose to use lower data rates to improve performance thanks to lowering this SNR threshold, avoiding packet loss. The SNR needed also depends on the hardware used. In [34], the authors show that two nodes with the same antenna model can have a different threshold, in their case the same they find a difference of almost 2 dB. The authors in the previously mentioned [33] also discuss a specific case in which the strongest signal comes after the first weaker signal, and in that case they found a minimum threshold of 10 dBs which they theorize could be programmed into the hardware to protect weaker signals up to that threshold.

We will begin with the three configurations we used previously, using 10 STAs and 2, 3 and 4 APs on the same channel. Figures 4.15, 4.17, and 4.19 show the average hidden and exposed nodes for each node in the network, as well as the average successful and unsuccessful transmissions averaged across 100 iterations.

The first thing to see from these plots is that the overall tendencies we saw before are still present, the middle AP in the 3 AP configuration is going to be more saturated than the others no matter what our SNR threshold is. What changes is the amount of hidden nodes and exposed nodes we perceive. When increasing the SNR threshold we make it so that signals are harder to decode because nodes that are relatively close to each other will usually have signals of similar strength, this means that the higher the threshold, the higher the distance (or the channel interference) between the contending nodes needs to be. We can see this in Figures 4.15a, 4.17a and 4.19a, where the amount of hidden nodes increases steadily with the threshold. Figures 4.15b, 4.17b and 4.19b however, show that the amount of exposed nodes has decreased, meaning that the nodes can sense their contenders more accurately.

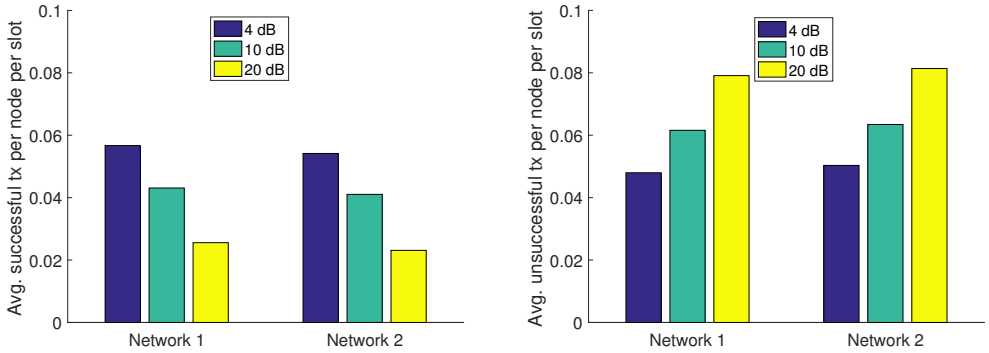
Finally, Figures 4.16a, 4.18a and 4.20a show that a lower threshold will benefit the network by having more successful transmissions and less unsuccessful ones.



(a) Hidden nodes when the threshold increases

(b) Exposed nodes when the threshold increases

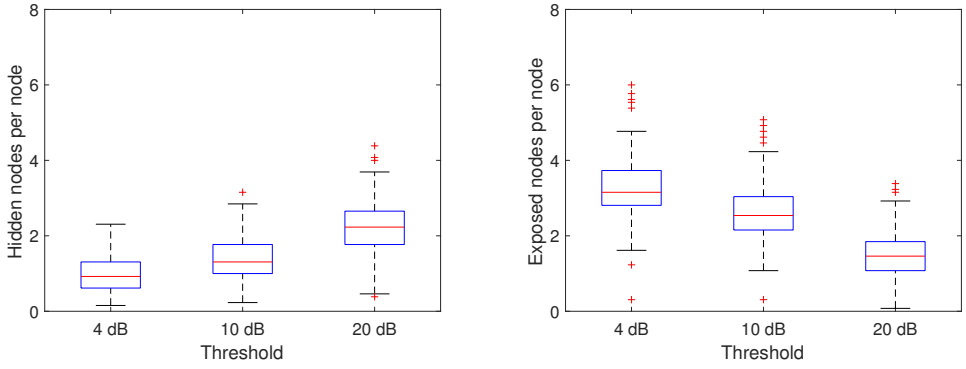
Figure 4.15: Hidden and exposed nodes with 2 APs and 10 STAs



(a) Successful transmissions of each network

(b) Unsuccessful transmissions of each network

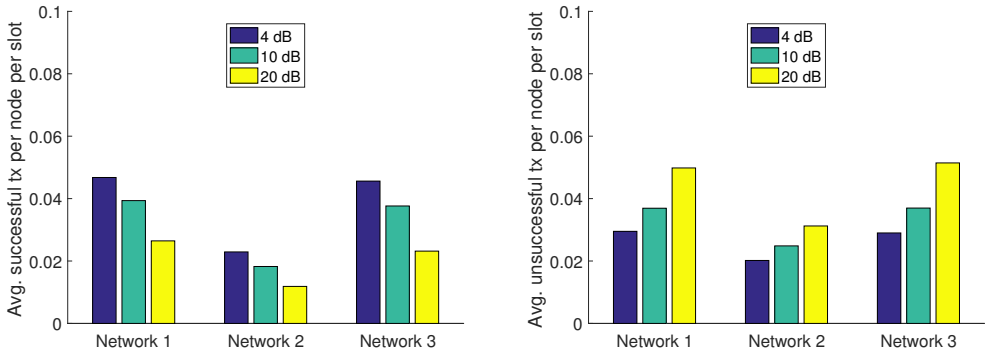
Figure 4.16: Transmissions with 2 APs and 10 STAs



(a) Hidden nodes when the threshold increases

(b) Exposed nodes when the threshold increases

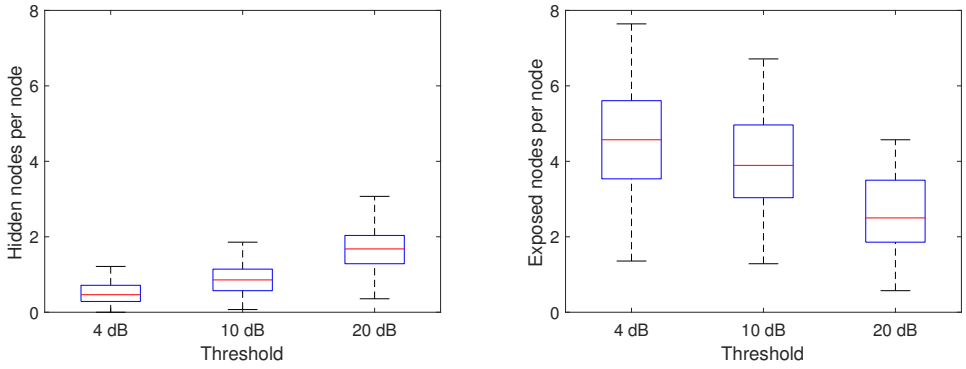
Figure 4.17: Hidden and exposed nodes with 3 APs and 10 STAs



(a) Successful transmissions of each network

(b) Unsuccessful transmissions of each network

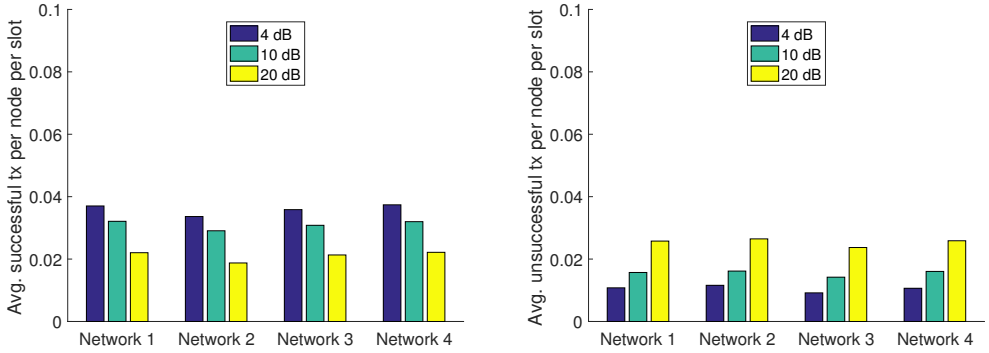
Figure 4.18: Transmissions with 3 APs and 10 STAs



(a) Hidden nodes when the threshold increases

(b) Exposed nodes when the threshold increases

Figure 4.19: Hidden and exposed nodes with 4 APs and 10 STAs

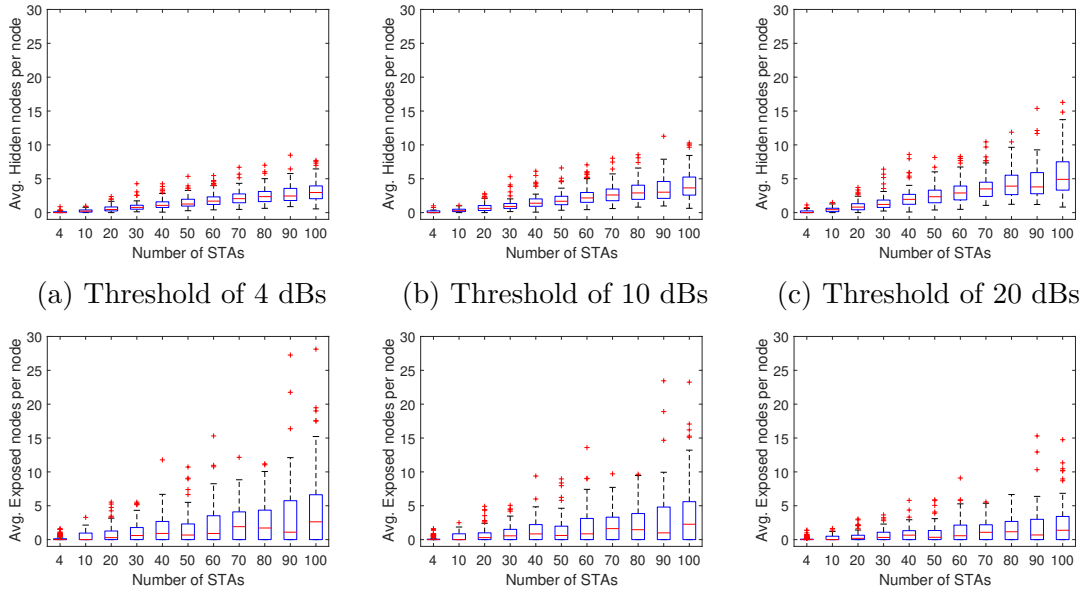


(a) Successful transmissions of each network

(b) Unsuccessful transmissions of each network

Figure 4.20: Transmissions with 4 APs and 10 STAs

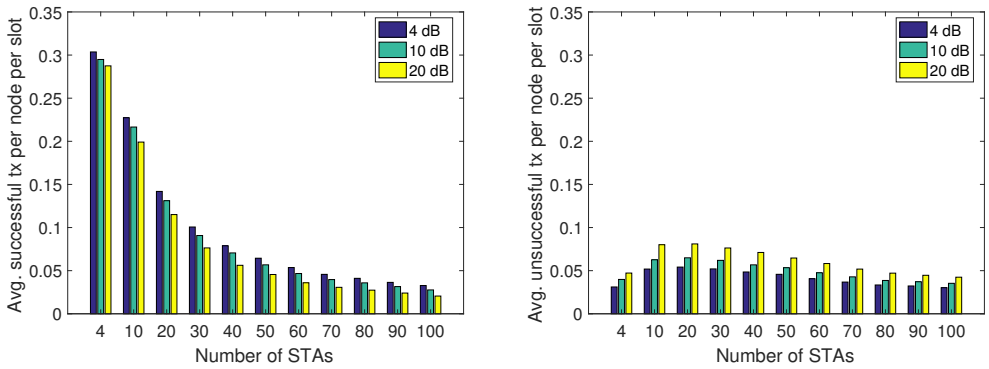
Now we will check if the results we saw in these first configurations hold when we increase the number of nodes in the network. Figure 4.21 shows the average hidden and exposed nodes per node as the number of STAs increases. Previously we have seen that increasing the number of STAs increases the amount of hidden and exposed nodes, and increasing the threshold increases the amount of hidden nodes and decreases the exposed nodes. When we do both, we see that hidden nodes do increase, but exposed nodes decrease slightly, at 100 STAs with 4 dBs we get 2.6 exposed nodes, while with 20 dBs we get 1.3. In this case, it seems that the change in the SNR threshold has bigger effects than the amount of STAs in the network.



(a) Threshold of 4 dBs (b) Threshold of 10 dBs (c) Threshold of 20 dBs

(d) Threshold of 4 dBs (e) Threshold of 10 dBs (f) Threshold of 20 dBs

Figure 4.21: Boxplots of the average hidden and exposed nodes with 4 APs and increasing STAs



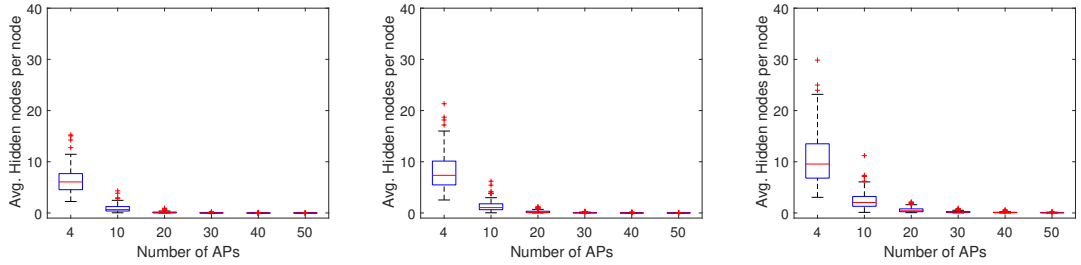
(a) Successful transmissions (b) Unsuccessful transmissions

Figure 4.22: Transmissions with 4 APs when the threshold and STAs increase

Figure 4.22 shows how many transmissions every node gets on average when we

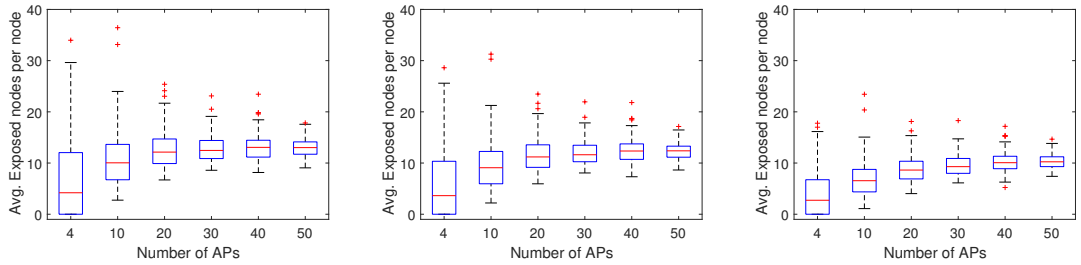
add STAs and change the threshold. The number of successful transmissions per node decreases the more nodes there are, they also decrease when the threshold is higher, and the reverse happens with the unsuccessful transmissions, we get more of them with a higher threshold. In this case, the difference between unsuccessful transmissions between thresholds seems to get smaller the more STAs we add.

Next, we will increase the number of APs. In Section 4.3.b we saw that as we add APs we get less hidden nodes and more exposed nodes. In this section we saw that when we increase the threshold we see more hidden nodes and less exposed ones. Figure 4.23 shows the hidden nodes for each threshold. While the effect of adding APs is still strong, we can see more hidden nodes with a smaller threshold. When we use 4 dBs and 10 APs, the median is 0.67 hidden nodes, and with 20 dBs we have 2.05. Which means that increasing the number of APs can mitigate the effect of the threshold, depending on how big the capture effect is, the bigger amount of APs we will need however.



(a) Threshold of 4 dBs (b) Threshold of 10 dBs (c) Threshold of 20 dBs

Figure 4.23: Boxplots of the average hidden nodes with 100 STAs and increasing APs



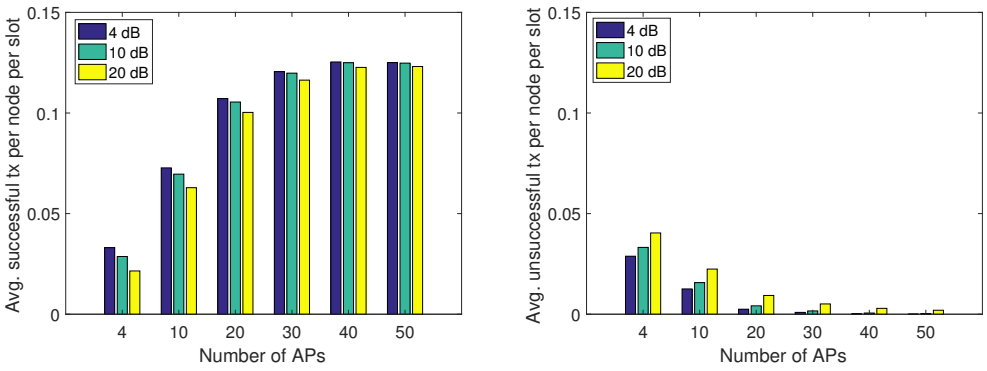
(a) Threshold of 4 dBs (b) Threshold of 10 dBs (c) Threshold of 20 dBs

Figure 4.24: Boxplots of the average exposed nodes with 100 STAs and increasing APs

In Figure 4.24 we see a similar tendency, adding APs increases the amount of exposed nodes, yet the higher the threshold the smaller the amount of exposed nodes we can find when we compare the three figures.

Finally, we check how the transmissions are affected by the threshold as we add APs. In Figure 4.25a we see that a smaller threshold gives us more successful

transmissions, yet this effect is diminished as we increase the number of APs, to the point that at 50 APs the values are almost the same. Figure 4.25b works as expected, with a higher threshold always leading to more unsuccessful transmissions.



(a) Successful transmissions (b) Unsuccessful transmissions

Figure 4.25: Transmissions with 100 STAs when the threshold and APs increase

4.3.d CCA and power adjustment

CCA adjustment

In this section we change the CCA of the STAs to see if it is a viable solution to increase the number of transmissions of our nodes. All our STAs start with a CCA of−82 dBm. This is the minimum RSSI required to maintain a connection between an AP and a STA.

Once all STAs are associated, we change the CCA so that the STAs use the minimum CCA that allows them to receive frames from the AP consistently. We do this following Equation 4.3, in which $\mathrm{CS}_{\mathrm{th\_max}}$ is a maximum threshold we choose as a stopping point so that our STA cannot end up with a CCA that is too high to be practical, in this case −42 dBm. $E_{\mathrm{AP\_RSSI}}$ is the mean received power from the AP beacons. Finally, the margin is another value we set to ensure that we do not let our CCA be too high. If the value chosen is lower than −82 dBm, we leave the CCA as the default.

$$\mathrm{CCA} = \begin{cases} -82, & \text{if } (E_{\mathrm{AP\_RSSI}} + \text{margin}) \leq -82 \\ \min(\mathrm{CS}_{\mathrm{th\_max}}, (E_{\mathrm{AP\_RSSI}} + \text{margin})), & \text{otherwise} \end{cases} \tag{4.3}$$

Simply put, if a STA receives −65 dBm from an AP, the default CCA of −82 dBm is too generous and will allow transmissions from other STAs that are beyond the AP to reach us, meaning that our number of contenders will be higher than necessary. By increasing the CCA we will avoid these situations.

Power adjustment

This works similarly to the CCA adjustment. In order to reduce the amount of nodes that receive the signal from a STA, we want to reduce its signal strength. The idea is to send the weakest signal possible that allows the STA and AP to remain connected. This way the nodes that are farther than the AP will not sense the STA, reducing their contender list. Since we do not modify the CCA of the APs, we can expect all the signals to end up close to -82 dBm.

Once the STAs are associated, we give them 6 options, 20 dBm, 10 dBm, 5 dBm, 0 dBm, -5 dBm and -10 dBm, the STAs go sequentially through these options, and as soon as the association is lost, they revert to the previous one.

CCA and power adjustment simultaneously

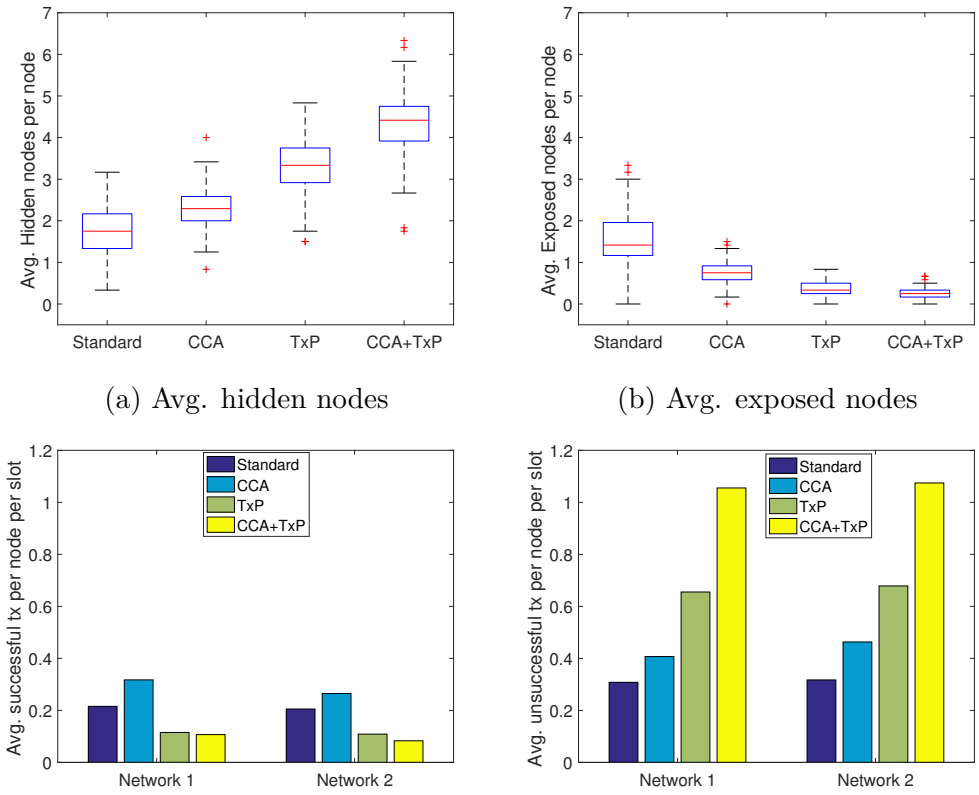
Both our CCA and transmission power adjustment are performed on the STAs. The APs will always use 20 dBm of transmission power and a CCA of -82 dBm. This means that both methods can be applied in any order without disrupting the other. When we use CCA control the STAs will lower their CCA according to the signal sent by the AP at 20 dBm, and when we apply power control the STAs adjust themselves to the -82 dBm CCA of the APs.

Results

Like in the previous section, we will start with the 2, 3 and 4 AP configurations we previously used, with 10 STAs and everyone on the same channel. Figure 4.26a shows the average hidden nodes for each node in the network when we change the CCA, the transmission power and both at the same time. We compare it to the original case in which everything is left untouched.

When we change the CCA we always increase it from the initial -82 dBm, it makes sense then that we get more hidden nodes since we are ignoring nodes that are farther from the STA or receive strong interference. The payoff to this can be seen in Figure 4.26b, where we see that our exposed nodes decrease more than our hidden nodes increase. Changing the transmission power seems to have the same effect as lowering the CCA, and combining both methods we get the same tendencies, however, when we use both CCA and power control, we get an increase of more than 1 hidden node when compared to only using power control, yet the exposed nodes decrease very slightly. Using both methods, at least at first glance, does not seem optimal.

Figure 4.26c and 4.26d show the average successful and unsuccessful transmissions per node, separated by network. Here we can see that the CCA adjustment seems to be the only method worth using in this case, since it gives us an increase in successful transmissions, with a similar increase in the unsuccessful ones. Every other method decreases the successful transmissions and highly increases the unsuccessful ones. We can also see here that using power control alone is far better than using both power and CCA control, since unsuccessful transmissions spike from less than 0.7 with only power control to more than 1 with both used, while leaving the successful ones at a similar amount.



(a) Avg. hidden nodes

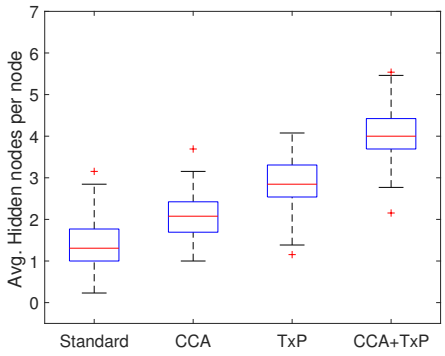
(b) Avg. exposed nodes

(c) Avg. successful transmissions by network

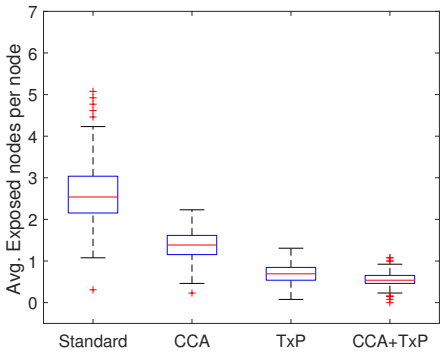
(d) Avg. unsuccessful transmissions by network

Figure 4.26: Results for 2 APs and 10 STAs

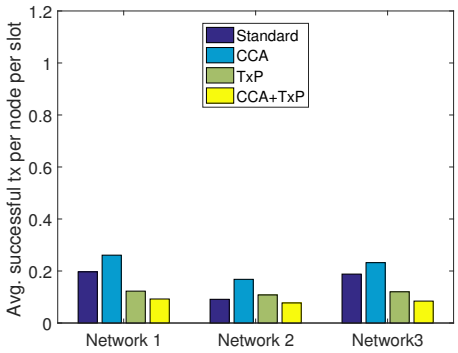
Figure 4.27 shows very similar results to the previous image. CCA control may be worth using but power control or power and CCA control at the same time are completely detrimental to the network. We could gather from this that perhaps avoiding exposed nodes is not as beneficial as avoiding hidden nodes.



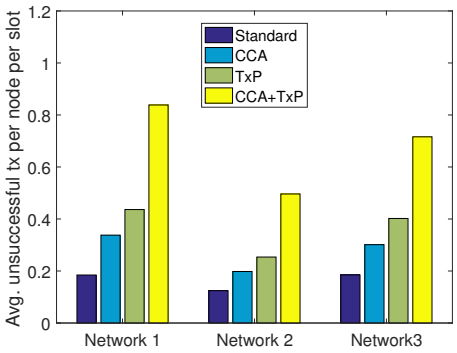
(a) Avg. hidden nodes



(b) Avg. exposed nodes

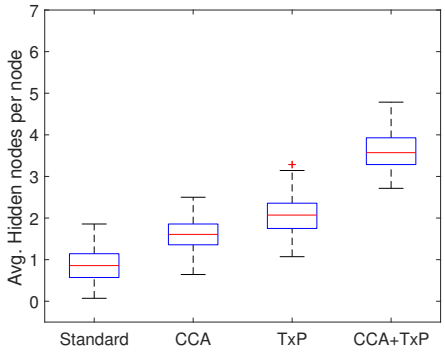


(c) Avg. successful transmissions by network

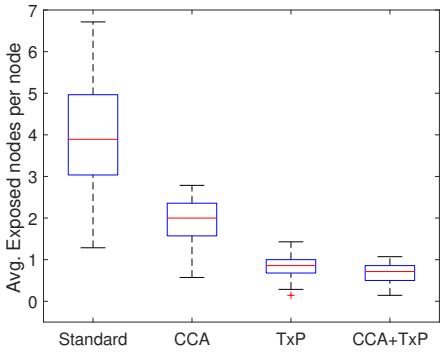


(d) Avg. unsuccessful transmissions by network

Figure 4.27: Results for 3 APs and 10 STAs



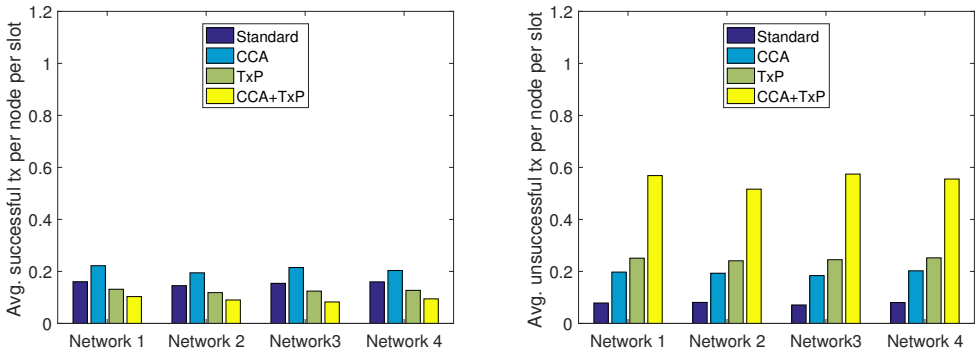
(a) Avg. hidden nodes



(b) Avg. exposed nodes

Figure 4.28: Hidden and exposed nodes for 4 APs and 10 STAs

Figure 4.28 shows the same problem, in this case we can see that the hidden nodes increase even more than before when using both methods of control. We can spot the same tendencies we saw before, CCA increases both successful and unsuccessful transmissions and every other method decreases successful ones and increases unsuccessful transmissions.

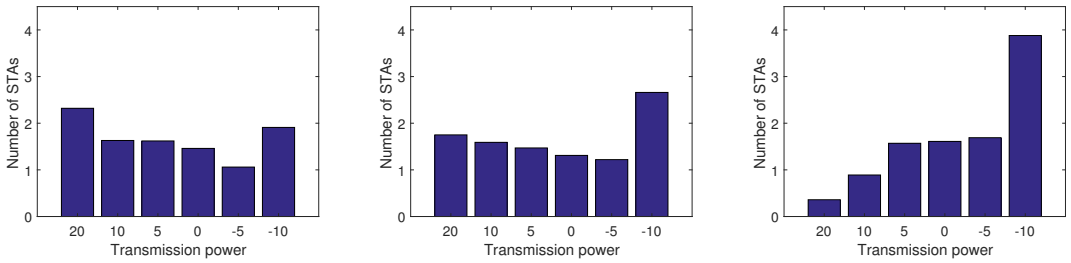


(a) Avg. successful transmissions by network

(b) Avg. unsuccessful transmissions by network

Figure 4.29: Transmissions for 4 APs and 10 STAs

In Figure 4.30 we show how many STAs on average pick each power option we give them. Keep in mind that the transmission power selection is independent to the CCA control, meaning that the STAs will choose the same option when we use only power control or both CCA and power control. We can see that the more APs in the network we have, the higher the amount of STAs that pick a lower transmission power. When we use 4 APs almost no STAs remain at 20 dBm.



(a) 2 APs

(b) 3 APs

(c) 4 APs

Figure 4.30: Transmission power for each configuration

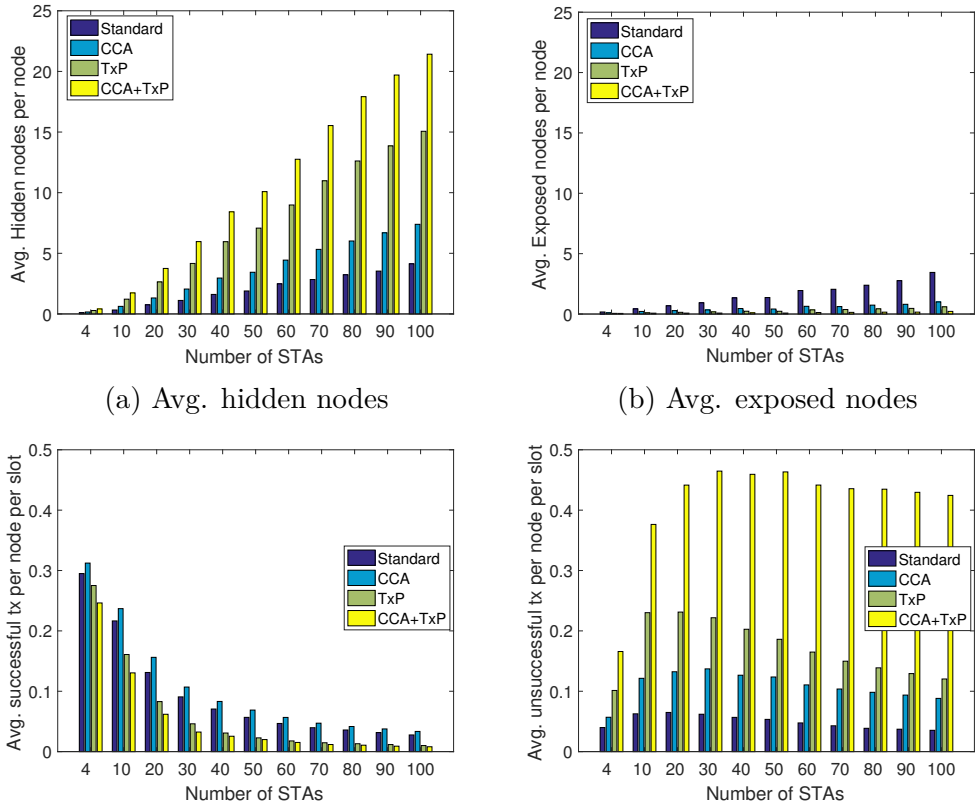
Next we use the same configurations we used in Section 4.3.b, we increase the amount of STAs and APs and we apply the control methods to see if they benefit from having more of one type of node.

First, we increase the number of STAs. In figure 4.31a and 4.31b we show the average hidden and exposed nodes for any node, and we can see that the hidden nodes increase both when we add STAs and when we use any method of control. If we look at the first and last steps we can see that with the standard configuration we would increase from almost 0 to around 4 hidden nodes, when we use CCA control they increase to almost double, this could be worth our while since the exposed nodes at the last step decrease from 3.5 on the standard configuration to 1 while lowering the CCA. We get double the hidden nodes but three times less exposed ones. If we use power control however, the amount of hidden nodes gained is more

than five times the standard's, while the decrease in exposed nodes is not that much better than the one offered by the CCA control.

Figures 4.31c and 4.31d give a similar impression. CCA control increases our successful and unsuccessful transmissions in a similar proportion. Power control in almost all cases gives us more unsuccessful transmissions than successful ones. And using both at the same time doubles the power control's effects.

In most cases, if not all, it seems that the best idea would be to leave everything to the default.



(a) Avg. hidden nodes

(b) Avg. exposed nodes

(c) Avg. successful transmissions

(d) Avg. unsuccessful transmissions

Figure 4.31: Results with 4 APs when we increase the amount of STAs in the network

Figure 4.32 shows how the STAs pick their transmission power depending on how many there are. It seems that half the STAs choose to stay at base transmission power or the lowest possible, with the other half choosing one of the intermediate options. Of this first half, the majority choose the lowest possible power. Since this method of power control always tries to use the lowest option available, what we have is that every STA uses an option that ends up sending really weak signals to the APs, always around -82 dBm, since it is the CCA of the APs. Knowing this, it makes sense that we get a lot of unsuccessful transmissions, since most signals will be very close in power, meaning that unless our hardware has a really small threshold for the capture effect, most transmissions will fail.

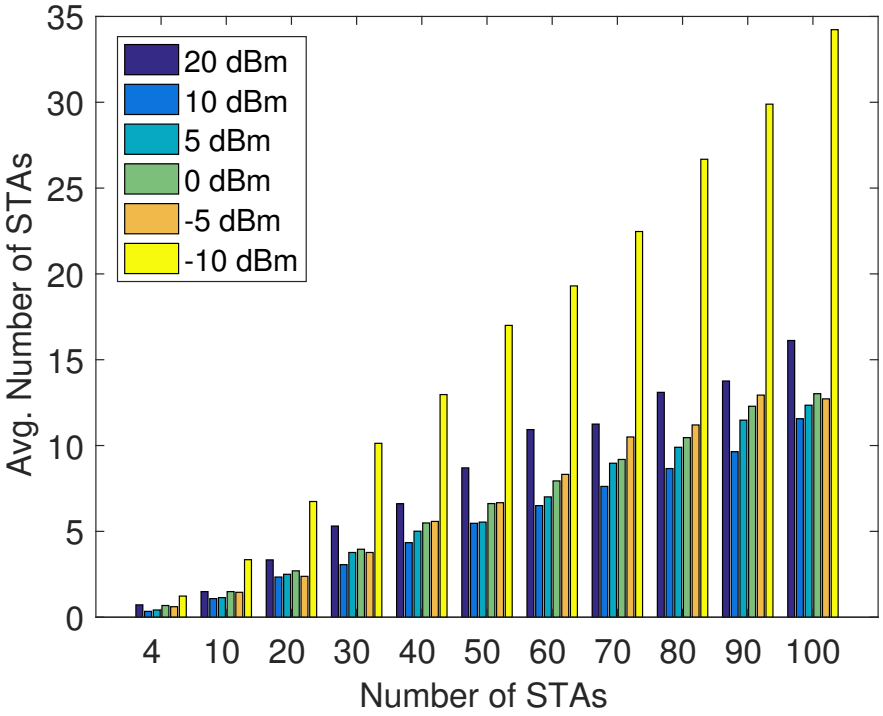
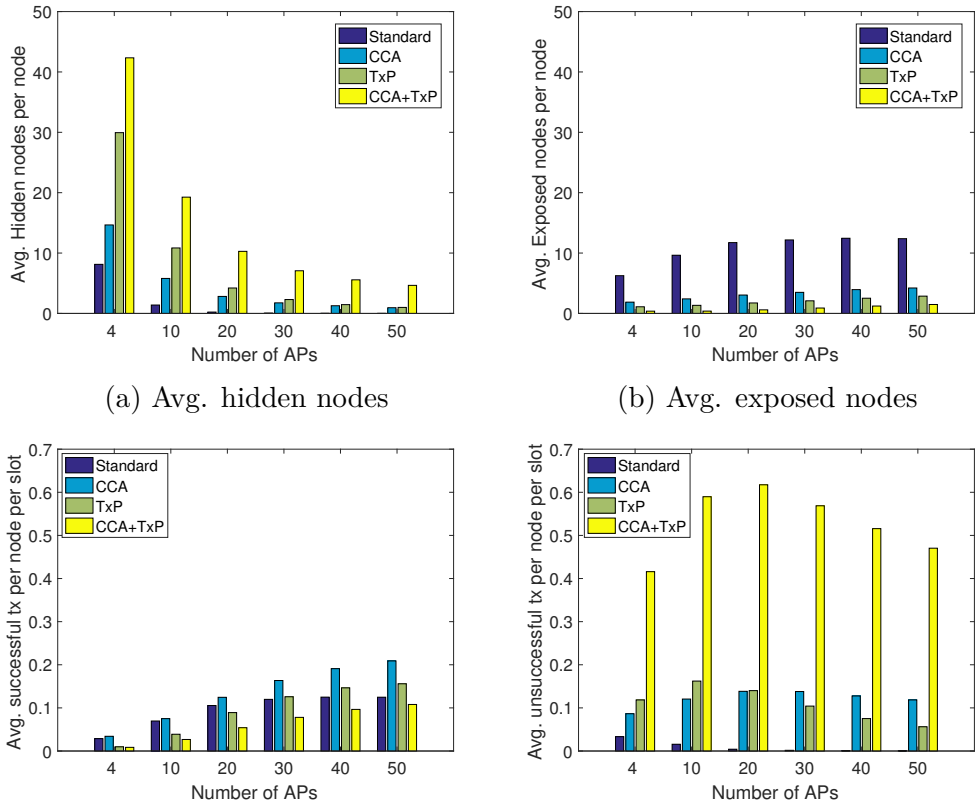


Figure 4.32: Transmission power selected when STAs increase

Next we will see what happens when we increase the number of APs, Figures 4.33a and 4.33b show an interesting result: even if using our methods of control gives us more hidden nodes, by adding APs we can nullify this increase to the point where it could be worth using these control methods. If we look at the last step we see that using the standard configuration we get almost no hidden nodes, yet when using CCA or power control alone we keep this value almost as small as before. The amount of exposed nodes however, changes dramatically, the standard configuration ends up with 12 nodes, while CCA, power control and both at the same time get 4, 3 and 2 respectively.

Figures 4.33c and 4.33d are not so promising, as they show that we still get plenty of unsuccessful transmissions. CCA control could be a good option, as it seems to give better results the more APs we add, but its number of unsuccessful transmissions

never seems to decrease much, while the standard configuration ends close to 0 by the time we reach 20 APs. As always, both CCA and power control is the worst by a big margin, with 5 times more unsuccessful transmissions than successful ones.



(a) Avg. hidden nodes

(b) Avg. exposed nodes

(c) Avg. successful transmissions

(d) Avg. unsuccessful transmissions

Figure 4.33: Results with 100 STAs when we increase the amount of APs in the network

Finally, we show the average CCA and power chosen by the STAs. Figure 4.34 shows the average CCA picked by the STAs as we increase the amount of APs. The more APs there are, the closer they are physically to the STAs, which means that they can lower their CCA. We haven't shown the CCA progression for any other case because this is the only one where there was any variation, when increasing the number of STAs, the average was around -67 dBm no matter how many STAs there were. For the initial cases of 2, 3 and 4 APs, the average were -70 dBm, -69 dBm and -65 dBm respectively.

Figure 4.35 shows the transmission power chosen by the STAs. As expected, the more APs there are, the lower the signal needed to reach one. By the time there were 20 APs almost no STAs remained at full power, on the last step, the average for -10 dBm is of 98.41 STAs.

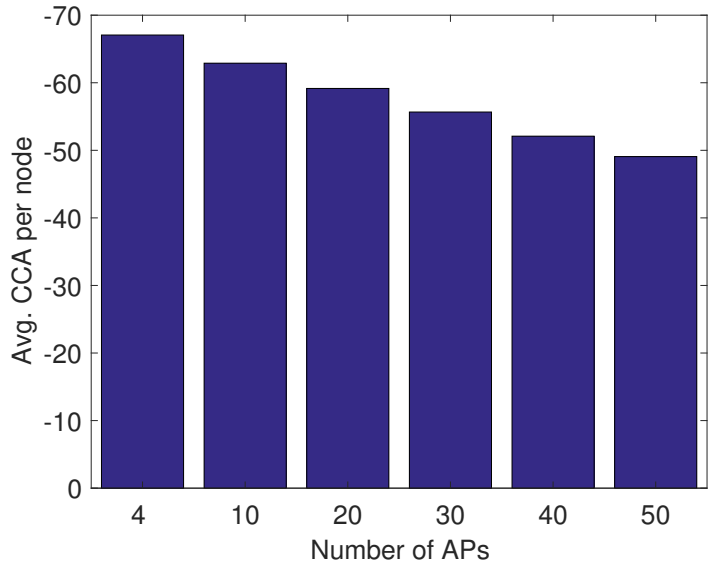


Figure 4.34: Average STA CCA with 100 STAs and increasing APs

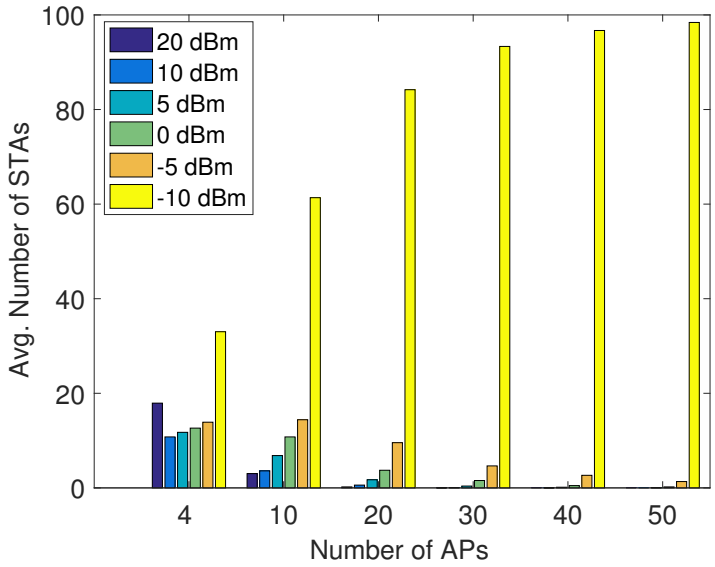


Figure 4.35: Transmission power used with 100 STAs and increasing APs

4.4 Conclusions

We have shown the effect of the hidden and exposed nodes problem in HD-WLANs, that poor channel allocation will greatly reduce the amount of transmissions in our network, how a high density of APs can be useful in evading hidden nodes while it can give us an increased number of exposed nodes. We have also shown that the hardware we use can have a big impact on the network through the capture effect, and finally we have seen that adjusting the CCA of the STAs can be a viable way of mitigating some of these problems, while power control seems detrimental in every way.

With all this, we believe that when designing an association mechanism, the avoidance of hidden and exposed nodes should not be a priority, as there are already several ways available to reduce their impact on the network. Dedicated mechanisms like the RTS/CTS further reduce this impact.

We have adjusted the CCA and TP of the STAs in accordance to the signal received from the AP. In the future, it would be interesting to adapt the CCA and TP of the APs themselves, and see if it can be beneficial to the network.

Chapter 5

Decentralized AP selection using RL

5.1 Introduction

In this Chapter we will find a way for the STAs in a network to associate to the best possible AP out of those they have in reach. We will use Reinforcement Learning, specifically a decentralized ϵ-greedy algorithm to let the STAs re-associate a number of times, hoping that by the last association they have found a better option than if they had used the Strongest Signal First method.

5.2 System Model

Here we will show how our simulation is created: we will start by describing the scenario we consider, then we will show the way we calculate the throughput of the nodes and we will end with an explanation of the way the simulation works.

5.2.a Scenario

For the scenario we reuse a few things from the previous Chapter. We still use a 2D plane of 50 by 50 metres with the APs and STAs placed randomly with a uniform distribution, and we still use the same path-loss model to measure the signal received by each node. We show it again in Equation 5.1.

Parameter	Meaning
RP(dBm)	Received Power
TP (dBm)	Transmission Power
PL (dB)	Path Loss
BPL	Break-Point Loss
PL_0(dB)	Path Loss at 1 meter
Gs(dB)	Shadowing interference
fc (GHz)	Frequency band used
d_walls (m)	Distance between walls

Table 5.1: Glossary

Parameter	Value
Area(m)	50x50
PL_0(dB)	40.05
Gs(dB)	Normally distributed with std. dev. 5
Channels	8
fc (GHz)	5
d_walls(m)	10

Table 5.2: Simulation parameters

Log distance path-loss model:

$$\begin{aligned} \text{RP} = \text{TP-PL} = \text{TP-PL}_0 + 20\log_{10}\left(\frac{\text{fc}}{2.4}\right) + 20\log_{10}(\min(\text{dist}, 10)) + \\ + \text{BPL} \cdot 35\log_{10}\left(\frac{\text{dist}}{10}\right) + 7 \cdot \left(\frac{\text{dist}}{d\_\text{walls}}\right) + \text{Gs} \end{aligned} \tag{5.1}$$

$$\text{BPL} = \begin{cases} 1, & \text{if dist} \geq 10 \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

5.2.b Network throughput model

While we reuse much of the scenario, the main difference in this Chapter is that we will not consider hidden and exposed nodes. In the previous Chapter we analyzed

their impact to the network and showed that there are several ways of avoiding them, such as proper channel allocation, increasing the density of APs or applying CCA control. For this Chapter we will consider their effect to be negligible. Further, we mentioned that a solution would be increasing the AP density to minimize the effect of hidden nodes, and this will work well with the algorithms we will use for association in this Chapter, as increasing the number of APs gives the STAs more options from which to choose, and Machine Learning will help us make the best possible choice.

We consider the occupation of the medium (the air) to measure the capacity of an AP to transmit to every STA associated. Let us consider a situation with one AP and two STAs: if we want the AP to transmit to both STAs in one second, we will use half a second for each transmission. Of course, depending on how much information each STA needs they may end up unsatisfied. If we consider their needs then maybe we can transmit 75% of the time to one STA and the rest to the other.

We start by having each STA in the network choose the amount of information they require in bits per second. Then, based on their SNR, we check how much airtime that would require based on their maximum achievable rate. We will use the rates achievable with IEEE 802.11ax, which we show in Table 5.4. We consider only one spatial stream and 20 MHz of channel bandwidth. We show the airtime calculation in Equation 5.7.

Explanation	**Name of the variable**	**Value**
Legacy preamble	$T_{\text{PHY-legacy}}$	$20\mu s$
HE Single-user preamble	$T_{\text{PHY-HE-SU}}$	$32\mu s$
OFDM symbol duration	σ	$16\mu s$
OFDM Legacy symbol duration	σ_{Legacy}	$4\mu s$
Short InterFrame Space	SIFS	$16\mu s$
DCF InterFrame Space	DIFS	$34\mu s$
Empty CSMA slot	T_e	$9\mu s$
Service Field	L_{SF}	32 bits
MAC header	L_{MH}	272 bits
Tail bits	L_{TB}	6 bits
Frame size	L_D	12000 bits
Average back-off	E[B]	7.5
Number of subcarriers	N_{Sc}	234
Number of Legacy subcarriers	N_{LSc}	52
Coding rate	Y_c	SNR dependent
Modulation used in bits/symbol	Y_m	SNR dependent
Transmission rate	$r(N_{Sc})$	SNR dependent

Table 5.3: Notation used

MCS	**Mod.** Y_m	**Coding rate** Y_c	**Rate (Mbps)**	**RP (dBm)**
0	BPSK	1/2	7.3	$[-82, -79)$
1	QPSK	1/2	14.6	$[-79, -77)$
2	QPSK	3/4	21.9	$[-77, -74)$
3	16-QAM	1/2	29.3	$[-74, -70)$
4	16-QAM	3/4	43.9	$[-70, -66)$
5	64-QAM	2/3	58.5	$[-66, -65)$
6	64-QAM	3/4	65.8	$[-65, -64)$
7	64-QAM	5/6	73.1	$[-64, -59)$
8	256-QAM	3/4	87.8	$[-59, -57)$
9	256-QAM	5/6	97.5	$[-57, -54)$
10	1024-QAM	3/4	109.7	$[-54, -51)$
11	1024-QAM	5/6	121.9	$[-51, \infty)$

Table 5.4: IEEE 802.11ax rates for 20 MHz and one spatial stream

To calculate the airtime percentage of a STA we start by calculating their transmission rate in bits per symbol in Equation 5.3, which depends on the modulation and coding rate used, which change with the SNR.

$$r(N_{Sc}) = N_{Sc} \cdot Y_c \cdot Y_m \tag{5.3}$$

Next we want to know the time it would take to transmit the frame data, the first two elements in Equation 5.4 are the legacy and high-efficiency preambles, these are transmitted at 20 MHz and legacy rate of 6 Mbps. The rest of the data, including the service field, tail bits and MAC header are sent at the maximum rate available, which is why they are divided by $r(N_{Sc})$ and multiplied by the OFDMA symbol time σ.

$$T_{\text{Data}} = T_{\text{PHY-legacy}} + T_{\text{PHY-HE-SU}} + \left(\frac{L_{\text{SF}} + L_{\text{MH}} + L_D + L_{TB}}{r(N_{Sc})} \right) \cdot \sigma \tag{5.4}$$

Equation 5.5 shows the time calculation for the acknowledgments that are received after each transmission. It works similarly to the previous equation, but in this case it is transmitted entirely in legacy parameters, which is why we use the legacy amount of subcarriers and the OFDM symbol time.

$$T_{\text{ACK}} = T_{\text{PHY-legacy}} + \left(\frac{L_{\text{SF}} + L_{\text{ACK}} + L_{TB}}{r(N_{LSc})} \right) \cdot \sigma_{\text{Legacy}} \tag{5.5}$$

We now calculate the time required for the entire process in Equation 5.6 we add the values of T_{Data} and T_{ACK}, plus the Short Interframe Space and DCF Interframe

Space, as well as the time for an empty slot, all according to the CSMA/CA protocol, to obtain the amount of time the air will be occupied.

$$T = T_{\text{Data}} + \text{SIFS} + T_{\text{ACK}} + \text{DIFS} + T_e \tag{5.6}$$

Finally, in Equation 5.7 we can transform this time into a percentage. We take the time we calculated, add to it the time of the average back-off and multiply it by the division of the bandwidth demanded and the size of our frames.

$$\text{STA Airtime Percentage} = \frac{\text{Bandwidth demanded}}{L_D} \cdot (T + \text{E[B]}) \tag{5.7}$$

Once we know how much time the STAs require to transmit, we check every AP that is in the same channel and add that percentage to their own. We show this in Equation 5.8.

$$\text{AP Airtime Percentage} = \sum_{\forall \text{ Stations}} \text{STA Airtime Percentage} \tag{5.8}$$

Finally, we check each AP's airtime percentage, and if the value is over 1 (meaning that all the STAs in that channel would need more than 100% of the airtime available to transmit), the AP gives them a proportional amount based on how much they need, as shown in Equation 5.9. If the AP has an airtime inferior or equal to 1, then they receive the airtime calculated in Equation 5.7 and the entirety of their desired bandwidth.

$$\text{STA Bandwidth achieved} = \text{ Bandwidth demanded} \cdot \left(\frac{\text{STA airtime}}{\text{AP airtime}} \right) \tag{5.9}$$

5.2.c Simulation approach

We will now discuss the way our simulation works[1]: We begin by setting a number of APs and STAs on our 2D plane. In this Chapter the APs will be in fixed positions and the STAs will generally be placed randomly.

The simulation then runs 101 iterations of association, following the pattern in Figure 5.1. Each iteration consists of two phases: the first one is the decision phase in which the association of the STAs to APs takes place. In the first iteration we use the SSF method so that each STA associates to the AP with the strongest received signal, and the 100 following ones use an ϵ-greedy algorithm that continually tries to find an optimal association. The second phase is the evaluation of the decision made. In this phase we check the satisfaction obtained by each STA and update their knowledge of the network. This is done by having the STA keep a list of each

[1]The code for this simulation can also be found in: https://github.com/MCarrascosaZ/Improving-user-association-in-HD-WLANs-using-Machine-Learning

AP, where every time an association is satisfactory the AP's reward increases. This information is used in the next decision phase by the ϵ-greedy algorithm to decide the next association, meaning that as time goes by the STA will make smarter decisions. The satisfaction of a STA is calculated through the airtime calculation shown in the previous subsection. If a STA receives its desired bandwidth from the AP it is satisfied, otherwise it is not.

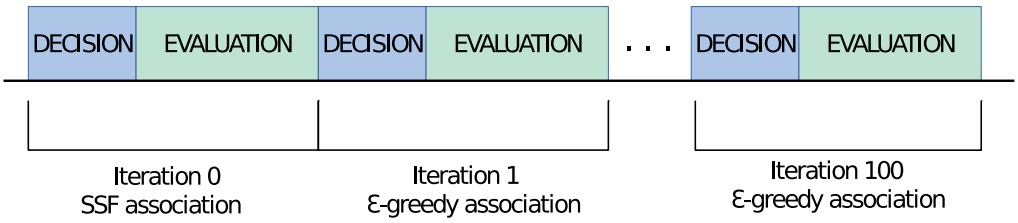


Figure 5.1: Structure of the simulation

5.3 Reinforcement Learning

Here we will introduce the multi armed bandit problem, a Reinforcement Learning technique that is often used to explain the type of algorithm we will use. We will present the standard ϵ-greedy algorithm and then expand upon it to create a version that suits our needs.

5.3.a Multi armed bandits

The multi armed bandits problem is a fairly popular Reinforcement Learning technique, it is based on the idea of a person playing several slot machines, with each one having a lever (the arm) that has a different probability of success. The objective is to find a sequence of lever pulls that maximizes the amount of times the gambler wins. Each pull of an arm receives a value of 1 if the gambler wins, or 0 if they do not. This means that we start with no knowledge of which lever to pull, but with each pull we get more information about each of the arms. The biggest challenge is to find the balance between using the knowledge already accumulated (exploit) or pulling levers at random (explore) to obtain more information that can possibly lead to a bigger exploitation.

5.3.b ϵ-greedy

We show the basic ϵ-greedy in Algorithm 1. We can see that it illustrates the multi armed bandits problem very clearly. First a value for ϵ is chosen between 0 and 1, and then for each agent (in this case the STAs), a random number is generated between 0 and 1, if the random number is smaller than the value of ϵ we explore and pick an AP at random. If the random number is higher than the value of ϵ we exploit our knowledge and pick the AP that has the maximum accumulated reward. It is common to pick a value for ϵ that changes over time, starting at 1 to ensure that the agent begins by exploring their options and then getting closer to 0, so that we exploit our knowledge almost constantly.

Algorithm 1: Implementation of ϵ-greedy in STAs to associate to the best AP possible

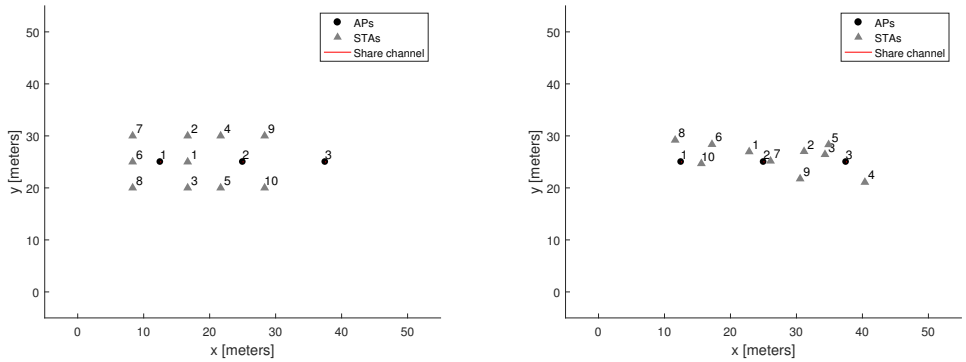
```
1  Function ϵ-greedy (STA);
   Input  : STA: struct containing all STAs and relevant information
2  initialize: ϵ ∈ [0, 1], Reward=0
3  for each STA do
4  |  if rand() < ϵ then
5  |  |  STA chooses any AP it senses at random
6  |  else
7  |  |  STA chooses AP with maximum accumulated reward
8  |  end
9  |  EvaluateAction()
10 |  if Action was positive then
11 |  |  Reward++
12 |  end
13 end
```

5.4 SSF, ϵ-greedy and STA placement

In this and future sections we will be comparing the ϵ-greedy algorithm to the Strongest Signal First association mechanism. One of the strengths of any machine learning algorithm is that they explore the possibilities that are available to them, which allows them to react to their environment. This is something that the SSF method does not do, and in this section we want to see how these methods of association deal with their environment, specifically the positions of the STAs.

We will do two different simulations: one will have the STAs in random positions (using a uniform distribution), and the other will use fixed positions. For the case in which we use fixed positions we will have most of the STAs around the first and second APs, and almost none near the third one. This is to test a network with an unbalanced placement of users, which will lead to some APs being congested and others underused. In the case of random STAs we will find that due to their uniform

distribution the load on the APs will be pretty even (on average). We show both configurations in Figure 5.2. For this section we use a limited range of placements for the STAs, since we want them to be able to sense all the APs so that they can recover from their bad placement in Figure 5.3a. We will talk about this limited range and its effects in Section 5.10.



(a) Fixed STAs (b) Random STAs

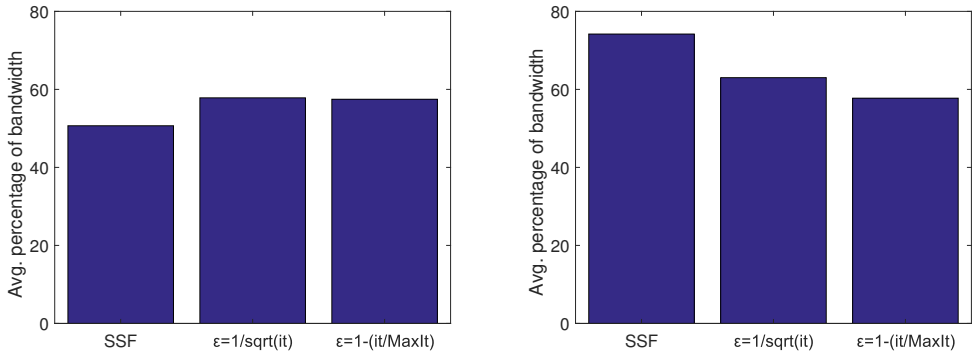
Figure 5.2: Node configurations

Our simulation will run 100 times for each configuration, in each simulation the first association is done through the SSF method, and after that, the ϵ-greedy algorithm will have 100 iterations in which to find the best possible association configuration, meaning that the full simulations is comprised of 10100 iterations.

In the random configuration the location of the STAs will be changed every simulation, and on the fixed configuration we will use the same positions but change the shadowing every time. The APs can also choose a channel out of 8 available, and do so randomly for every simulation. All the STAs will demand the same bandwidth of 7.5 Mbps.

We mentioned previously that it is desirable to use an ϵ value that changes over time. We will try two different ways of calculating this value based on the amount of iterations that have passed. The first one is $\epsilon = \frac{1}{\sqrt{iteration}}$, which starts at 1 but rapidly decreases, allowing us to exploit early; the other method is $\epsilon = 1 - \frac{iteration}{MaxIteration}$, which decreases linearly, meaning that we will explore quite a bit more than with the previous method.

Figure 5.3 shows the average percentage of bandwidth obtained after the last ϵ-greedy iteration of each one of the 100 simulations. In Figure 5.3a we see that both ϵ-greedy algorithms are better than SSF when the STAs are badly distributed, and in Figure 5.3b we can see that the opposite is true when the STAs are spread out uniformly. We can also see that using the square root of the iteration to calculate the value of ϵ gives better results than the other method.



(a) Fixed STAs (b) Random STAs

Figure 5.3: Percentage of bandwidth obtained

Table 5.5 shows some statistics for each case. The first field "Iterations that converged" shows the amount of time that the system found an association that satisfied every STA, meaning how many of the 10100 total iterations the system spent satisfied. The second field shows how many of the 100 simulations managed to find satisfaction for every STA at any point. This is related to the third field, which shows how many of the 100 simulations could be satisfied using only the SSF method. Finally, the fourth field compares the bandwidth obtained with each ϵ-greedy method and SSF. This is another representation of the information shown in Figure 5.3.

For the fixed STAs we see that our algorithms can find the optimal solution 65% and 68% of the time, while SSF is the optimal solution only 11 % of the simulations. An interesting detail however is that even if they find the optimal solution it is hard for them to enforce it. We see this in the Iterations that converged, where the linear approach stays converged 10.68% of the time, which ends up being less than the 11% obtained by SSF. The square root algorithm however reaches 15.28%, which is a slight increase over SSF performance. What happens here is that both approaches keep exploring throughout the simulation, which takes them away from the optimal solution. Still, on average we manage to give each STA a bigger amount of bandwidth by using any of the ϵ-greedy approaches.

When we distribute the STAs uniformly we see that SSF is much better as an option, since it finds the optimal solution 55% of the time, leaving both ϵ-greedy behind with a 28.3% and 12.45%, and with a decrease in bandwidth obtained for both cases.

Overall, it looks like the squared root method is the better of the two ϵ-greedy algorithms, and if we have an uneven STA distribution it may be worth using, but with a random uniform distribution of STAs we can see that SSF method outperforms them both.

Fixed STAs		
Algorithm	$\epsilon = \frac{1}{\sqrt{iteration}}$	$\epsilon = 1 - \frac{it}{MaxIt}$
Iterations that converged	15.28%	10.68%
Simulations where convergence was found	65%	68%
Satisfied with SSF	11%	11%
Avg. Bandwidth (vs.SSF)	+14,1%	+13.3%
Random STAs		
Algorithm	$\epsilon = \frac{1}{\sqrt{iteration}}$	$\epsilon = 1 - \frac{it}{MaxIt}$
Iterations that converged	28.3%	12.45%
Simulations where convergence was found	73%	71%
Satisfied with SSF	55%	55%
Avg. Bandwidth (vs.SSF)	−15.06%	−22.15%

Table 5.5: Statistics for each configuration

5.5 Our modifications to the algorithm

Now that we have seen the standard algorithm we adapt it to the situation we will be treating from now on. We will be using the concept of stickiness in a similar way as the one found in [35]. When a STA finds a satisfactory AP, a sticky counter will increase, which will make the STA exploit its knowledge as long as the counter does not go back to 0. This will allow the STAs to converge to a satisfactory association even if their ϵ value would force an exploration, and in the case of the sticky counter being higher than 1, if an iteration happens to be unsatisfactory due to some other STA's actions, we can keep the STA exploiting to see if the next iteration the situation fixes itself. We show the full algorithm in Algorithm 2.

5.6 Initial simulation

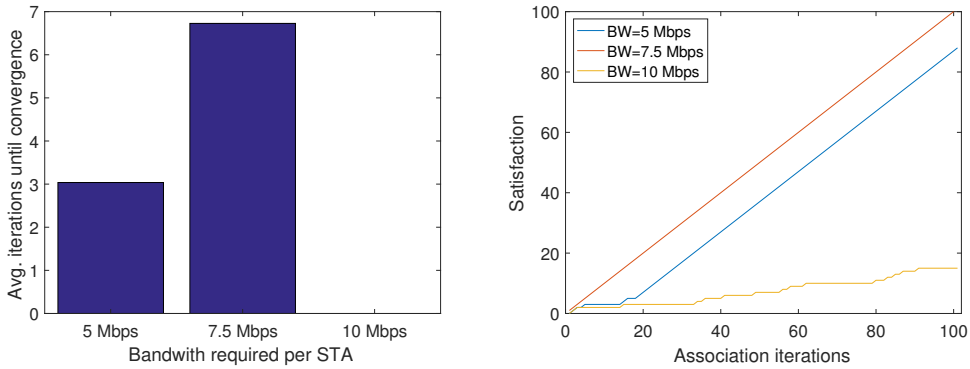
We will start by studying how the bandwidth required by each STA has an effect on the capacity of the algorithm to converge. We use 3 APs in a line and 10 randomly located STAs. This time the STAs can pick any position of the 50 by 50 area, as opposed to the limited range we gave them in the previous section. By doing this we allow situations in which STAs will only sense a single AP or maybe they will not sense even one. This benefits the SSF method, and we do it so that we can see if the ϵ-greedy algorithm can outperform it even in unfavourable situations. We have every STA demand the same amount of bandwidth from the APs and see if we can find an optimal association that can satisfy everyone. We use 5, 7.5 and 10 Mbps. Figure 5.4a shows the average number of associations required for all STAs to converge and find an association that keeps them satisfied. We can observe that we never converge when using 10 Mbps, while we find a solution really quickly in the other two cases. For 5 Mbps we barely need to have the STA try each AP once,

Algorithm 2: Implementation of ϵ-greedy with stickiness in STAs to associate to the best AP possible

```
1  Function ϵ-greedy (STA, iteration);
   Input  : STA: struct containing all STAs and relevant information
            iteration: Number of times the algorithm has been applied
2  initialize: ϵ ∈ [0, 1]
3  if STA.sticky > 0 then
4  |  ϵ = 0
5  else
6  |  ϵ ∈ (0, 1)
7  end
8  for each STA do
9  |  if rand() < ϵ then
10 |  |  STA chooses an AP at random as long as it does not have the max
   |  |   reward
11 |  else
12 |  |  STA chooses AP with maximum accumulated reward
13 |  end
14 end
15 APs update their airtime according to new associations
16 for each STA do
17 |  if STA received bandwidth < STA required bandwidth then
18 |  |  STA.satisfaction(iteration) = 0;
19 |  |  STA.sticky- -;
20 |  else
21 |  |  STA.satisfaction(iteration) = 1;
22 |  |  STA.reward(current_AP)++;
23 |  |  STA.sticky++;
24 |  end
25 end
```

and for 7.5 Mbps we need a bit more than double that amount. This would lead us to believe that as long as convergence is possible, we can find it pretty quickly.

Figure 5.4b shows the satisfaction of a single STA across 100 associations for a single simulation. In it we can see that with 5 and 7.5 Mbps we find satisfaction sooner or later and manage to stay satisfied after that, while with 10 Mbps our STA keeps trying to find a solution throughout the entire simulation.



(a) Avg. iterations until convergence (b) Satisfaction for one STA

Figure 5.4: Convergence

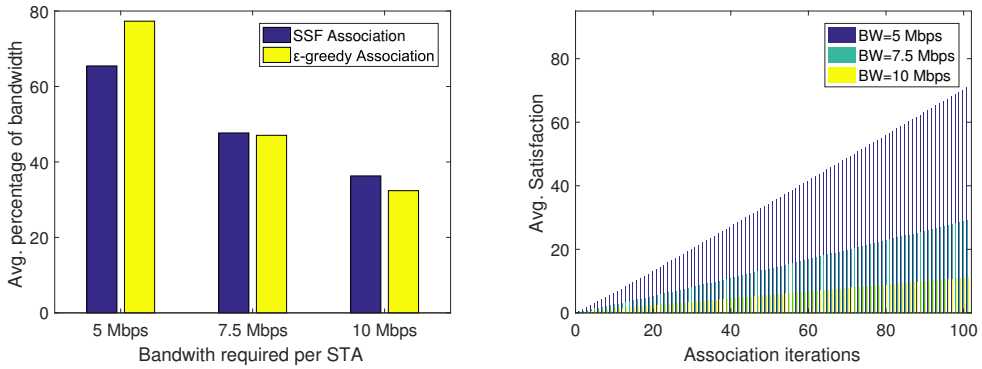
Table 5.6 shows some statistics from our simulations, the first is the amount of unassociated STAs. Since we locate the STAs randomly, they may end up in bad positions where they cannot reach an AP, and in those events we do not consider these STAs when we check if convergence is reached, as they can never be satisfied. The next fields function in the same way as they were presented in Table 5.5, back in Section 5.4. We see that the amount of bandwidth desired by the STAs has an effect on the capacity to reach convergence, with 10 Mbps making satisfaction impossible.

Field	**5 Mbps**	**7.5 Mbps**	**10 Mbps**
Avg. Unassociated STAs	0.12 %		
Max. Unassociated STAs	2		
Iterations that converged	52.89%	10.39%	0%
Simulations where convergence was found	61%	15%	Na
Satisfied with SSF	34 %	5%	Na

Table 5.6: Statistics for the simulation of 3 APs and 10 STAs

For the next figure we will look at satisfaction in two different ways: the first one is by just checking the satisfaction accumulated according to our algorithm; the second one is to check the last iteration of each simulation and compare the bandwidth received with the bandwidth demanded.In Figure 5.5a we use the second one, and compare the result obtained with our algorithm to the result we get in the first association iteration, in which we use SSF. We can observe that when using 5 Mbps we reach 77% satisfaction by the final iteration, which is an improvement of 18% over the SSF method; when using 7.5 Mbps we get very similar performances, both reach 47 % of the desired bandwidth with ϵ-greedy behind by only 0.6%, and when using 10 Mbps we get a performance decrease of 10.7%.

These results reinforce what we see in Figure 5.4, where we saw that the bandwidth required is a deciding factor on convergence being achievable. Figure 5.5b shows the average satisfaction for all STAs, where we see that the amount of times a STA can be satisfied decreases heavily when increasing the airtime demands, with 5 Mbps reaching more than 70 satisfaction, 7.5 Mbps struggling at less than 30 and 10 Mbps barely making it over 10.



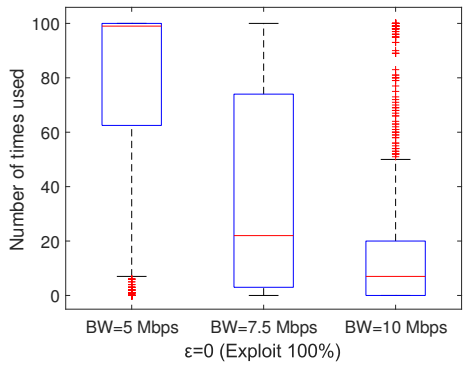
(a) Bandwidth achieved by association (b) Mean satisfaction

Figure 5.5: Bandwidth and satisfaction achieved

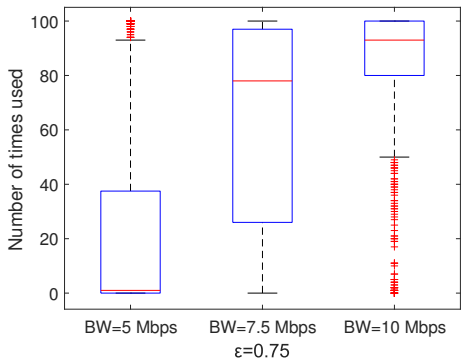
Our algorithm has two different states, the first one happens when the sticky counter is above 0, we will call this the sticky state; the other when the sticky counter is 0 or less, which we call regular state. In the sticky state the STA will always exploit (use $\epsilon = 0$) and re-associate to the AP with a higher reward, and as long as it stays satisfied this state will remain active. The regular state uses an epsilon between 0 and 1, so that the STA will alternate between exploring and exploiting, and for our simulations we have used $\epsilon = 0.75$ to compensate for the fact that the sticky state only exploits. The intention is for the STA to explore most of the time in this state, and then exploit with enough information to enter the sticky state.

Figures 5.6a and 5.6b show how many times the STAs spend on average in each state, and we can see that it behaves in the way we expect it, when convergence is easily found (5 Mbps) most of the simulation is spent in the sticky state, and as convergence becomes harder to find the STAs spend more and more time in the regular state, with 10 Mbps being almost exclusively in it.

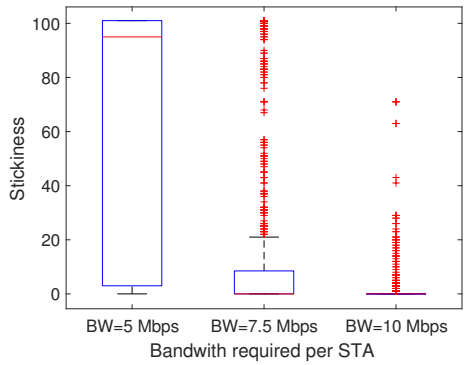
In Figure 5.6c we show the amount of times the sticky counter raised for each bandwidth. An interesting detail is that the higher outlier for 10 Mbps is 71, meaning that in 100 simulations not a single STA managed to remain satisfied the entire time.



(a) Iterations spent in sticky state



(b) Iterations spent in regular state



(c) Stickiness counter increases

Figure 5.6: States of the algorithm and STA stickiness vs. bandwidth

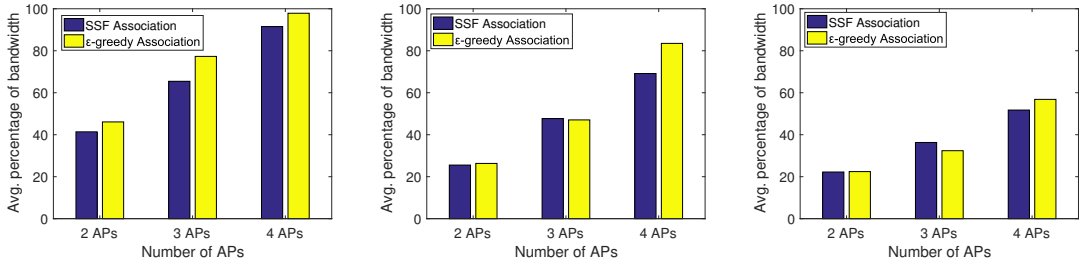
5.7 Different network topologies

For this section we use different AP configurations to see how our algorithm reacts to having a different amount of options to choose. We will compare the previous configuration of 3 APs with two others: the first one will use 2 APs set in a line across the x-axis and centered along the y-axis; the second one will use 4 APs spread in a square formation. We keep using 10 randomly located STAs for every simulation.

We begin by checking the bandwidth obtained at the end of each simulation when using the 3 bandwidths we used before, and the results are shown in Figure 5.7.

In the previous section we saw that when using 3 APs increasing the bandwidth demands led to a loss in bandwidth obtained when convergence was impossible, a loss that went up to 10.7% for 10 Mbps. If we look at Figure 5.7c we see something interesting in the 2 AP configuration: with both 7.5 Mbps and 10 Mbps convergence is impossible, yet its performance does not suffer much: it increases in the case of 7.5 Mbps, with a 3.05% increase over SSF, and only suffers a loss of 0.85% with 10 Mbps. In that case at least, it would seem like our algorithm can be considered successful, especially when we consider that when using 5 Mbps we reach an improvement of 11.37% over SSF.

When we use 4 APs we can finally achieve convergence at 10 Mbps, and so the algorithm results in an improvement in all cases, we get a 6.96%, 22.54% and 9.83% improvement for 5, 7.5 and 10 Mbps respectively. The small improvement in the first scenario is probably due to the fact that with SSF we reach 91% satisfaction already, so there is little room for improvement there.



(a) STAs demand 5 Mbps (b) STAs demand 7.5 Mbps (c) STAs demand 10 Mbps

Figure 5.7: Bandwidth achieved with different requirements

In Table 5.7 we show the same statistics we obtained in 5.6 for 3 APs and compare them to the new configurations. The first thing we can see is that the more APs we add to the system the longer the system can stay converged, with 2 APs and 5 Mbps we barely get a 15% of the time of all STAs satisfied, and with 4 APs we manage 92.96%. If we compare the iterations that converged and the time we would be satisfied with SSF we can see the improvement that the algorithm brings in each case. With 2 APs and 5 Mbps the SSF method works 8% of the time, but with our ϵ-sticky algorithm we reach 15% of satisfaction, almost twice as much. The biggest difference is in 4 APs with 7.5 Mbps, where we get satisfaction almost a third of the

time with SSF, and close to half the time with ϵ-sticky. If we compare the iterations that converged and the simulations where convergence was found we see that the first one is always smaller than the second one. This is because of the exploration period, if we were able to find convergence on the first iteration, then this values would coincide. In a way, it is a representation of how much time is needed to obtain enough information to reach satisfaction.

2 APs			
Bandwidth	**5 Mbps**	**7.5 Mbps**	**10 Mbps**
Avg. Unassociated STAs	0.26 %		
Max. Unassociated STAs	3		
Iterations that converged	15.01%	0%	0%
Simulations where convergence was found	19%	Na	Na
Satisfied with SSF	8 %	Na	Na
3 APs			
Bandwidth	**5 Mbps**	**7.5 Mbps**	**10 Mbps**
Avg. Unassociated STAs	0.12 %		
Max. Unassociated STAs	2		
Iterations that converged	52.89%	10.39%	0%
Simulations where convergence was found	61%	15%	Na
Satisfied with SSF	34 %	5%	Na
4 APs			
Bandwidth	**5 Mbps**	**7.5 Mbps**	**10 Mbps**
Avg. Unassociated STAs	0.02 %		
Max. Unassociated STAs	1		
Iterations that converged	92.69%	48.99%	11.66%
Simulations where convergence was found	94%	58%	16%
Satisfied with SSF	83 %	30%	2 %

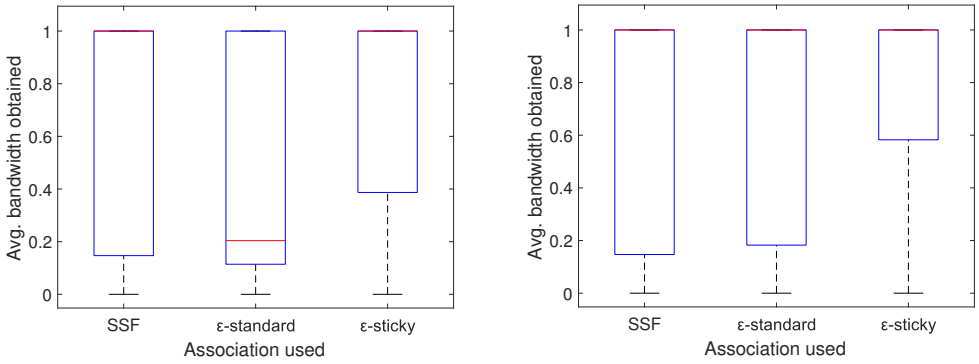
Table 5.7: Statistics when changing the amount of APs

With the results presented in this section we can say that as long as convergence is possible (which depends on the area covered, the amount of APs and STAs, as well as their requirements in bandwidth), ϵ-sticky will find an optimal solution that is better than the SSF method.

5.8 Stickiness vs. regular algorithm

In this section we want to compare the effectiveness of the regular ϵ-greedy algorithm to our modified one with a sticky counter. The regular algorithm would always go into the regular state we described earlier, and its effectiveness depends on the ϵ used. In the previous sections we used a fixed ϵ of 0.75 to compensate for the high exploitation of the sticky state, but if we were to only use the regular state, we would want the ϵ to change with time, so that it starts close to 1, high exploration, and gets closer to 0 with time, exploiting the knowledge accumulated. We will use the fixed ϵ, as well as one that changes with the number of iterations that pass. We use $\frac{1}{\sqrt{iteration}}$, which at the first iteration gives us $\epsilon = 1$ and in the last one $\epsilon = 0.1$. For this entire section, we use the same simulation parameters as section 1: 3 APs and 10 STAs.

Figure 5.8a shows the results for the fixed $\epsilon = 0.75$. In this case both SSF and our ϵ-sticky implementation are superior to the ϵ-standard implementation. They both have medians of 1 and their 25th percentiles are higher. In Figure5.8b we can see a massive improvement by the ϵ-standard implementation, with not only a median of 1 but a higher 25th percentile than SSF. ϵ-sticky however seems to have improved even more in this case, with its 25th percentile going from 40% bandwidth obtained to 60%. Considering the other two have theirs around 20%, it seems that the sticky state does improve the basic algorithm.



(a) $\epsilon = 0.75$ (b) $\epsilon = 1/\sqrt{\text{iteration}}$

Figure 5.8: Changing the epsilon value

While Figure 5.8 implies that using a decreasing ϵ improves the ϵ-sticky implementation, Table 5.8 shows a different story: the STAs in our simulation spent less time converged when using a dynamic ϵ. This means that we can trade some of our capacity to find convergence for a slight overall increase in bandwidth acquired. Considering that our median stays at 1 for both cases, we have decided to prioritize our capacity to converge, and keep using a fixed ϵ.

$\epsilon = 0.75$		
Bandwidth	**Standard**	**Sticky**
Iterations that converged	4.34%	52.89%
Simulations where convergence was found	53%	61%
Satisfied with SSF	34 %	34%
$\epsilon = 1/\sqrt{\textbf{iteration}}$		
Bandwidth	**Standard**	**Sticky**
Iterations that converged	20.02 %	50.1%
Simulations where convergence was found	53%	60%
Satisfied with SSF	34 %	34%

Table 5.8: Statistics as the ϵ value changes

5.9 Tweaking the algorithm

In this section we will add some slight variations of our ϵ-sticky algorithm. Since we played with the ϵ value in the previous section we will not be changing it here and use the same one throughout. There are two changes we want to address: the first one is to use non-binary rewards, and the second one is to change the action to perform when the sticky counter is above 0.

For this section we will still use 3 APs with 10 randomly placed STAs, as well as using two bandwidths: 5 Mbps and 10 Mbps, to showcase the effects of the changes when convergence is easily achievable and when it is not.

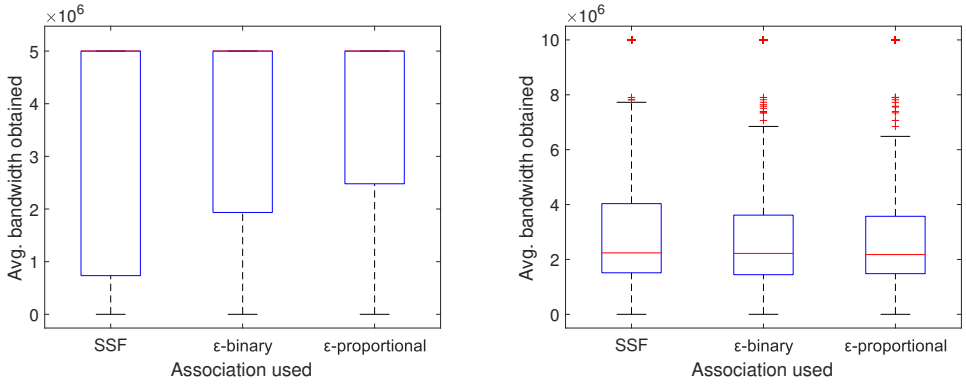
5.9.a Non-Binary rewards

In our implementation of the algorithm we have only used binary rewards, meaning that we only update the reward for an AP we have chosen if the STA is completely satisfied. This means that if a STA does not get all the bandwidth it wants it does not take into consideration if it received half of it in one AP and a third of it in another. For this section we will have the STAs update the reward of each AP with the proportion of bandwidth that they received, meaning that an AP still gets a 1 when it satisfies the STA, and when it does not we set the reward to whichever proportion we get from 0 to 0.99.

This small change means that each iteration obtains some information from the network, while previously we could go several iterations without updating the reward for any of the APs the STA senses; as a result, the regular state of the STA always acted like it had $\epsilon = 1$. Now even in cases where full satisfaction is unreachable, when the STA exploits it will be going to the AP that has given it the most bandwidth out of all available ones, instead of a random one.

We show our results in Figure 5.9, we call our ϵ-sticky implementation ϵ-binary here to reflect how we address the reward updating, and the new one ϵ-proportional.

Figure 5.9a shows that for 5 Mbps using proportional rewards lands us an increase in bandwidth obtained, with the 25th percentile going from 1.93 Mbps using binary rewards to 2.47 Mbps, a 28% increase. In Figure 5.9b however, the opposite happens, and the proportional rewards get the lowest value. In this case however, the decrease is not so substantial, with a 1.8% decrease in the median when comparing to binary rewards. This could be considered an overall improvement.



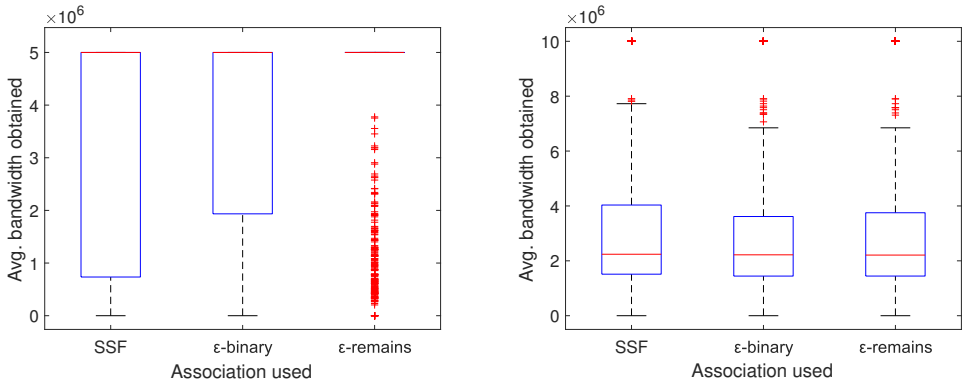
(a) Each STA demands 5 Mbps (b) Each STA demands 10 Mbps

Figure 5.9: Bandwidth achieved with each association

5.9.b Exploit vs. inaction

For this section we will keep the rewards binary again, and change how the sticky state behaves. Now, instead of exploiting when the sticky counter is above 0, the STA will just stay associated to its AP. With this change we would ideally evade the situation in which a satisfied STA would change to another AP due to the rewards of two APs being similar; or allow an AP's reward to catch up to the others if it had fallen behind because of the STA sticking to another AP.

We show the results in Figure 5.10. We call this new approach ϵ-remains, and keep the nomenclature of ϵ-binary for our initial implementation.



(a) Each STA demands 5 Mbps (b) Each STA demands 10 Mbps

Figure 5.10: Bandwidth achieved with each association

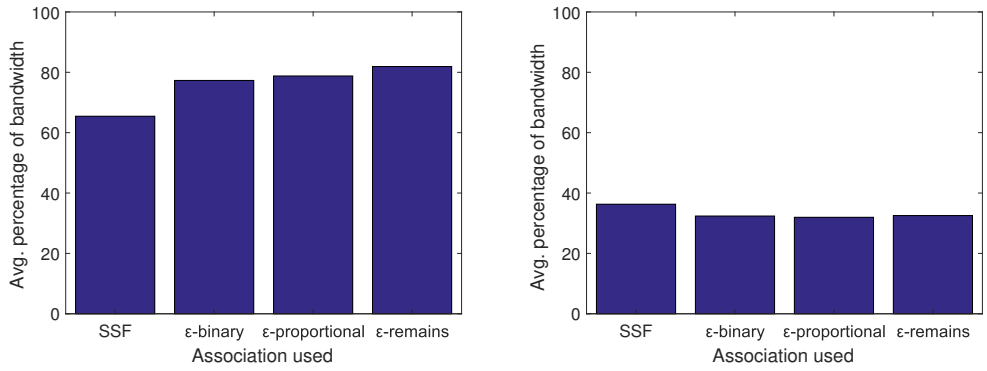
Figure 5.10a shows an impressive difference in the 25th percentile of the ϵ-remains approach, with an increase of 158%, going from under 2 Mbps to 5 Mbps. Still, much like before we see a decrease in the median when using 10 Mbps. This time it decreases even less than the ϵ-proportional implementation however, with a decrease of a 0.4% comparing with ϵ-binary.

5.9.c Comparison

We will now compare all of our implementations and find which one works better overall, first we will do this by checking the average bandwidth obtained and then we will look at the convergence statistics like in previous sections.

We start by checking the average bandwidth obtained with each implementation in Figure 5.11. The first thing we can notice is that all of them work the same way in relation to SSF, improving the performance when convergence is available and slightly decreasing it when convergence is unreachable. In the case of 5 Mbps in Figure 5.11a we can see that ϵ-remains is undoubtedly the best approach, giving us a 25.1% increase in bandwidth reached when comparing to SSF. While the others also give us better performance, ϵ-remains gives us almost 5% more than the others. It is also the superior one in Figure 5.11b, where it has a 10.3% decrease over SSF, which is the lowest decrease available, with ϵ-binary being almost identical with 10.7% and ϵ-proportional having the worst one with an almost 12 % decrease.

It is worth noting that while ϵ-proportional and ϵ-remains outperform ϵ-binary in general, if we combine them both we get an average bandwidth of 75.9%, making it the worst of the bunch.



(a) STAs demand of 5 Mbps (b) STAs demand 10 Mbps

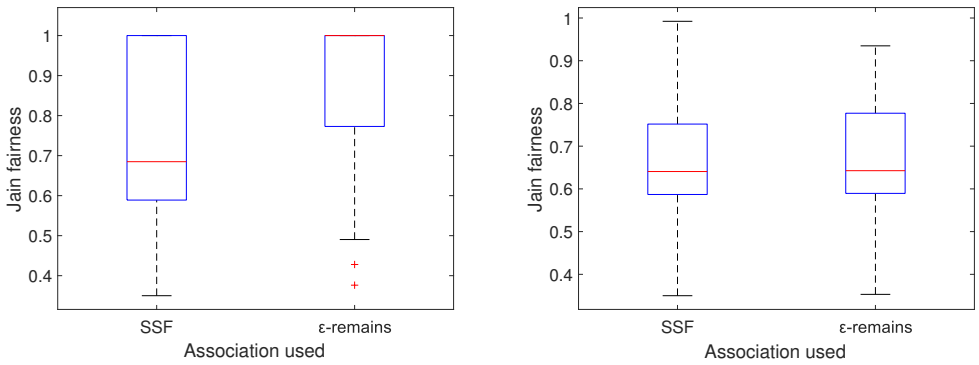
Figure 5.11: Comparison of bandwidth achieved

Much like in previous sections, we now check the convergence statistics for each algorithm in Table 5.9. We can see that the results align with the ones in Figure 5.11, with ϵ-remains having the highest time spent converged. And while it loses to ϵ-proportional in the capacity to find convergence, ϵ-remains is still the better option due to reaching higher bandwidths.

5 Mbps			
Algorithm	**binary**	**proportional**	**remains**
Iterations that converged	52.89%	55.28%	57.64%
Simulations where convergence was found	61%	63%	60%
Satisfied with SSF	34%	34%	34%
Avg. Bandwidth	77.31%	78.79%	81.91%
Avg. Bandwidth vs.SSF	+18,1%	+20.3%	+25.1%
10 Mbps			
Algorithm	**binary**	**proportional**	**remains**
Avg. Bandwidth	32.39%	31.97%	32.53%
Avg. Bandwidth vs.SSF	−10.7%	−11.9%	−10.3%

Table 5.9: Statistics for each algorithm

We have mentioned several times how the ϵ-greedy algorithm works well when convergence can be reached, but has some flaws when it is unreachable. The reason for this comes from it being a decentralized approach to association. Much like with SSF, every STA looks only for its own satisfaction, meaning that there will be cases where a STA with a bad connection will remain associated to an AP while blocking the others and taking all of the airtime available, making it an unfair system. Since we value bandwidth satisfaction and not fairness, this problem appears whenever convergence is unavailable. When convergence can be reached however, the system does find a fairer situation in which everyone is satisfied. We show this in Figure 5.12, where we use Jain's fairness index with the bandwidth obtained by each STA at the end of each simulation. Figure 5.12a shows that with 5 Mbps we can reach a median of 1, making it a fair system, and greatly improving on SSF. But when we use 10 Mbps our fairness looks similar to that of SSF, and even worse, since our upper adjacent is less than 1, while SSF can reach 1.



(a) Fairness when STAs ask for 5 Mbps (b) Fairness when STAs ask for 10 Mbps

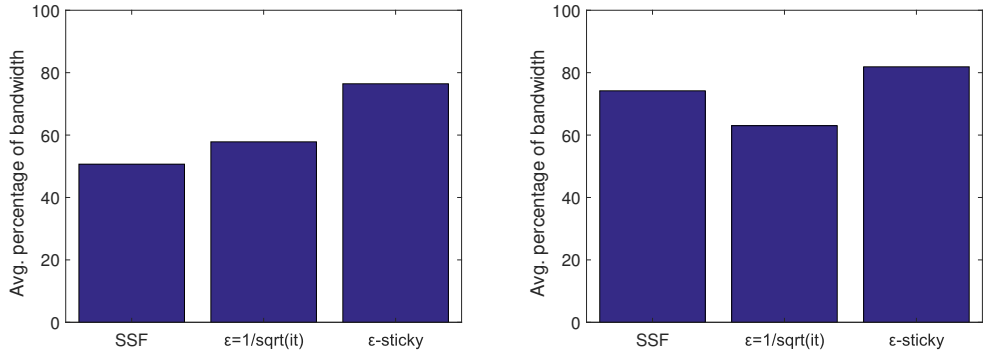
Figure 5.12: Jain's fairness

5.10 A second look at fixed STA positions

Now that we have seen several modifications of our ϵ-sticky algorithm, we want to go back to the configurations we used in Section 5.4. We saw there that if the STAs are all clustered around some of the APs, the SSF method performs badly because it does not distribute the load across all APs, and instead overloads some and leaves others without users. We also saw that the standard ϵ-greedy with $\epsilon = \frac{1}{\sqrt{iteration}}$ could outperform SSF in both the capacity of reaching convergence as well as reaching a higher amount of bandwidth per STA. We will now compare the standard ϵ-greedy and our sticky variation in a case that should benefit them both, and see which one performs better than the other.

We will use the ϵ-remains sticky version we saw in the previous section, which has the STA stay in whichever AP satisfies it and only uses binary rewards. We choose this because it outperforms every other variation we tried.

Figure 5.13 shows the results when we use both configurations of STAs. We can see that for both cases the sticky algorithm outclasses the standard one, and that with this configuration in which most STAs see every AP, the sticky version can work better than the SSF method even with the random STAs.



(a) Fixed STAs (b) Random STAs

Figure 5.13: Percentage of bandwidth obtained

Let us concentrate on the fixed STAs case. We can see that the sticky algorithm can find the optimal solution 61% of the time, which is less than the 65% that the standard algorithm can do. The sticky algorithm however, can enforce the optimal solution 55.28% of the time, while the standard algorithm can do so only 15.28% of the time. So we can see that the sticky algorithm uses the information acquired far better than the standard one, and it also manages to reach a much higher bandwidth per node.

In the case of random STAs, we can see that the sticky algorithm can find the optimal solution 73% of the times, same as the standard algorithm, but it can enforce it 70.24% of the time, making it the superior option. It also manages to improve the bandwidth obtained, while the standard one loses to SSF.

Fixed STAs		
Algorithm	$\epsilon = \frac{1}{\sqrt{iteration}}$	**ϵ-sticky**
Iterations that converged	15.28%	55.28%
Simulations where convergence was found	65%	61%
Satisfied with SSF	11%	11%
Avg. Bandwidth (vs.SSF)	+14,1%	+50.82%
Random STAs		
Algorithm	$\epsilon = \frac{1}{\sqrt{iteration}}$	**ϵ-sticky**
Iterations that converged	28.3%	70.24%
Simulations where convergence was found	73%	73%
Satisfied with SSF	55%	55%
Avg. Bandwidth (vs.SSF)	−15.06%	+10.3%

Table 5.10: Statistics for each configuration

For the fixed STAs case, we know the STA positions and implemented an association that would work optimally, one that we could call the optimal solution to the problem. Since the APs still choose the channel randomly and the shadowing is changed every iteration we could not find an association that works 100% of the time, but we found one that manages to converge 66.45% of the time, and offered an increase in bandwidth of 62.7% over SSF. We can see that this is not so far from the 55.28% that our sticky algorithm achieves, especially if we consider that we enforced the association by hand instead of reaching it through any exploration, meaning that no time was lost exploring options.

Previously, we mentioned that the uniform distribution of STAs benefits the SSF method, but in this section we saw that what actually has an effect on the effectiveness of the ϵ-greedy algorithm is not only the distribution used when placing STAs, but how many options the algorithm has to explore. In the previous sections we have many situations with a STA only sensing a single AP, meaning that the algorithm can not do anything in that case. Here however, since we place the STAs close enough to see most if not all the APs, they can explore more options and have a bigger chance of finding an optimal solution.

5.11 Variable bandwidth

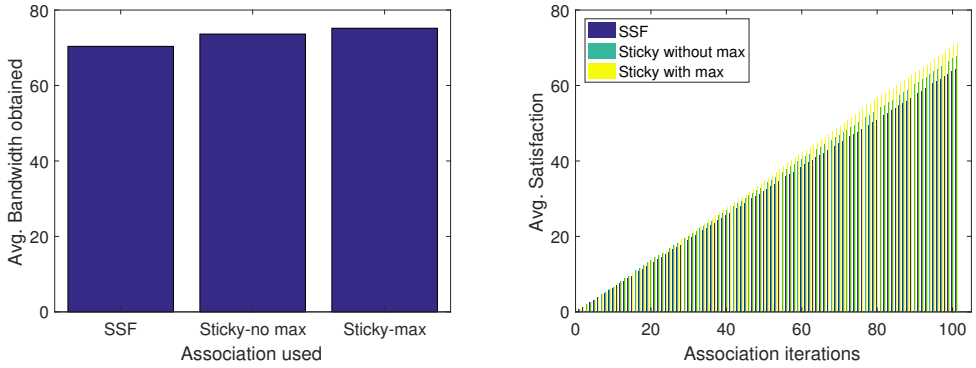
In the previous sections we had all the STAs demanding the same bandwidth during the entire simulation, and since a STA would increase its sticky counter by one for each iteration that was successful, we could find sticky counters that reached over 50, meaning that if the STA suddenly became unsatisfied it would take 50 iterations for the sticky counter to return to 0. In this section we will now have the STAs have different demands to show how the algorithm reacts to these changes. We will use the ϵ-sticky with binary rewards and the implementation of the sticky state in ϵ-remains that has the STA stay associated to its AP when it is satisfied. We will

also implement a limit on the sticky counter to make it easier for the STA to reset it.

Like before, we use 3 APs, 10 randomly placed STAs and have every STA pick a bandwidth at random from 0 to 10 Mbps, as well as have them pick a new bandwidth in iterations 20,40, 60 and 80.

In Figure 5.14a we show the bandwidth that the STAs have on the last iteration of the simulation, in which we should see if the algorithm has reacted properly to the changes in the network and managed to find an optimal solution for the last change in the STAs demands. As we can see, both ϵ-sticky implementations manage to outperform SSF by the end of the simulation, ϵ-sticky without limiting the sticky counter offers an improvement of 4.64% over SSF, while limiting the sticky counter offers the bigger improvement of the two, with a 6.83 % over SSF.

In Figure 5.14b we show the average accumulated satisfaction across the entire simulation, there SSF reaches 64.3, while ϵ-sticky unlimited reaches 67.82 and ϵ-sticky limited manages to get 71.28. Both figures suggest the ϵ-sticky approach can adapt to the networks' demands changing, and that limiting the sticky counter is a better option. And if we look at Table 5.11 we can see the same, with the limited ϵ-sticky managing to spend more than half the time converged, while the others stay close to 40%.



(a) Avg. bandwidth in the last iteration (b) Avg. satisfaction

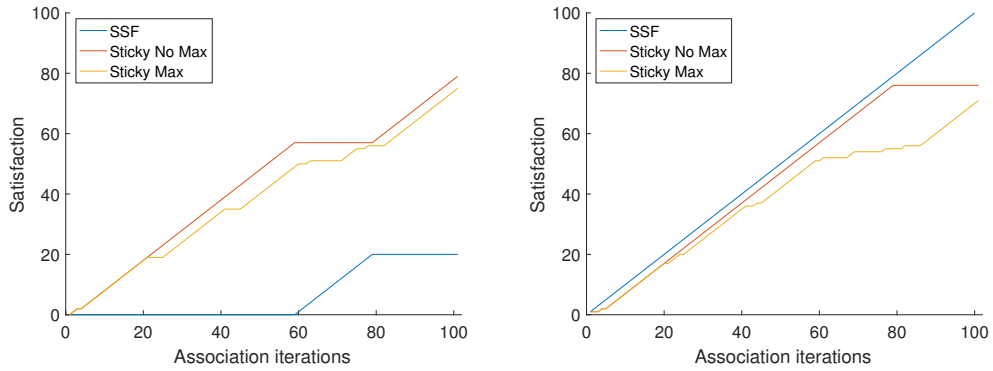
Figure 5.14: Performance when the demands change

Algorithm	SSF	Sticky No Max	Sticky Max
Iterations that converged	40.13%	42.44%	51.05%
Avg. Bandwidth in last iteration	70.37 %	73.64%	75.18%

Table 5.11: Statistics for each approach

Finally we want to show a particular iteration and how the STAs react with each algorithm. We have chosen two STAs in this iteration that have very different accumulated satisfaction counters, and we show their progression over time in Figures 5.15. The first STA in Figure 5.15a shows that SSF keeps the STA unsatisfied until

iteration 60, and then it is satisfied until iteration 80, where it becomes unsatisfied again. The unlimited ϵ-sticky shows the opposite progression, with the STA being satisfied every iteration except for the interval between 60 and 80. The limited ϵ-sticky struggles every once in a while but manages to find satisfaction in pretty much every interval. In this figure we can see that the unlimited ϵ-sticky had such a bloated sticky counter that it could not react to the change in the network before there was another change at iteration 80.



(a) Satisfaction of STA 1 (b) Satisfaction of STA 2

Figure 5.15: Performance of two STAs in a single simulation

In Figure 5.15b we see the interesting case in which a STA was satisfied with SSF the entire time, but when applying ϵ-sticky it struggles a bit more to be satisfied. Much like before, in the interval between 80 and 100 the sticky counter for the unlimited ϵ-sticky is so bloated that it can not react before the simulation ends, while the limited ϵ-sticky manages to recover in iteration 87.

In Table 5.12 we show how much time each algorithm managed to stay converged. And while Figure 5.15 could lead us to believe that the unlimited ϵ-sticky could be better than the limited one, we see here that is not the case, as the limited ϵ-sticky manages to converge for more periods of time, as well as longer amounts. It can react to the change in iteration 20 by iteration 26, while the unlimited one takes 9 iterations longer, it manages to adapt to the change in iteration 40 in 6 iterations, and takes 7 iterations to adapt to the change in iteration 80, while the limited ϵ-sticky never does.

Algorithm	**SSF**	**Unlimited Sticky**	**Limited Sticky**
Intervals	$60-79$	$6-19,\ 35-39,\ 45-59$	$6-19,\ 26-39,\ 46-59,\ 87-101$

Table 5.12: Convergence intervals for each algorithm

With all the changes in mind, this is our final algorithm:

Algorithm 3: Implementation of ϵ-greedy with stickiness in STAs to associate to the best AP possible

```
Function ϵ-greedy (STA, iteration);
  Input  : STA: struct containing all STAs and relevant information
           iteration: Number of times the algorithm has been applied
initialize: ϵ ∈ [0, 1]
if STA.sticky > 0 then
    ϵ = −1
else
    ϵ = 0.75
end
for each STA do
    if rand() < ϵ then
        STA chooses an AP at random as long as it does not have the max
         reward
    else
        if ϵ ≥ 0 then
            STA chooses AP with maximum accumulated reward
        else
            STA stays in the same AP
        end
    end
end
APs update their airtime according to new associations
for each STA do
    if STA received bandwidth < STA required bandwidth then
        STA.satisfaction(iteration) = 0;
        if STA.sticky>10 then
            STA.sticky=0;
        else
            STA.sticky- -;
        end
    else
        STA.satisfaction(iteration) = 1;
        STA.reward(current_AP)++;
        STA.sticky++;
    end
end
```

5.12 Conclusions

Machine Learning has the potential to improve HD-WLANs reach better performance, we have seen this by creating an algorithm that can find an optimal association where Strongest Signal First cannot.

We have also shown that both mechanisms have different strengths: when the scenario is uniform the SSF method can be really close to the optimal solution, since it will associate each STA to its closer AP, spreading the STAs evenly. When the scenario is not uniform then SSF cannot compete with the ϵ-greedy mechanism, which can distribute the STAs among the APs and find a better solution through trial and error.

This means that as they are now, we require knowledge of the user distribution to choose the better option. An algorithm that outperforms them both SSF and our implementation of ϵ-sticky would need to use aspects from each of them, be it through acquiring information on the distribution of the network or by defaulting to SSF if ϵ-sticky cannot reach an optimal solution past a certain amount of time.

If we consider that both association mechanisms are flawed, then it is clear why SSF is the undisputed standard. SSF is a simple mechanism that does not need any extra information or steps. A STA just needs to receive a signal for SSF to work. On the other hand, ϵ-greedy or any other Reinforcement Learning algorithm will require time to reach their potential, as well as being able to re-associate from one AP to another, which means that service could be interrupted. This is why ϵ-greedy will need to outperform SSF in every possible user distribution to succeed it, because we can say that SSF is unoptimal, but it has no major drawbacks in terms of setup. On the other hand, Machine Learning algorithms will always have a setup time, but in the future we believe that their benefits will make their drawbacks worth it.

Chapter 6

Conclusions

Over the course of this work we have seen that the 802.11 protocol has received a lot of changes over the years, and some of the most recent amendments are trying to improve user association. The 802.11ai amendment is going to improve the roaming capabilities of STAs, while 802.11k and the neighbor report starts to move the standard towards smart networks where information is shared between nodes to improve the user experience.

We have studied the hidden node problem in HD-WLANs, and how proper planning of the network is necessary for it to not become a problem. We have seen that channel allocation is necessary to mitigate it, that increasing the density of APs in the network can actually be beneficial in reducing the problem. We have also checked the effectiveness of CCA and power control, and seen that these methods will decrease the amount of exposed nodes at the cost of adding more hidden nodes. In the case of CCA control we can consider these exchange worthwhile, since the increase in hidden nodes is almost negligible compared to the decrease in exposed nodes. In the case of power control, as well as using both techniques combined, we have seen that they create far more issues than they solve, and that it is not worth it to use them.

Another way to improve the network's performance is to change the way STAs associate with APs, we have studied how Machine Learning algorithms can help the STAs obtain a better association, improving overall user satisfaction. These algorithms require time and information to work properly however, and it will take some tweaking before SSF can be dethroned as the main user association mechanism.

Looking forward, networks are going to be getting smarter with time, some manufacturers are already implementing load control mechanisms in their APs to mitigate the weaker aspects of the SSF mechanism. IEEE 802.11 amendments are also moving towards this, with the 802.11k and the neighbor report, which will receive improvements in the 802.11ai amendment, compressing the information sent in the report, as well as reducing the re-association time between APs. It is possible that in the future a single re-association will be enough to obtain the optimal association between STA and APs.

Bibliography

[1] Matthew S. Gast. *802.11 Wireless Networks: The Definitive Guide, 2nd Edition*. O'Reilly Media, Inc., 2005.

[2] Glenn Judd and Peter Steenkiste. Fixing 802.11 access point selection. *ACM SIGCOMM Computer Communication Review*, 32(3):31–31, 2002.

[3] Anand Balachandran, Geoffrey M. Voelker, Paramvir Bahl, and Venkat P.Rangan. Characterizing user behavior and network performance in a public wireless lan. *ACM SIGMETRICS Performance Evaluation Review - Measurement and modeling of computer systems*, 30(1):195–205, 2002.

[4] Magdalena Balazinska and Paul Castro. Characterizing mobility and network usage in a corporate wireless local-area network. *MobiSys '03 Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 303–316, 2003.

[5] Sangho Shin, Andrea G.Forte, Anshuman Singh Rawat, and Henning Schulzrinne. Reducing mac layer handoff latency in ieee 802.11 wireless lans. *MobiWac '04 Proceedings of the second international workshop on Mobility management & wireless access protocols*, pages 19–26, 2004.

[6] Tingting Sun, Yanyong Zhang, and Wade Trappe. Improving access point association protocols through channel utilization and adaptive switching. *Eighth IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, 2011.

[7] Dawei Gong and Yang Yuanyuan. Ap association in s02.lln wlans with heterogeneous clients. *2012 Proceedings IEEE INFOCOM*, pages 1440–1448, 2012.

[8] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, and D. Towsley. Facilitating access point selection in ieee 802.11 wireless networks. *IMC '05 Proceedings of the 5th ACM SIGCOMM conference on Internet measurement*, pages 26–26, 2005.

[9] Wei Li, Shengling Wang, Yong Cui, Xiuzhen Cheng, Ran Xin, A. Al-Rodhaan Mznah, and Abdullah Al-Dhelaan. Ap association for proportional fairness in multirate wlans. *IEEE/ACM Transactions on networking*, 22(1), 2014.

[10] Mohammed Amer, Anthony Busson, and Isabelle Guérlin Lassous. Association optimization in wi-fi networks: Use of an access-based fairness. *The 19th ACM*

International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM), pages 119–126, 2016.

[11] Cisco. Configuring aggressive load balancing in cisco wireless lan controller configuration guide, release 7.4, 2017. Accessed: 08/06/2018.

[12] Cisco. The benefits of centralization in wireless lans via the cisco unified wireless network, 2006.

[13] Yunze Zeng, Ioannis Pefkianakis, Kyu-Han Kim, and Prasant Mohapatra. Mu-mimo-aware ap selection for 802.11ac networks. *Mobihoc '17 Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2017.

[14] K. Heck. Wireless lan performance in overlapping cells. In *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484)*, volume 5, pages 2895–2900 Vol.5, Oct 2003.

[15] Z. Zhong, P. Kulkarni, F. Cao, Z. Fan, and S. Armour. Issues and challenges in dense wifi networks. In *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 947–951, Aug 2015.

[16] M. Drieberg, F. C. Zheng, R. Ahmad, and M. Fitch. Impact of interference on throughput in dense wlans with multiple aps. In *2009 IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 752–756, Sept 2009.

[17] A. Ozyagci, K. W. Sung, and J. Zander. Effect of propagation environment on area throughput of dense wlan deployments. In *2013 IEEE Globecom Workshops (GC Wkshps)*, pages 333–338, Dec 2013.

[18] Natsumi Kumatani, Mitomo Isomura, Tutomu Murase, Masato Oguchi, Akash Baid, Shweta Sagari, Ivan Seskar, and Dipankar Raychaudhuri. [paper] throughput characteristics in densely deployed wireless lans with various channel assignments and differing numbers of terminals. *ITE Transactions on Media Technology and Applications*, 2(4):336–344, 2014.

[19] M. Drieberg, F. C. Zheng, R. Ahmad, and S. Olafsson. An asynchronous distributed dynamic channel assignment scheme for dense wlans. In *2008 IEEE International Conference on Communications*, pages 2507–2511, May 2008.

[20] T. Tandai, K. Toshimitsu, and T. Sakamoto. Interferential packet detection scheme for a solution to overlapping bss issues in ieee 802.11 wlans. In *2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, Sept 2006.

[21] Vivek P. Mhatre and Konstantina Papagiannaki. Optimal design of high density 802.11 wlans. In *Proceedings of the 2006 ACM CoNEXT Conference*, CoNEXT '06, pages 8:1–8:12, New York, NY, USA, 2006. ACM.

[22] H. Ma and S. Roy. Contention window and transmission opportunity adaptation for dense ieee 802.11 wlan based on loss differentiation. In *2008 IEEE International Conference on Communications*, pages 2556–2560, May 2008.

[23] B. Han, L. Ji, S. Lee, R. R. Miller, and B. Bhattacharjee. Channel access throttling for overlapping bss management. In *2009 IEEE International Conference on Communications*, pages 1–6, June 2009.

[24] Yue Fang, Daqing Gu, A. B. McDonald, and Jinyun Zhang. A two-level carrier sensing mechanism for overlapping bss problem in wlan. In *2005 14th IEEE Workshop on Local Metropolitan Area Networks*, pages 6 pp.–6, Sept 2005.

[25] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. C. Chen, and L. Hanzo. Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Communications*, 24(2):98–105, April 2017.

[26] L. Stabellini and J. Zander. Interference aware self-organization for wireless sensor networks: A reinforcement learning approach. In *2008 IEEE International Conference on Automation Science and Engineering*, pages 560–565, Aug 2008.

[27] N. Rupasinghe and İ. Güvenç. Reinforcement learning for licensed-assisted access of lte in the unlicensed spectrum. In *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1279–1284, March 2015.

[28] M. Simsek, M. Bennis, M. Debbah, and A. Czylwik. Rethinking offload: How to intelligently combine wifi and small cells? In *2013 IEEE International Conference on Communications (ICC)*, pages 5204–5208, June 2013.

[29] Biljana Bojovic, Nicola Baldo, Jaume Nin-Guerrero, and Paolo Dini. A supervised learning approach to cognitive access point selection. *GLOBECOM Workshops, Joint Workshop of SCPA 2011 and SaCoNAS 2011*, 2011.

[30] Tingting Sun, Wade Trappe, and Yanyong Zhang. Improved ap association management using machine learning. *SIGMOBILE Mob. Comput. Commun. Rev.*, 14(4):4–6, November 2010.

[31] Francesc Wilhelmi, Cristina Cano, Gergely Neu, Boris Bellalta, Anders Jonsson, and Sergio Barrachina-Muñoz. Collaborative spatial reuse in wireless networks via selfish multi-armed bandits. *CoRR*, abs/1710.11403, 2017.

[32] Task Group AX. Tgax simulation scenarios. Technical report, IEEE P802.11, 2015.

[33] Jeongkeun Lee, Wonho Kim, Sung-Ju Lee, Daehyung Jo, Jiho Ryu, Taekyoung Kwon, and Yanghee Choi. An experimental study on the capture effect in 802.11a networks. In *Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, WinTECH '07, pages 19–26, New York, NY, USA, 2007. ACM.

[34] Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 237–250, New York, NY, USA, 2006. ACM.

[35] Jaume Barcelo, Boris Bellalta, Cristina Cano, Anna Sfairopoulou, Miquel Oliver, and Kshitiz Verma. Towards a collision-free wlan: Dynamic parameter adjustment in csma/e2ca. *EURASIP Journal on Wireless Communications and Networking*, 2011(1):708617, Mar 2011.