Flexible Spectrum Assignment for Local Wireless Networks

THÈSE N^{O} 6736 (2015)

PRÉSENTÉE LE 25 SEPTEMBRE 2015

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 3

PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Julien HERZEN

acceptée sur proposition du jury:

Prof. J.-P. Hubaux, président du jury
Prof. P. Thiran, directeur de thèse
Prof. A. Banchs, rapporteur
Prof. G. Bianchi, rapporteur
Prof. B. Rimoldi, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2015

Acknowledgements

First, I would like to express my deep gratitude to my advisor, Patrick Thiran, for trusting me and giving me the opportunity to work in his lab. The satisfaction experienced during the Ph.D. years largely depends on the relationship between the student and the advisor, and I'm happy that in my case it was both fruitful and enjoyable. I would also like to thank him for letting me momentarily "deviate" my research efforts towards other topics, which did not have much in common with the content of my thesis (besides from being interesting).

I would also like to thank the members of my thesis committee, Jean-Pierre Hubaux, Giuseppe Bianchi, Albert Banchs and Bixio Rimoldi, for their constructive and detailed feedback. It was a true honor for me to present my work to them.

I am thankful to all my friends and colleagues at EPFL; Vincent, Mohamed, Mathias, Igor, Christina, Robin, Kévin, Lucas, Farid, Sébastien, Denisa, Adel, Sylviane, Nicolas, Ramin, Dan and all the others. Several of them have become close friends over the years, and I'm very happy to have spent this time with them. I had the privilege of collaborating with some very nice and bright people. I am thankful to Christina, Albert, Vincent, Adel, Ruben, Matthias, Nidhi, Henrik, Seva and Cédric (who is largely responsible for initiating me to research). I also thank Yves and Marc-André (without whom my testbed setup would not have been as efficient as it was) and the awesome LCA secretaries, Patricia, Danielle, Angela and Holly (who is directly responsible for many English mistakes that this thesis does *not* contain).

Of course, the Ph.D. years would not have been as nice as they were without my precious friends from the outside world: Valentin, Sylvain, Maximilien, Julien, Justine, Marc, Nathalie, Maxime, Jean, Tiffany, Vincent, Sophie, Luc, Mathieu, Ana, Max, Sachin, Olivier, Michel, Simon and several others. You were invaluable to have around!

I would like to express my deep gratitude to my family, in particular to Valérie, Guillaume and Eva, as well as to Nicolas and to my parents Claire and Michel, for giving me the opportunity to go to the university and for their unconditional love and support. It means a lot to me.

Finally, I would like to express my love to Mathilde. She is always by my side and is the best teammate I could have (of course, together with our cat Séraphin). I dedicate this thesis to her, for making my life better.

Abstract

In this dissertation, we consider the problem of assigning spectrum to wireless local-area networks (WLANs). In line with recent IEEE 802.11 amendments and newer hardware capabilities, we consider situations where wireless nodes have the ability to adapt not only their channel center-frequency but also their channel width. This capability brings an important additional degree of freedom, which adds more granularity and potentially enables more efficient spectrum assignments. However, it also comes with new challenges; when consuming a varying amount of spectrum, the nodes should not only seek to reduce interference, but they should also seek to efficiently fill the available spectrum. Furthermore, the performances obtained in practice are especially difficult to predict when nodes employ variable bandwidths.

We first propose an algorithm that acts in a decentralized way, with no communication among the neighboring access points (APs). Despite being decentralized, this algorithm is self-organizing and solves an explicit tradeoff between interference mitigation and efficient spectrum usage. In order for the APs to continuously adapt their spectrum to neighboring conditions while using only one network interface, this algorithm relies on a new kind of measurement, during which the APs monitor their surrounding networks for short durations. We implement this algorithm on a testbed and observe drastic performance gains compared to default spectrum assignments, or compared to efficient assignments using a fixed channel width.

Next, we propose a procedure to explicitly predict the performance achievable in practice, when nodes operate with arbitrary spectrum configurations, traffic intensities, transmit powers, etc. This problem is notoriously difficult, as it requires capturing several complex interactions that take place at the MAC and PHY layers. Rather than trying to find an explicit model acting at this level of generality, we explore a different point in the design space. Using a limited number of real-world measurements, we use supervised machine-learning techniques to learn implicit performance models. We observe that these models largely outperform other measurement-based models based on SINR, and that they perform well, even when they are used to predict performance in contexts very different from the context prevailing during the initial set of measurements used for learning.

We then build a second algorithm that uses the above-mentioned learned models to assign the spectrum. This algorithm is distributed and collaborative, meaning that neighboring APs have to exchange a limited amount of control traffic. It is also utility-

optimal – a feature enabled both by the presence of a model for predicting performance and the ability of APs to collaboratively take decisions. We implement this algorithm on a testbed, and we design a simple scheme that enables neighboring APs to discover themselves and to implement collaboration using their wired backbone network. We observe that it is possible to effectively gear the performance obtained in practice towards different objectives (in terms of efficiency and/or fairness), depending on the utility functions optimized by the nodes.

Finally, we study the problem of scheduling packets both in time and frequency domains. Such ways of scheduling packets have been made possible by recent progress in system design, which make it possible to dynamically tune and negotiate the spectrum band used by each frame. We observe that including frequency-domain decisions enables the scheduling algorithms to overcome important inefficiencies that are present in the time domain. In order to achieve these gains, we propose TF-CSMA/CA, a frequency-domain extension of the CSMA/CA backoff mechanism employed by IEEE 802.11. TF-CSMA/CA is completely decentralized, and it requires no explicit signaling, synchronization or spectrum scanning. Yet, we show that it exhibits desirable self-organizing properties in the spectral domain and achieves excellent performance and fairness.

Keywords: Spectrum allocation, resource allocation, wireless networks, algorithms, WLANs, performance modeling, self-organization, packet scheduling

Résumé

Dans cette thèse, nous étudions le problème d'attribution de spectre à des réseaux sans-fil locaux (WLANs). En accord avec les récentes normes IEEE 802.11 et les nouvelles possibilités matérielles, nous considérons des situations où les noeuds sans-fil ont la possibilité d'adapter non seulement leur fréquence centrale, mais également leur largeur de bande. Cette possibilité apporte un important degré de liberté, amène plus de granularité et permet d'attribuer le spectre de manière potentiellement plus efficace. Néanmoins, cela comporte également de nouveaux défis ; lorsque les noeuds peuvent utiliser une quantité variable de spectre, ils ne doivent plus seulement chercher à minimiser les interférences, mais également à utiliser le spectre existant de manière efficace. En outre, les performances obtenues en pratique deviennent particulièrement difficiles à prédire lorsque les noeuds utilisent une quantité de spectre variable.

Nous proposons tout d'abord un algorithme qui agit de manière décentralisée, sans communication entre points d'accès voisins. Bien qu'il soit décentralisé, cet algorithme permet aux noeuds d'organiser leur consommation de spectre, et il optimise un compromis explicite entre le besoin de minimiser l'interférence et celui d'utiliser efficacement le spectre. Afin que les points d'accès puissent continuellement adapter leurs bandes de fréquences en réponse aux conditions avoisinantes, tout en n'utilisant qu'une seule interface réseau, cet algorithme utilise un nouveau type de mesures, durant lesquelles les points d'accès observent les réseaux avoisinants pendant de courtes durées. Nous avons implémenté cet algorithme sur du matériel de test et avons observé une amélioration drastique des performances en comparaison aux attributions de spectre utilisées par défaut, ou en comparaison à d'autres techniques d'attribution de spectre opérant avec une largeur de bande fixe.

Nous proposons ensuite une procédure pour prédire la performance qui peut être obtenue en pratique, lorsque les noeuds opèrent avec des configurations arbitraires de spectre, charge de trafic, puissance de transmission, etc. Ce problème est notoirement difficile, car il requiert de prendre en compte plusieurs interactions complexes qui ont lieu aux couches d'accès (MAC) et physique (PHY). Plutôt que d'essayer de trouver un nouveau modèle explicite qui soit suffisamment général, nous explorons une approche différente. Nous utilisons un nombre limité de mesures provenant d'un vrai réseau, afin d'obtenir des modèles de performance implicites en utilisant des techniques d'apprentissage automatique. Nous observons que ces modèles se comportent beaucoup mieux que d'autres modèles basés sur le rapport du signal sur le bruit et l'interférence (SINR), et qu'ils

se comportent bien même lorsqu'ils sont utilisés pour prédire la performance dans des contextes très différents de ceux qui prévalaient lors de l'obtention des mesures initiales utilisées pour l'apprentissage.

Nous construisons ensuite un second algorithme qui utilise les modèles précédemment appris afin d'attribuer le spectre. Cet algorithme est distribué et collaboratif, ce qui signifie que les points d'accès voisins ont besoin d'échanger une quantité limitée de trafic de contrôle. Cet algorithme est également optimal, dans le sens où il maximise une somme de fonctions d'utilité. Cette propriété est rendue possible d'une part par l'existence d'un modèle qui permet de prédire les performances et d'autre part par le fait que les points d'accès peuvent prendre des décisions de manière collaborative. Nous avons implémenté cet algorithme sur du matériel de test, et nous avons mis au point une technique qui permet à des points d'accès voisins de se découvrir mutuellement et de mettre en oeuvre la collaboration en utilisant leur réseau filaire. Nous observons qu'il est possible de choisir le type de performance obtenue en pratique en fonction de différents objectifs (décrits en termes d'efficacité et/ou d'équité), qui dépendent des fonctions d'utilité utilisées par les noeuds.

Finalement, nous étudions le problème qui consiste à ordonnancer les paquets dans les domaines temporel et fréquentiel. Cette façon d'ordonnancer les paquets a été récemment rendue possible par la mise au point de nouveaux systèmes qui permettent d'adapter et de négocier la bande de fréquences utilisée par chaque frame. Nous observons qu'inclure des décisions portant sur le domaine fréquentiel permet aux algorithmes d'ordonnancement de surmonter d'importantes inefficacités présentes dans le domaine temporel. Afin d'obtenir ces gains, nous proposons TF-CSMA/CA, qui est une extension dans le domaine fréquentiel de la procédure d'évitement des collisions utilisée par IEEE 802.11. TF-CSMA/CA est complètement décentralisé et ne requiert aucun trafic de contrôle, aucune procédure de synchronisation et aucune analyse de spectre. Nous montrons que TF-CSMA/CA permet aux noeuds de s'auto-organiser dans le domaine fréquentiel et qu'il fournit d'excellentes caractéristiques de performance et d'équité.

Mots clefs : Attribution de spectre, attribution de resources, réseaux sans-fil, algorithmes, WLANs, modèles de performance, auto-organisation, ordonnancement de paquets.

Contents

1 Introduction

In recent years, Wi-Fi networks have been massively adopted worldwide. These networks represent an inexpensive and convenient way to provide high-speed connectivity to end-users in homes, enterprises and commercial hotspots. Recent IEEE 802.11 amendments, such as 802.11n and 802.11ac, are capable of providing up to several Gbps of raw transmission speeds, which posits Wi-Fi as an attractive option to replace wired networks, even in the presence of relatively heavy traffic demands. As of 2015, more than two billion Wi-Fi devices have been sold annually, and 25% of all households are equipped with a Wi-Fi network [wi-14][1].

This adoption has been so ubiquitous that it is now common in urban environments to detect several dozens of operating Wi-Fi access points from any given location. In fact, it is even possible to use databases of such lists of visible access points in order to build accurate population-density maps or geo-localization services. In Figure 1.1, we show a density map of the access points on the East Coast of the United States and in Figure 1.2, we show a map of some access points in Manhattan. It is clear that the density of Wi-Fi

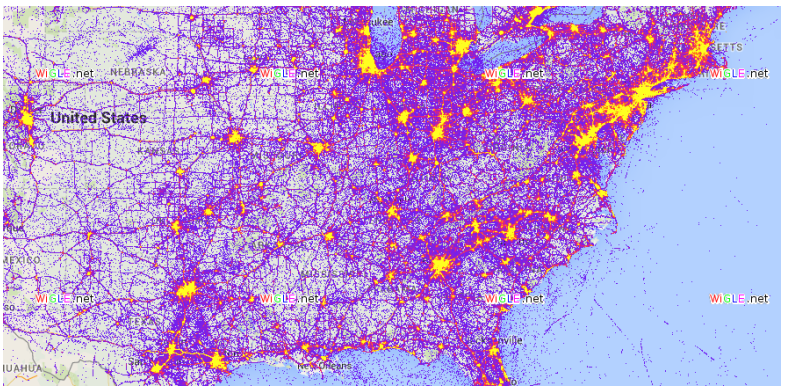

Figure 1.1 – Density map of access points detected on the East Coast of the United States by the community of users of the Wigle.net project. Source: `www.wigle.net`.

[1] The fraction is about 50% to 80% of households in developed countries [wia].

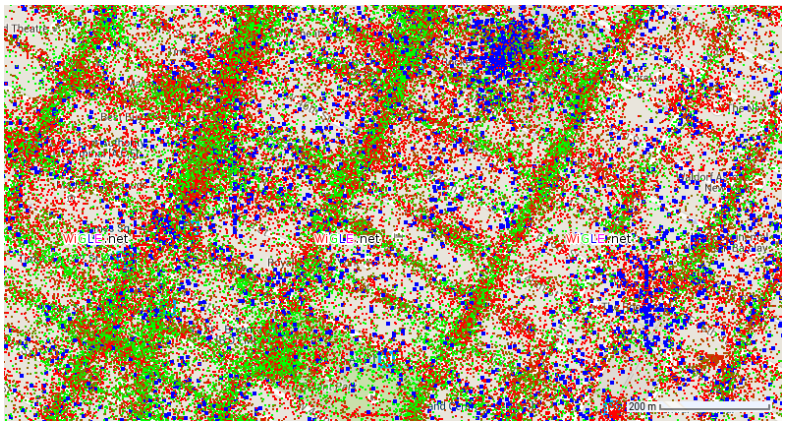

Figure 1.2 – Map of some access points detected in Manhattan. Each dot represents the approximate localization of an access point. Source: `www.wigle.net`.

deployments varies greatly over space (following that of human settlements), and in urban areas such as Manhattan, Wi-Fi deployments have become extremely dense.

Wi-Fi networks are successful under many aspects, but their success also causes them to experience a tragedy of the commons. Contrary to wired networks, wireless devices are susceptible to produce interference with each other if they operate too closely. In general, interference appears if some neighboring Wi-Fi devices transmit at the same time and on the same frequency; these situations can cause packet losses and reduce the effective capacity achievable on the corresponding Wi-Fi links. This is an important problem in practice, because the ongoing success of Wi-Fi often results in a high average number of devices that are physically close enough in order to interfere with each other (at least in urban environments such as Manhattan).

In order to mitigate interference, to maintain a high effective capacity and to exploit as efficiently as possible the wireless medium, neighboring devices need to separate their transmissions in time and/or frequency domains (i.e., avoid transmitting at the same time on the same frequency). To this end, 802.11 employs a time-domain mechanism based on CSMA/CA, whereby the stations "listen" on the wireless medium before attempting a transmission. This mechanism has been an important part of the success of Wi-Fi; it is completely decentralized and performs well in practice. Ideally, however, this time-domain operation should also be complemented by some efficient way of deciding on which spectrum band the networks should operate. Separating transmissions also in the frequency domain (rather than in the time domain only) is desirable for four main reasons. First, it is clear that Wi-Fi networks benefit from some level of organization in the spectral domain, in order to efficiently exploit the available spectrum and avoid that some frequency bands remain vacant (or lightly used) when the overall load is high. Second, we will see that separating transmissions in frequency domain drastically reduces a number of significant time-domain overheads of the MAC layer. These overheads are due

in part to the CSMA/CA mechanism and they have been exacerbated by recent 802.11 physical layers (this point will be made precise in Chapter 5). Third, we will also see that adapting the *amount* of consumed spectrum (i.e., the channel width) is an efficient way to achieve load balancing among several Wi-Fi links contending for access. Finally, the recent 802.11n and 802.11ac amendments have the ability to use large channel widths (up to 40 MHz for 802.11n and up to 160 MHz for 802.11ac – in contrast to 20 MHz for older 802.11 amendments). These large bandwidths, along with the use of aggressive modulation schemes and sophisticated signal processing techniques such as multi-user MIMO (MU-MIMO), enable these amendments to obtain very high transmission rates of several Gbps at the physical layer. However, using large bandwidths creates more interference, and these recent amendments therefore make it increasingly important to efficiently assign the available spectrum bands[2].

Although organizing transmissions in the frequency domain has the potential to greatly improve performance, current Wi-Fi networks are usually far from using the spectrum efficiently. The 802.11 standard does not specify how to use the frequency domain to improve performance, and typical off-the-shelf access points often use a fixed and arbitrary channel, or at best provide some "auto-channel" functionality, whereby they scan the available spectrum (usually at boot time) and select the channel with the smallest number of detected concurrent networks. This approach to spectrum assignment is mostly *monolithic*, in the sense that it does not modulate the amount of consumed spectrum (but selects only fixed-width channels), and it chooses an operating channel once and for all (or very infrequently).

The main goal of this thesis is to propose new spectrum-assignment algorithms that depart from this monolithic approach. The proposed algorithms are *flexible* both in terms of spectrum consumption – they do not only choose the center-frequencies, but also decide *how much* spectrum should be used by each device –, and in terms of dynamics – they continuously change the spectrum assignments in order to adapt to varying network conditions. The design of such algorithms is challenging for several reasons. When adapting the channel bandwidth (instead of the center-frequency only), the algorithms need to balance two conflicting goals. On the one hand, they need to minimize interference, which pushes them to reduce their channel width. On the other hand, they should try to occupy as much spectrum as possible (for efficiency), which incites them to increase their channel width. It turns out that this conflict makes the problem of flexible spectrum allocation significantly different from the more "classic" channel-assignment problem, where only the center frequency needs to be chosen and which can, for instance, be solved by variations of graph-coloring algorithms. In fact, how to balance these two goals is not clear *a priori* as, in addition, an efficient solution should depend on the number of

[2]In fact, recent physical layers reach transmission speeds that are close to information-theoretic upper bounds. As a result, it currently appears difficult to keep increasing raw transmission speeds, which makes the design of efficient spectrum-assignment schemes an attractive avenue to keep improving the performance of 802.11 networks.

interfering devices and their respective traffic loads. An additional major challenge comes from the fact that Wi-Fi devices are typically deployed in a chaotic and disorganized fashion by different individuals and administrative entities. These devices often do not have access to a complete view of the network, and there is no controller with a centralized knowledge that can make spectrum allocation decisions for them. Therefore, even though flexible algorithms should tend to require more knowledge and continuous updates (e.g., about the traffic loads of surrounding networks), we target only distributed algorithms that do not require centralized computation.

In the remainder of this introductory chapter, we first present more precisely the spectrum allocation problem. We then provide an outline of the dissertation and briefly mention the main contribution of each chapter. Finally, we compare our different approaches and describe the different points in the design space explored in each of the following chapters.

1.1 The Spectrum-Allocation Problem

Wi-Fi devices have the ability to operate using different channels, which correspond to different chunks of the total available spectrum. Currently, the total available spectrum usually corresponds to one or two of the unlicensed bands, which are the 2.4 GHz ISM and the 5 GHz U-NII bands. Future Wi-Fi networks might be able to operate in *white spaces*, which consist essentially in UHF TV bands that are typically licensed, but might be left vacant (and available for utilization) by their incumbents (see e.g., [NIK12]). The main challenge for exploiting white spaces consists in designing systems that are able to use the spectrum in an opportunistic manner, in order to avoid interfering with incumbents (which are essentially TV transmitters and wireless microphones). A substantial amount of research has been dedicated to this problem and, in particular, avoiding interference with incumbents has been the main focus of several years of research on cognitive radios (see e.g., [WL11] for a survey). In this dissertation, we do not study this problem, and we do not attempt to determine the boundaries of the total available spectrum. Instead, we assume that these boundaries are known and determined by an exogenous process. In practice, this means that the total available spectrum is fixed for networks that operate using only unlicensed bands, and it could be variable for future wireless networks that also exploit white spaces. This is mainly a technicality, and in this thesis we focus on finding efficient algorithms for organizing the spectrum usage of secondary users, assuming that the total available spectrum is fixed. We note, however, that all our algorithms can be easily extended to the case where the total available spectrum varies over time (even if the nodes do not have a consistent view of the boundaries of the spectral bands).

The channels are determined by their center-frequency and their bandwidth, which we also call *channel width*. For the older 802.11b/g/a standards, the channel width is fixed and set to 20 MHz. In Figure 1.3, we represent the available 20 MHz channels in

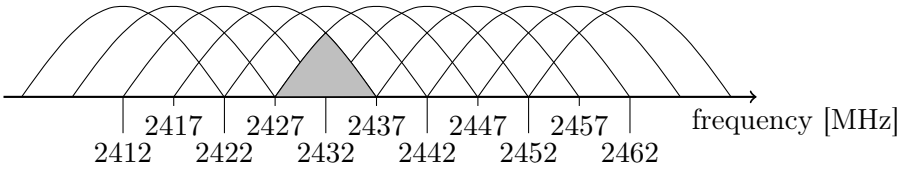

Figure 1.3 – Schematic view of the 11 IEEE 802.11 fixed-width channels in the 2.4 GHz ISM band. The inter-channel separation is 5 MHz and each channel is 20 MHz wide.

the 2.4 GHz band. In this band, the separation between the center frequencies of each channel is only 5 MHz, and so some channels overlap with each other, as shown by a gray area in the figure. The 5 GHz band provides several channels between 5.17 GHz and 5.825 GHz[3]. In contrast to the 2.4 GHz band, all the channels in the 5 GHz band are separated by at least 20 MHz, so that channels using a width of no more than 20 MHz do not overlap.

The newer 802.11n and 802.11ac standards can use a variable channel width. In practice, the different widths are obtained by bonding several sub-channels together. 802.11n can operate on the 2.4 or 5 GHz bands, and use the (legacy) 20 MHz bandwidth, as well as a 40 MHz bandwidth, which is obtained by bonding two 20 MHz channels together. 802.11ac operates only in the 5 GHz band, and it can use widths of 20 MHz, 40 MHz (2×20 MHz), 80 MHz (2×40 MHz) and 160 MHz (2×80 MHz). Note that 160 MHz is large compared to the spectrum available in the 5 GHz band. In fact, in most of Europe and North America, there is currently only one contiguous 160 MHz band available in the 5.17-5.33 GHz range. An immediate consequence is that two neighboring networks currently operating on a contiguous band of 160 MHz must share the same spectrum. In addition to the variable bandwidth capabilities of 802.11n and 802.11ac, let us mention that it is possible to configure some 802.11g/a wireless cards to use non-legacy bandwidths of 5, 10 and 40 MHz (in addition to 20 MHz – see [CMM+08] and the Appendix). We will use such configurations to test the algorithm presented in Chapter 2.

1.1.1 Interference versus Capacity

Let us now explain how spectrum assignment affects both interference and capacity, and how these two metrics can be used to build two qualitative optimization objectives that will guide our design of spectrum assignment strategies.

Interference can occur when two 802.11 transmitters use overlapping parts of the spectrum at the same time. In this case, one (or both) of the receivers might be unable to filter out the signal produced by the other (unintended) transmitter. If the interfering

[3]These channels are spread over non-contiguous bands and the exact number of available channels depends on local regulations. In Switzerland, there is a total of 23 different 20 MHz channels in the 5 GHz band at the time of writing.

signal is strong enough, it might prevent the receiver from correctly demodulating some of the symbols. In turn, if too many symbols are lost (more than what the error correcting code can handle), there is a collision and the frame is lost. It has been empirically shown in [MSBA06] that the actual amount of interference (measured in terms of throughput) depends on the amount of spectrum overlap. This is intuitive, as the amount of overlap determines the amount of interfering power and the likelihood of corrupting symbols. Therefore, everything else being equal, partial spectrum overlap is more desirable than full overlap. In practice, however, it is difficult to know *a priori* what configuration performs best. In particular, partial spectrum overlap can still result in situations where the amount of received power is high enough to trigger carrier-sensing at neighboring transmitters, and thus where CSMA/CA shares the transmission opportunities in the time domain exactly as if the the concurrent transmitters were operating on the same band with full overlap. Yet, in general, although performance for various overlap configurations is difficult to predict, it benefits from the minimization – or, to an even larger extent, the complete avoidance – of spectrum overlap[4]. Based on these considerations, we can state the first qualitative objective of spectrum assignment.

First objective of spectrum assignment (interference avoidance):
Neighboring networks should use configurations that minimize spectrum overlap.

This first objective is intuitive, and it is the primary optimization objective followed by all "classic" channel-allocation strategies (see e.g., [MBB+06, MSA+06, AJSS05, LCBM12, KBC+07]) as well as, to some extent, by the "auto-channel" mechanisms of many off-the-shelf access points. However, when the devices have the ability to select their channel width (and not only their channel center-frequency), this first objective cannot be considered in isolation. In this case, it is necessary to consider the potential capacity gains offered by various spectrum configurations.

In general, for networks that are not subject to interference, the effective capacity (i.e., the achievable throughput) increases with the channel width. For 802.11n and 802.11ac, which use physical layers based on OFDM, this is because the subcarriers consume a fixed bandwidth, and hence the total number of subcarriers increases with the total bandwidth. More generally, the increase of capacity with bandwidth is consistent with Shannon's formula [Sha48]; if we write B for the bandwidth, then the information-theoretic capacity

[4] In practice, the lowpass filters used to shape the baseband signals at the transmitters are not perfect, which results in non-ideal spectrum masks (the channels in Figure 1.3 are represented in a schematic way). Usually, this means that there is some amount of power leaking onto the adjacent channels (i.e., the channels beyond the nominal bandwidth). This phenomenon is known as *adjacent-channel interference*. The 802.11 standard [IEE12] imposes constraints on the effective spectrum masks, so that the attenuation is relatively strong (at least about 20dB) outside of the channel boundaries (see Figure A.2 in the Appendix for an example), and adjacent channel interference is limited. Nevertheless, it is helpful to know that adjacent channels can overlap and create some interference.

C obtained on a link subject to white-noise interference is given (in bps) by

$$C = B \log_2 \left(1 + \frac{P_{\text{rx}}}{N_0}\right), \tag{1.1}$$

where P_{rx} is the power received from the transmitter and N_0 is the noise power (hence P_{rx}/N_0 denotes the SNR at the receiver). Therefore, in regimes where the SNR is approximately constant with the bandwidth, the information-theoretic capacity increases approximately linearly with the bandwidth. To be more precise, however, we should note that the effective capacity grows in fact sub-linearly with the bandwidth for two reasons [CMM+08]. First, for a fixed transmit power, small bandwidths pack more power per unit-Hertz than larger bandwidths, which increases the SNR. Equivalently, this can be seen as an effect of the fact that N_0 is increasing with B in expression (1.1). Second, for 802.11 devices, the time-domain overheads due to the backoff mechanism (along with other time-domain overheads – see Chapter 5) become proportionally more important when the physical rates increase, which penalizes the efficiency of 802.11 networks operating with larger bandwidths. Nevertheless, for devices that are not subject to interference and that have a reasonably high SNR, it is beneficial for performance to use a larger channel width [CMM+08]. We can thus state the second qualitative objective of spectrum assignment.

Second objective of spectrum assignment (capacity maximization):
Wi-Fi devices should, in general, use as much spectrum as possible.

It is clear that the two above objectives are in contradiction with each other. The first objective is easier to satisfy if the stations use small channel widths, but such allocations do not satisfy the second objective. Throughout this thesis, we propose different ways to balance this "interference versus capacity" tradeoff. In Chapter 2, we propose an algorithm that optimizes an objective function that is an explicit formulation of this tradeoff. In Chapters 3 and 4, we propose a way to model the intricate performance patterns of interfering networks, in order to find *utility-optimal* spectrum allocations, which optimally balance this tradeoff. Finally, in Chapter 5, we again consider this tradeoff explicitly, but at the packet-scheduling layer.

1.1.2 Flexible Spectrum Allocation

Because interference between 802.11 networks is caused by packet transmissions, it is clear that the proportion of time during which a given network interferes with (or experiences interference from) its neighbors depends on the traffic loads on each of these networks. Therefore, any ideal balance between the two qualitative objectives stated in the previous section should be a function of the current traffic conditions of neighboring networks. For example, it is clear that a network that is surrounded only by inactive neighbors can use

as much spectrum as desired, in which case interference avoidance does not matter and only the second objective (of capacity maximization) is relevant. Obviously, the same consideration also holds for isolated networks with no neighbors. In contrast, for networks operating with active neighbors (e.g., in dense urban environments), the first objective (of interference avoidance) needs to be given more weight. In fact, in this case, the second objective (of capacity maximization) is less likely to be useful as an optimization objective, as in dense environments the spectrum is likely to be crowded, and therefore its usage often does not need to be increased.

Notably, traffic loads, client associations and activity levels vary over time. Networks can have several clients who join and leave (typically at timescales of minutes to hours), and each of these clients might be idle or produce different traffic patterns (varying at timescales as low as tens or hundreds of milliseconds). Therefore, a desirable feature (and a design goal) of spectrum-assignment strategies is the ability to continuously re-evaluate and adapt spectrum consumption over time. In particular, in this thesis, we do not target algorithms that "freeze" spectrum assignments once and for all after some convergence criterion has been met. Instead, all the proposed algorithms provide some convergence guarantees when the traffic is in steady-state conditions, but these guarantees describe the average *fraction of time* spent in desirable configurations, and the algorithms always continuously adapt their spectrum usage. In the following, we refer to the *flexibility* of spectrum assignment schemes as (i) the ability to adapt the *amount* of spectrum consumed by each network, and (ii) the fact that spectrum allocations are *continuously adapted* over time (in reaction to surrounding conditions).

1.2 Outline and Contributions

We give an overview of the outline of the thesis and summarize the main contribution of each chapter.

- **Chapter 2:** We propose SAW (Spectrum Assignment for WLANs), an algorithm for the joint allocation of center frequencies and bandwidths of 802.11 networks. SAW is tailored for home networks; it is completely decentralized and produces no control traffic. Yet, we show that it converges to configurations that solve a network-wide explicit version of the interference-versus-capacity tradeoff presented in Section 1.1.1. Notably, we observe that it is possible to use a fixed optimization objective to find efficient solutions to the interference/capacity tradeoff, irrespective of the network spatial density. SAW constantly evaluates the spectrum used by each AP at random time intervals (every few minutes on average). We implement it on a 802.11 testbed (the details of which will be presented in the appendix).
  Main contribution: To the best of our knowledge, SAW is the first algorithm for the joint allocation of center frequency and bandwidth, which is decentralized and requires no control traffic.

- **Chapter 3:** We take a step back from the pure spectrum-assignment problem, and consider the task of predicting the performance achievable by interfering networks when operating with arbitrary traffic loads, channel qualities, spectrum configurations and transmit powers. Predicting performance in such arbitrary situations is challenging due to complex interactions at the MAC and PHY layers. We show that, using an initial measurement campaign performed in a controlled fashion on real networks, it is possible to use supervised machine-learning techniques to build implicit performance models. When trained properly, these models can capture many complex dependencies and outperform existing wireless models (e.g., SINR-based models) that are also seeded with measurements. Importantly, we observe that these models still perform reasonably well when predicting performance in conditions that are drastically different from those that prevail during the initial measurement campaign.
  Main contribution: We show that, as far as performance prediction is concerned, models coming from the machine-learning domain can outperform specialized wireless models fitted to measurements.

- **Chapter 4:** We formulate a utility-optimal algorithm for the joint allocation of center frequency, bandwidth and transmit power. Converging to utility-optimal configurations requires a sophisticated performance model in order to carefully select configurations. For this reason, we take the approach proposed in Chapter 3 and use models obtained using machine learning for predicting the performance achievable in the various configurations. Contrary to the algorithm of Chapter 2, the algorithm of Chapter 4 is distributed, but not fully decentralized. It relies on the exchange of control messages among immediate neighbors. These messages are required for the algorithm to be utility-optimal; they are used by the APs to predict performance and quantify the effect (in terms of utility) of each configuration on neighboring networks. Similarly to the algorithm of Chapter 2, this algorithm re-evaluates spectral configurations at random intervals (also every few minutes on average). We implement it on our testbed and show that it is possible to find spectral and transmit-power configurations that meet precise optimization objectives (in terms of performance and fairness), as determined by utility functions.
  Main contribution: We propose the first utility-optimal algorithm for the joint allocation of center frequency, bandwidth and transmit power. To the best of our knowledge, we provide the first observation that utility-driven optimization of spectrum usage can be employed in practice, on a real network, to explicitly tune a balance between throughput and fairness.

- **Chapter 5:** We propose TF-CSMA/CA, an extension of the backoff mechanism of 802.11 to the frequency domain. With TF-CSMA/CA, when a station is involved in a collision, it performs backoff in both time and frequency domains. Backing off also in the frequency domain enables the transmitters to amortize the severe overheads created by recent 802.11 PHY layers, and to be much more aggressive in

time domain. TF-CSMA/CA is an algorithm targeting new systems that implement *flexible channelization*, whereby wireless stations adapt their spectrum on a per-frame basis. Contrary to existing channelization schemes, TF-CSMA/CA is entirely decentralized and only reacts to packet collisions, successful transmissions and carrier sensing. Although it uses only transmission outcomes as implicit signals, we show that TF-CSMA/CA provides self-organization in the spectral domain, and that interfering links spend the majority of the time in non-interfering bands. Overall, relying only on transmission outcomes provides a simple and effective way to assign spectrum to stations directly at the MAC layer, in a way that departs from the usual "reservation-based" spectrum usage, but that is instead determined by instantaneous traffic loads, just like CSMA/CA in the time domain. We implement TF-CSMA/CA in a packet-level simulator and observe drastic gains compared to perfect time-domain scheduling (i.e., compared to perfect TDMA schedules), and performance similar to what can be obtained by 802.11 operating with optimal (but monolithic) spectrum allocation.

Main contribution: To the best of our knowledge, TF-CSMA/CA is the first random-access mechanism performing backoff in both time and frequency domains. It is the first scheduling algorithm for flexible channelization that does not require any form of explicit signaling, synchronization, spectrum scanning or central control.

1.3 Comparison of Proposed Spectrum-Assignment Strategies

There is no single best way of designing a spectrum-allocation scheme, and the algorithms presented in this thesis explore different points in the space of possible tradeoffs. In this section, we divide the design space into four aspects that we think are important: decentralization (i.e., the ability of the algorithms to act in a distributed way), flexibility (i.e., the ability to adapt the amount of consumed spectrum and react on short timescales), practicality with today's hardware, and strength of the optimality guarantees. We illustrate the position occupied by each algorithm in a schematic way in Figure 1.4 (where the strength of the optimality guarantees is represented by a level of gray), and in the remainder of this section, we explain the corresponding design decisions.

1.3.1 Decentralization

In order to find efficient spectrum assignments, each access point needs a way to frequently learn about the state and condition of its neighbors before taking a potential decision for itself. In this thesis, we propose three different ways of achieving this goal[5]. In

[5]We always assume that the wireless stations use only one wireless interface. Using an additional interface (e.g., dedicated for control traffic) could help spectrum-assignment schemes to acquire information about neighboring networks, but most off-the-shelf devices have only one interface. Furthermore, having

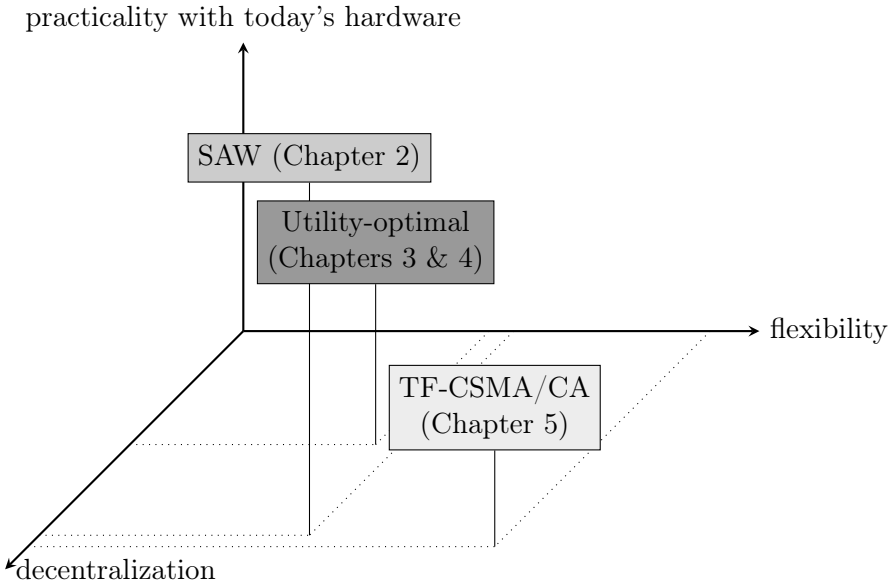

Figure 1.4 – Representation of the different points in the design space explored by each of the proposed spectrum assignment strategies. The strength of the optimality guarantees is represented by a level of gray.

Chapter 2, we propose and implement a *micro-sensing* technique, whereby the access points spend short durations out of their operating bands. The durations are kept short in order not to interrupt traffic with the access point's clients (who do not switch and still use the main operating band), but they are sufficiently long in order to estimate the activity levels of neighboring networks on different spectrum bands and take efficient spectrum decisions without requiring any form of control traffic. In Chapter 4, we allow some communication between neighboring access points to exchange control information. Such control information enables the access points to learn the exact configuration of each of their neighbors and to predict the capacity achievable on each of their own links in each possible configuration. This knowledge enables them to weigh the effect that each spectrum configuration could have on each neighbor. In fact, although this form of collaboration introduces some extra complexity, it enables us to implement richer optimization criteria and design an algorithm where the local decisions taken by each access point maximize a network-wide sum of arbitrary utility functions. In order to make it practical, we propose and implement a neighbor-discovery mechanism, whereby the access points periodically broadcast their public IP address, and subsequently use their wired backbone connection for exchanging control traffic with their neighbors without creating overhead on the wireless medium. Finally, with the algorithm proposed in Chapter 5, the spectrum decisions are taken on a per-frame basis by each transmitter. In this case, the stations learn implicitly about their neighbors through the outcomes

access to an additional interface changes the nature of the problem (as it could, in some cases, increase the overall achievable capacity if the additional interface is itself used for data transmission).

of packet transmissions (i.e., the successes and collisions), as well as carrier sensing in their current operating band. As a result, this last scheme is the "most decentralized" of all, as the stations do not even use any explicit knowledge of the spectrum used by their neighbors. This is in contrast to the algorithm of Chapter 2, which is decentralized (in the sense that it requires no control traffic), but where the access points need to acquire some explicit knowledge of the spectrum consumed by their neighbors. Overall, the fully decentralized settings of Chapters 2 and 5 are suitable for chaotic network deployments by different administrative entities, whereas the collaborative utility-optimal setting of Chapter 4 is suitable for enterprise networks.

1.3.2 Flexibility

All of the algorithms presented in this thesis adapt dynamically the amount of spectrum consumed by each station, and all of them respond to the interference-versus-capacity tradeoff described earlier. Despite this similarity, the algorithms differ on the timescales at which they operate. With the algorithms of Chapters 2 and 4, each access point is in charge of choosing the spectral configuration for its own network. In this setting, changing the operating band incurs some overhead; all the clients have to be notified, and the access point has to learn about the state of neighboring networks. For this reason, with these algorithms, every access point re-evaluates its configuration at random time intervals, every few minutes on average. In contrast, with the algorithm proposed in Chapter 5, acquiring information about neighboring configurations is inexpensive (it comes implicitly from transmission outcomes and carrier sensing), and the spectrum can be adapted for every frame. As a result, this algorithm operates at a much faster timescale that is determined by the speed at which packets are scheduled (and is typically of the order of the millisecond). In this sense, this algorithm is the most flexible of all as it can nearly instantly adapt the spectrum consumption to varying network conditions.

1.3.3 Practicality

In order for communication to take place, the transmitter and the receiver have to know the frequency band used for transmission. With the algorithms of Chapters 2 and 4, the access points use dedicated signaling in order to notify the clients when the operating spectrum band changes[6]. In this sense, these algorithms are practical and implementable with today's hardware; we are able to evaluate them using testbed experiments. In contrast, with the algorithm of Chapter 5, the transmitter has to indicate the spectrum band used by each frame in its physical layer preamble. This has been recently shown to be feasible on real hardware (see e.g., [CLL+12, YKQ13]), but more time is required until such functionalities become supported by off-the-shelf hardware. In this thesis,

[6]Such signaling is required by any spectrum-assignment mechanism in one form or another. In practice, it could be integrated within addendum of the IEEE standards (e.g., IEEE 802.11h).

we do not study this practical aspect, but rather focus on the algorithmic challenge of assigning spectrum at the MAC layer. Due to this constraint, we evaluate the algorithm of Chapter 5 by using packet-level simulations.

1.3.4 Optimality

Finding optimal spectrum assignments requires solving a complex optimization problem. The number of possible configurations is combinatorial and, in the simplest case where there is only one channel width and where the channel and traffic conditions are homogeneous across all network nodes, optimizing spectrum assignment solves an NP-complete graph-coloring problem. Due to this complexity, we resort to iterative search techniques in order to design practical algorithms. The challenge, therefore, becomes to design schemes that provide some *self-organization* properties, in the sense that the iterative search procedures converge towards optimal solutions, even though the access points take local decisions in a decentralized fashion. In this context, the algorithms proposed in this thesis differ in the strength and generality of the target that they optimize. The algorithm of Chapter 2 converges to configurations that are arbitrarily close to the optimal solution of a cost minimization problem, where the cost function explicitly accounts for both interference and capacity. In contrast, the algorithm of Chapter 4 provides the most general optimality guarantees; it converges arbitrarily close to configurations that maximize the sum of arbitrary *utility functions* of the achieved throughput. This setting enables the algorithm to find configurations that optimize any arbitrary balance between, for instance, performance and fairness. Finally, contrary to the first two algorithms, the algorithm of Chapter 5 does not provide network-wide convergence guarantees. However, we show that it is also self-organizing, in the sense that neighboring stations (belonging to the same contention domain) spend an arbitrarily large fraction of their time in states without interference (when such states exist).

2 Decentralized Spectrum Assignment for WLANs

2.1 Introduction

In this chapter, we present SAW (Spectrum Assignment for WLANs), our first algorithm for spectrum assignment. SAW takes a direct approach for finding efficient solutions to the "interference versus capacity" tradeoff presented in Section 1.1.1. In particular, SAW is a decentralized algorithm that conciliates an efficient *global packing* of spectrum chunks (which corresponds to minimizing network-wide interference) with the *local benefits* of using appropriate bandwidths (which typically corresponds to maximizing the spectrum usage of each WLAN).

SAW is a practical algorithm for online and distributed center-frequency and bandwidth assignment, which targets *home networks*, i.e., residential WLANs with access points (APs) deployed in a chaotic fashion by individuals or different administrative entities. It runs at the AP of a WLAN and relies exclusively on inter-neighbor measurements, without generating additional traffic. Furthermore, it is transparent and operates while the network is up and running. Despite these characteristics, it is self-organizing and provably converges towards optimal spectrum allocations, in a sense that we will define later. Finally, it improves the performance, even when only a subset of the interfering WLANs use it (therefore providing incentives for incremental deployment), or when some access points behave selfishly.

We organize the remainder of this chapter as follows. We start by describing some notation and a model for interacting WLANs in Section 2.2. We then describe the operation of SAW, as well as its convergence properties in Section 2.3. Next, we evaluate the performance of the algorithm on large ecosystems of interfering WLANs by using simulations in Section 2.4. We complement this performance evaluation by testbed experiments in Section 2.5. We then discuss relevant related work in Section 2.6 and, finally, we close the chapter with a summary in Section 2.7.

2.2 A Model for WLAN Interactions

As explained in the introduction, the spectrum occupied by neighboring WLANs affect performance through interference. Therefore, in order to quantify this performance effect, SAW uses an interference metric. In order to formally define this metric, we need to define what composes neighborhood relationships among WLANs. This is achieved in two steps; we first build a link-centric model in Section 2.2.1, and then extend it to the specifics of WLANs, with APs and clients, in Section 2.2.2.

2.2.1 Link-Centric Model and Neighborhood System

Let $\mathcal{L}$ define a set of links, $\mathcal{F}$ a finite set of center frequencies and $\mathcal{B}$ a finite set of channel bandwidths. In the following, we use the term *band* to denote a particular combination of channel frequency and bandwidth. Each link $l \in \mathcal{L}$ comprises two nodes, acting as a transmitter or receiver for this link. As we assume that each node uses only one wireless interface, the transmitter and receiver of a given link must use the same band in order to communicate[1]. For a link l, $f_l \in \mathcal{F}$ denotes the center-frequency used by link l and b_l denotes the bandwidth used by link l. Finally, for a link l, $\mu_l \in [0, 1]$ denotes the average fraction of time during which a node occupies the medium (which we also call the *airtime ratio of* l). In practice, μ_l depends on the 802.11 CSMA/CA time-sharing mechanism, the physical rates used on link l, and b_l (because transmission delays are inversely proportional to the channel width).

For any pair of links l, k, we say that l and k are mutual *neighbors* (and interfere) if there exists a configuration (f_l, b_l, f_k, b_k) such that two of the four nodes composing l and k belong to different links and receive frames from each other. Then, we write $\mathcal{N}_l$ for the set of neighbors of link l. By construction, the neighborhood relationship is symmetric, i.e., $k \in \mathcal{N}_l \Leftrightarrow l \in \mathcal{N}_k$. Note that this model does not imply symmetric interference levels: as specified later in Section 2.2.3, two neighbors can mutually interfere with a different extent (e.g., if they are subject to different traffic loads). With this model, a link is considered as a neighbor if its transmitter is in the communication range of any node of another link. In this sense, it captures both exposed and hidden terminal situations. However, it does not capture interferers that are not within communication range, as it relies on the ability of the interferers to decode each other's frames. Nevertheless, we use this model as it is the one that fits best the operation of SAW in practice (in our implementation, the APs need to decode their neighbors' frames in order to detect them). Note that detecting interferers outside the communication radius in a distributed setting is an interesting problem on its own (see e.g. [MRWZ06])[2].

[1] Note that a receiver could decode a signal sent with a narrower width, but this would require special non-commodity hardware. In this Chapter, we target an algorithm that runs on off-the-shelf hardware.

[2] The algorithm proposed in Chapter 5 also goes in this direction, as it can react without decoding any neighboring frames

2.2.2 From Link to WLANs: BSS-Centric Model

We now tailor our model to co-existing and possibly interfering BSSs[3]. A BSS is a set of nodes, where one node is an AP and those remaining are clients. Compared to isolated links, the main difference is that all traffic goes either to or from the AP. Therefore, all nodes of a BSS must use the same band. In this chapter, we assume that the AP is in charge of choosing the band for its BSS.

To account for this new constraint, we extend the neighborhood system defined for links, and we define a neighborhood system for BSSs. Let $\mathcal{A}$ be a set of N BSSs. For a BSS $A \in \mathcal{A}$, a link l belongs to A (and we write $l \in A$) if both nodes of l belong to A. In this case, one node of l is of course the AP of A. Then, two BSSs A and B are *neighbors* if there exist two links $l \in A$ and $k \in B$ such that $l \in \mathcal{N}_k$. If A and B are neighbors, we write $A \in \mathcal{N}_B$. The symmetry of the link neighborhoods implies $A \in \mathcal{N}_B \Leftrightarrow B \in \mathcal{N}_A$. In addition, we write $f_A \in \mathcal{F}$ and $b_A \in \mathcal{B}$ for the center frequency and channel bandwidth used by the BSS A, respectively. We denote by $\mathbf{F} \in \mathcal{F}^N$ and $\mathbf{B} \in \mathcal{B}^N$ the center frequencies and the bandwidths used by the N BSSs, respectively.

2.2.3 Interference Metric

We now define the metric used by SAW to quantify the amount of interference between any two links. For two links l and k, let $I_l(k)$ denote the *link-interference* created by k on l. In addition, let $IF(l,k)$ denote the *interference factor* (see [MSBA06]). This factor describes the amount of overlap between the two spectrum masks used on links l and k. We can now define $I_l(k)$ as

$$I_l(k) := \begin{cases} \mu_k \cdot IF(l,k) & \text{if } k \in \mathcal{N}_l, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

with

$$IF(l,k) = \int_{-\infty}^{+\infty} S_k(f) S_l(f - |f_l - f_k|) df,$$

where S_k is the transmit mask of link k, and S_l is the mask used on link l. The 802.11 standard defines the characteristics of masks [IEE12]. As already mentioned in introduction, they change with channel bandwidth: for a given transmit power, the emitted power per unit Hertz increases as the channel bandwidth decreases (see [CMM+08] for a detailed explanation). Note that $I_l(k)$ is not equal to $I_k(l)$ in general.

Equation (2.1) requires some discussion. With partially overlapping channels, $IF(l,k)$ accurately capture the interference between l and k (this has been empirically observed

[3]We use the term BSS (Basic Service Set) to designate WLANs here, as this is the usual 802.11 nomenclature. Note that the concepts presented here apply to any kind of cell architecture composed of a base station and some clients

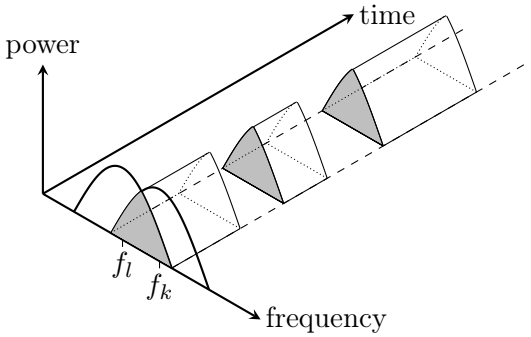

Figure 2.1 – The interference $I_l(k)$ produced by link k on link l (Eq. (2.1)) can be seen as the average sum of the volumes spanned by the channel overlaps over time. The time intervals without volume correspond to the intervals during which link k is idle.

in [MSBA06], and later been confirmed for 802.11n with channel bonding [SRYB08]). In Equation (2.1), we augment this interference factor and multiply it by the fraction of time a given interfering link is active (μ_k). This naturally extends the notion of interference to both the spectral and temporal domains (see Figure 2.1), and it accounts for the fact that a link is more likely to cause interference if its airtime is high. This is important, as it indirectly accounts for the traffic load on the different links. We also introduce it as a way to account for the difference in airtime consumption at different channel widths. Note that a seemingly reasonable extension could be to modulate $I_l(k)$ by the power that link l receives from an interfering neighbor. However, as we will illustrate in Chapter 3, the transmit and receive powers are very intricate indicators of performance (or interference) in practice[4], and we do not use them here.

Finally, let $I_A(B)$ be the *BSS interference* that a BSS A experiences from a BSS $B \in \mathcal{N}_A$. Using the link-interference, we have

$$I_A(B) = \sum_{l \in A} \sum_{k \in B} I_l(k). \tag{2.2}$$

Again, in general $I_A(B) \neq I_B(A)$.

2.3 SAW Algorithm

Recall that an efficient joint allocation of center frequencies and bandwidths needs to balance a minimization of interference with an appropriate spectrum consumption. In this chapter, we formulate the center-frequency and bandwidth allocation task as a network-wide cost minimization problem, where the cost is the *global* sum of the BSS interference, plus a *local* penalty that each BSS attributes to the exploitation of a given bandwidth. As will become clear later, this formulation conveniently exhibits optimal

[4] In particular, it could be the case that the throughput achievable on a given link is an *increasing* function of the transmit power of an interfering transmitter (see Section 3.2 for a detailed example)

Algorithm 1: SAW algorithm at BSS A

Initialization:
Setup an exponential timer of mean wake-up time $1/\lambda$
Set the temperature $T > 0$
Pick a random configuration $(f_A, b_A) \in \{\mathcal{F} \times \mathcal{B}\}$

When the timer fires:
Pick a random configuration $(f_{\text{new}}, b_{\text{new}}) \in \{\mathcal{F} \times \mathcal{B}\}$
Measure $\mathcal{K}_{i,A} := \sum_{B \in \mathcal{N}_A} (I_A(B) + I_B(A)) + \text{cost}_A(b_A)$, when A *does use* the configuration (f_A, b_A)
Measure $\mathcal{K}_{j,A} := \sum_{B \in \mathcal{N}_A} (I_A(B) + I_B(A)) + \text{cost}_A(b_{\text{new}})$, if A *were to use* the configuration $(f_{\text{new}}, b_{\text{new}})$
Compute

$$\beta_{ij} = \begin{cases} e^{(\mathcal{K}_{i,A} - \mathcal{K}_{j,A})/T} & \text{if } \mathcal{K}_{j,A} \geq \mathcal{K}_{i,A}, \\ 1 & \text{if } \mathcal{K}_{j,A} < \mathcal{K}_{i,A}. \end{cases}$$

Set $(f_A, b_A) = (f_{\text{new}}, b_{\text{new}})$ with probability β_{ij}
Reschedule the timer

solutions that can be well approximated by the steady state of a Markov chain, whose transitions are precisely determined by our algorithm. Formally, let the *energy* of the network be

$$\mathcal{E}(\mathbf{F}, \mathbf{B}) := \sum_{A \in \mathcal{A}} \sum_{B \in \mathcal{N}_A} I_A(B) + \sum_{A \in \mathcal{A}} \text{cost}_A(b_A), \tag{2.3}$$

where $\text{cost}_A(b_A)$ quantifies the penalty that BSS A attributes to using bandwidth b_A. In most cases, it is advantageous for the APs to choose a penalty function that favors larger bandwidths[5]. This formulation is similar to the energy of a magnetic system in statistical physics, where the local spin interactions correspond to the interference and an external field favors "better" bandwidths. The optimization problem is then

$$\text{minimize} \quad \mathcal{E}(\mathbf{F}, \mathbf{B}) \quad \text{over} \quad \mathbf{F}, \mathbf{B} \in \{\mathcal{F} \times \mathcal{B}\}^N. \tag{2.4}$$

SAW is described in Algorithm 1. The algorithm runs at the AP of each BSS. In the next section, we show that with SAW running at each AP, the spectrum allocation converges towards the minimum of Problem (2.4). The algorithm uses two real parameters, the average wake-up time $1/\lambda$ and a temperature T, whose role is explained in Section 2.3.1. At the AP of a BSS A, SAW is executed repeatedly, at random time intervals. During an execution, the AP randomly samples a center frequency and a channel bandwidth

[5]Note that an AP can also decide to favor narrow bandwidths if some of its links have very poor SNRs (see Section 1.1.1 of the Introduction). In practice however, we recommend to use a penalty function favoring larger bandwidths as, everything else being equal, narrow bandwidths rarely outperform wide bandwidths [CMM+08]. Furthermore, poor SNRs can be compensated by auto-rate mechanisms, which select more robust modulations on the affected links.

$(f_{\text{new}}, b_{\text{new}})$. The AP measures $\mathcal{K}_{i,A}$ (line 8), the *local* sum of (a) the interference currently experienced by the BSS A, (b) the interference caused by the BSS A on its neighbors with the current band (f_A, b_A), and (c) the penalty that A attributes to using b_A. It then measures $\mathcal{K}_{j,A}$ (line 9), the same sum if the BSS A were to use $(f_{\text{new}}, b_{\text{new}})$ instead. We explain how to measure $\mathcal{K}_{i,A}$ and $\mathcal{K}_{j,A}$ in Section 2.3.2, and we give more information on the influence of the function $\text{cost}_A(b_A)$ in Section 2.4.3. If the new band $(f_{\text{new}}, b_{\text{new}})$ sampled by the AP appears better than (f_A, b_A) (in the sense of Eq. (2.3)), it is accepted and the BSS A switches to this new band. If it is worse, a chance is left to this band, and it is accepted with a non-zero probability by the AP. The acceptance probability depends on how bad the band is: bands that appear very bad are less likely to be accepted by the AP. Having a non-zero probability of accepting worse bands is necessary in order to ensure that the algorithm does not remain stuck in a local minimum of Problem (2.4).

SAW is a Metropolis sampler for the channel center frequency and bandwidth. The main advantage of this sampling strategy is that it only needs to assess two configurations at a time. This is important, because SAW operates in a fully decentralized way and assessing a larger number of bands would be prohibitive, as it would require more measurements. In contrast, in Chapter 4, we use a different strategy based on Gibbs sampling. Gibbs sampling considers all the $|\mathcal{F} \times \mathcal{B}|$ potential configurations at each iteration, which can potentially improve convergence speed. However, it is only feasible in a setting where neighboring APs can collaborate (as in Chapter 4) hence do not need to measure all the combinations of center frequencies and bandwidths used by neighboring BSSs at each iteration. SAW retains similar asymptotic convergence properties (and requires very few iterations to converge in practice), but the number of measurements that are required in each time step is scalable with respect to the set of possible configurations, which enables the implementation to fully decentralized.

2.3.1 Convergence Analysis

We now provide an analytical characterization of the convergence of the algorithm. Let us discretize time. A time slot is started immediately before any one AP fires its timer[6]. We denote by $X_n \in \{\mathcal{F} \times \mathcal{B}\}^N$ the global state of the network at time slot n. The following theorem states that the probability distribution taken by X_n converges towards a steady distribution that largely favors the states producing low energies.

Theorem 1. *Consider a network where all the BSSs run SAW with a given temperature $T > 0$. Then X_n converges in distribution to*

$$\pi_i(T) = \frac{e^{-\mathcal{E}(i)/T}}{Z}, \tag{2.5}$$

where Z is an appropriate normalizing constant.

[6]Note that the time slots have variable durations that are only determined by the stochastic sequence of the timer events.

Proof. Because of the exponential timers used by the APs, the discrete time process X_n is a Markov chain. We use the classic nomenclature of Metropolis sampling (see for instance [Bre01], Chapter 7). For any two states $i, j \in \{\mathcal{F} \times \mathcal{B}\}^N$, we write the transition probabilities from i to j as $p_{ij} = P(X_n = j | X_{n-1} = i) = q_{ij} \cdot \alpha_{ij}$. With this notation, q_{ij} denotes the probability to sample state j when the chain is in state i, and α_{ij} is the probability, when in state i, to accept state j if it is sampled. We start by giving the expressions for q_{ij} and α_{ij} when the distributed algorithm is applied, and we then establish the convergence. In the following, we say that two states $i, j \in \{\mathcal{F} \times \mathcal{B}\}^N$ differ at exactly one BSS A if all the BSSs have the same configuration of center frequency and bandwidth in states in i and j, except for the BSS A that has a different configuration in i and j.

Because a time slot starts whenever a timer expires, exactly one AP out of the N APs wakes up at each time slot. This AP samples uniformly at random one new configuration in $\{\mathcal{F} \times \mathcal{B}\}$. Therefore, for any two states $i \neq j$,

$$q_{ij} = q_{ji} = \frac{1}{N|\mathcal{F} \times \mathcal{B}|},$$

if i and j differ at exactly one BSS, and $q_{ij} = q_{ji} = 0$ otherwise.

We now characterize the acceptance probabilities α_{ij}. Observe that given $i = (\mathbf{F}, \mathbf{B}) \in \{\mathcal{F} \times \mathcal{B}\}^N$, we can rewrite $\mathcal{E}$ as

$$\mathcal{E}(i) = \sum_{\{A,B\} \subseteq \mathcal{A},\, A \neq B} (I_A(B) + I_B(A)) + \sum_{A \in \mathcal{A}} \text{cost}_A(b_A).$$

This can in turn be rewritten as a sum over all the cliques C of the BSS neighborhood system

$$\mathcal{E}(i) = \sum_C V(C),$$

with

$$V(C) = \begin{cases} I_A(B) + I_B(A) & \text{if } C = \{A, B\} \text{ and } A \in \mathcal{N}_B, \\ \text{cost}_A(b_A) & \text{if } C = A, \\ 0 & \text{otherwise.} \end{cases}$$

Note that this means that $\mathcal{E}$ *derives from a potential* (see [Bre01]). For our purpose, it implies that if we consider any two states $i, j \in \{\mathcal{F} \times \mathcal{B}\}^N$ that differ at exactly one BSS A, we have

$$\mathcal{E}(i) - \mathcal{E}(j) = \mathcal{K}_{i,A} - \mathcal{K}_{j,A},$$

where $\mathcal{K}_{i,A}$, respectively $\mathcal{K}_{j,A}$, are the interference values observed by A at lines 7 and 8 of Algorithm 1, when the network is in state i, respectively in state j. Therefore, we have

$$\alpha_{ij} = \beta_{ij} = \min\{1, e^{(\mathcal{E}(i) - \mathcal{E}(j))/T}\},$$

where β_{ij} is the local acceptance probability used at line 10 of Algorithm 1.

X_n is ergodic and it is easy to check that $\pi_i(T)$ satisfies the detailed balance equations $\pi_i p_{ij} = \pi_j p_{ji}$ when plugging in the expressions for q_{ij} and α_{ij}. Therefore, (2.5) is the unique stationary distribution of X_n. Also, since the chain is aperiodic, X_n converges in distribution to (2.5). □

Furthermore, we can use a classic Markov chain argument to show that the convergence to steady state happens at geometric speed. The following proposition is a direct consequence of Theorem 3.3 in [Bre01], Chapter 6, together with the Perron-Frobenius Theorem for nonnegative matrices.

Proposition 1. *Let $\mathbf{P} = [p_{ij}]$ be the transition matrix of the Markov chain X_n and $i \in \{\mathcal{F} \times \mathcal{B}\}^N$ an initial state. For all $n \geq 1$, we have*

$$d_V(\delta_i^T \mathbf{P}^n, \pi)^2 \leq \frac{\rho^{2n}}{4\pi_i},$$

where $\delta_i \in \{0,1\}^{|\{\mathcal{F}\times\mathcal{B}\}^N|}$ is the vector with a 1 at the position of the i-th state and 0's elsewhere, d_V is the distance in variation and $\rho < 1$ is the second largest eigenvalue modulus of $\mathbf{P}$.

The distribution (2.5) puts "exponentially" more mass on configurations that produce low global energies, especially if T is small. Indeed, consider the set of *global minima* of problem (2.4)

$$H := \{i \in \{\mathcal{F} \times \mathcal{B}\}^N : \mathcal{E}(i) \leq \mathcal{E}(j) \; \forall \; j \in \{\mathcal{F} \times \mathcal{B}\}^N\},$$

then $\pi_i(T)$ is maximal on H, and

$$\lim_{T \downarrow 0} \pi_i(T) = \begin{cases} \frac{1}{|H|} & \text{if } i \in H, \\ 0 & \text{if } i \notin H. \end{cases}$$

(see Example 8.6, Chapter 7, in [Bre01]).

The temperature T represents a trade off between *exploration* and *exploitation*. In particular, a small value of T ensures near asymptotic convergence to the global minima of problem (2.4). Larger values of T can be used to introduce more randomness that help the algorithm to avoid being trapped in a local minimum. However, as we observe in Section 2.4.2, realistic network topologies convey enough natural randomness so that $T \sim 0$ yields the best results in practice. This also implies that the algorithm converges to global minima of Problem (2.4).

2.3.2 Interference Measurements

All the decisions taken by SAW rely on the measurements of $K_{i,A}$ and $K_{j,A}$ at lines 7 and 8 of Algorithm 1. At the AP of BSS A, computing $K_{i,A}, K_{j,A}$ requires measuring the link-interference between links belonging to A and links in neighboring BSSs. If any neighbor of A uses a band that partially overlaps with A and comprises some links with a non-zero airtime, it contributes to the interference term. Thus, in order to evaluate $\mathcal{K}_{i,A}$, respectively, $\mathcal{K}_{j,A}$, the algorithm needs to measure the link-interference in *all* the bands that overlap with (f_A, b_A), respectively, with $(f_{\text{new}}, b_{\text{new}})$. We refer to these measurements as *out-of-band* measurements, because to be performed they require tuning to different bands.

Micro-Sensing

We enable out-of-band measurements to be taken by implementing what we call *micro-sensing* operations. After randomly picking a new band, the AP of a BSS A computes the list of all bands that could potentially interfere with the current band (f_A, b_A) and the sampled band $(f_{\text{new}}, b_{\text{new}})$. By knowing $\mathcal{F}$, $\mathcal{B}$ and the spectrum masks defined in [IEE12], this list is straightforward to obtain. The AP then tunes to each of these bands for a short amount of time. Then, instead of scanning all of these bands at once, the AP comes back to the operating band (f_A, b_A) between each individual scan. The whole procedure is depicted in Figure 2.2. The amount of time spent in out-of-band sensing must be large enough for the nodes to have a fair chance of efficiently monitoring the band, and small enough so as not to disrupt traffic. This duration also depends on the bandwidth of the configuration currently being scanned, because the time required to send or capture a packet at a given rate is inversely proportional to the channel bandwidth; therefore, larger bandwidths can be monitored faster. We denote by t_{m-s} the overall time taken by one micro-sensing operation. As a micro-sensing operation requires switching back and forth between the operational and the monitored band, we have

$$t_{m-s} = 2t_{switch} + t_{sensing}, \tag{2.6}$$

where t_{switch} is the time required to tune to the target center-frequency and bandwidth, and $t_{sensing}$ is the time spent monitoring, which depends on the channel bandwidth. In our implementation, we set $t_{sensing} = 240/b$ ms, where b is the bandwidth of the band to monitor in MHz. This duration is long enough to capture packets sent at low rates, but short enough (below 50 ms) to accommodate delay-sensitive traffic, even when a 5 MHz band is being sensed.

A trade-off must be found between the amount of sensing and the accuracy of the interference estimation. As one micro-sensing operation is fast and inexpensive, our implementation senses each band several times to increase the probability of detecting potential neighbors, even if they do not transmit back-to-back packets. Even in this case,

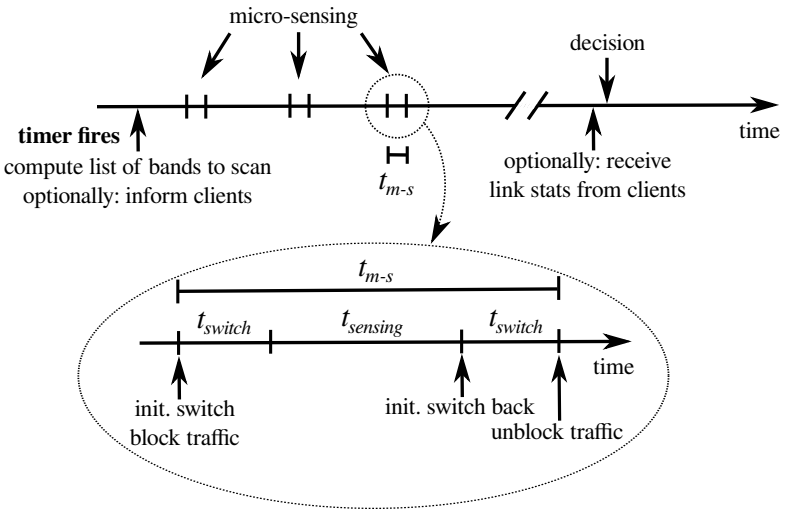

Figure 2.2 – Implementation of out-of-band monitoring with micro-sensing.

the algorithm could miss some neighbors that send only sporadic traffic and occupy little airtime. Note however that, by definition, these neighbors do not consume a significant portion of the available capacity, hence missing them is less critical.

During each micro-sensing period, the AP monitors the medium and gathers link statistics. For each packet that it overhears, the AP records the corresponding band, a link ID (specifically, the pair of source-destination MAC addresses), and it keeps an estimation of the airtime ratio of the link by computing the airtime consumed by the packet. This airtime is computed from the length of the packet, its physical rate, and from the bandwidth that it occupies

Client-Aware Extension

Up to this point, the measurements are performed at the AP only. This is indeed a desirable feature, as it does not require client-side modification. In this case however, the AP could miss hidden nodes that interfere with some of its clients. This problem can be important in practice, as observed in works proposing centralized channel-assignment schemes [MBB+06, RMAQ07]. For this reason, and to remain consistent with our link neighborhood definition of Section 2.2.1, we propose an *optional* extension of SAW that performs monitoring at the clients as well. When the timer of an AP fires, this AP broadcasts a modified beacon that contains the sampled frequency band and a schedule for the corresponding micro-sensing operations. When the clients receive this beacon, they schedule the micro-sensing of the bands accordingly. Once they have monitored all the required bands, the clients wait a small, random amount of time (in order to avoid inter-client collisions) and send back to the AP all the statistics for the links that they overheard. This feature mitigates the impact of links that are hidden from the

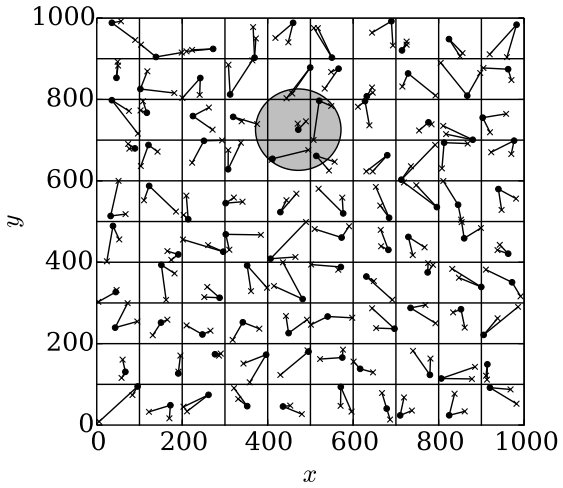

Figure 2.3 – Example of a grid composed of 100 cells, each of which contains one AP and two clients. The interference radius is $R = 100$m, as shown for one particular AP on the figure.

APs, but it comes at the price of client-side modifications. We implemented SAW both with ("client-aware"), and without ("client-agnostic") client monitoring. We compare the performance of the two versions in Section 2.5.

2.4 Simulation Results

2.4.1 Simulation Setup

Before giving a detailed evaluation of SAW on an indoor 802.11 testbed in the next section, we first use simulations in order to investigate its self-organization properties on large ecosystems of interfering WLANs. To this end, we developed our own simulator in Python. We do not simulate at the packet level, because this would not scale well to such large networks. Instead, we use simple models for computing interference and capacity.

We assume Gaussian white noise, so that a link l benefits from a theoretical capacity $C_l = b_l \cdot \log(1 + \text{SINR}_l)$, where SINR_l refers to the signal to interference-plus-noise ratio at the receiver of l. For any two nodes i and j within interference range, we compute the power received by j from i to be proportional to $d(i,j)^{-\alpha} \cdot IF(i,j)$, where $d(i,j)$ is the Euclidean distance between i and j, α is the path loss exponent – that we take equal to 3 in our simulations – and $IF(i,j)$ is the corresponding interference factor [MSBA06]. Note that this simple formulation for the capacity captures the trade-off between interference mitigation and the usage of larger bandwidths, through the logarithmic and pre-logarithmic terms, respectively. Unless otherwise stated, we take the penalty function to be $\text{cost}_A(b_A) = 1/b_A$ for each BSS, where b_A is the bandwidth in MHz. Such a function favors wider bandwidths, and we evaluate its effect in Section 2.4.3.

Unless otherwise stated, we consider a 1000m × 1000m square grid, divided in 100 square cells. Each cell is occupied by one BSS, which is composed of one AP and two clients. The AP and the clients are placed uniformly at random within their cell. The interference radius is $R = 100$m (see Figure 2.3 for an example). The results are insensitive to the scale of the units, and this setting can, for instance, be thought of as a simple model for residential WLANs, where each cell corresponds to an apartment in a building. We simulate downlink traffic. The APs transmit 100% of the time and the clients are idle. We consider a 2.4 GHz scenario, with eleven center frequencies separated by 5 MHz, and four possible channel bandwidths (5, 10, 20 and 40 MHz). At initialization, each BSS picks a random channel and uses the largest width.

We evaluate three metrics: **(1)** The total amount of **interference** in the network (specifically, the first term of the energy function $\mathcal{E}$ given by Eq. (2.3)); **(2)** the sum of **capacities** of links in the network; and **(3)** the **Jain fairness index** of the capacities experienced by each BSS. The Jain fairness index is given by

$$\frac{\left(\sum_{A\in\mathcal{A}} C_A\right)^2}{N\sum_{A\in\mathcal{A}} (C_A)^2},$$

with C_A denoting the sum of the link capacities of BSS A. We show the median values over 50 simulation runs, and the error bars on the plots are the 95% confidence intervals for the median.

2.4.2 Influence of the Temperature T

The temperature T represents a trade-off between the likelihood of remaining stuck in a local optimum and the asymptotic efficiency of SAW (see Section 2.3.1). To understand this trade-off, we perform simulations with various temperatures spanning six orders of magnitude. Each simulation runs until each AP has performed on average 30 iterations of SAW. In order to conveniently display the three metrics on a common plot, we normalize the capacity and the interference by their largest values. Figure 2.4 shows that SAW performs better, with respect to all the metrics, when T is small. In practice, this implies that the risk of remaining trapped in a local optimum is very low and small values of T can be used (specifically, it implies that greedy policies perform well on this kind of problems). As such values also ensure the best asymptotic performance of SAW, we use $T = 0.1$ in the following.

2.4.3 Capacity versus Interference

In this section, we explore the influence of the weight that each BSS puts on its local penalty function. We consider a scenario where each BSS A uses the function $\text{cost}_A(b_A) = c/b_A$, where c is a weighting parameter. The BSSs can use different cost functions, according to

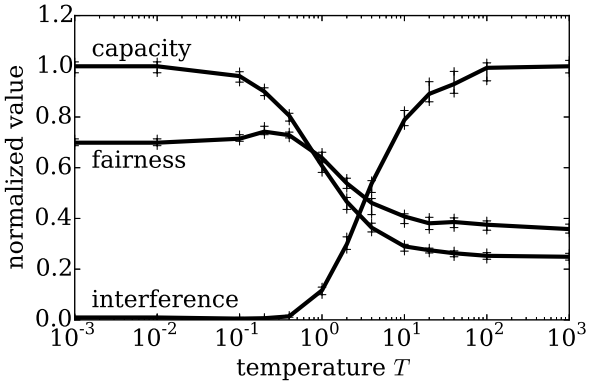

Figure 2.4 – Capacity, fairness and interference as functions of the temperature T.

the local benefit of each bandwidth. However, we study this particular function in detail because it decreases with the bandwidth b_A, and therefore exhibits well the inherent conflict between interference mitigation and the maximization of theoretical capacity. It is also a practical function that the BSSs can use whenever using a larger bandwidth would give them a better throughput. As already mentioned, this is often the case in practice, when the links have sufficiently good SNRs [CMM+08]. We this cost function, the energy function given by Equation (2.3) becomes

$$\mathcal{E}(\mathbf{F}, \mathbf{B}) = \sum_{A \in \mathcal{A}} \sum_{B \in \mathcal{N}_A} I_A(B) + c \cdot \sum_{A \in \mathcal{A}} 1/b_A.$$

We show the influence of c on our three performance metrics in Figure 2.5. When c is zero, no weight is given to the local preferences of the BSSs, and the scheme targets only global interference minimization. In this case, it indeed finds interference-free configurations in a decentralized way. This setting is well suited for fixed-width channel allocation, but it is not appropriate for varying bandwidths. Indeed, in this case, there is an increase in capacity of up to 66% when using configurations that use the spectrum more aggressively and yield a non-zero interference level (with $1 \leq c \leq 6$). Using too large a value for c, however, decreases the benefits of all three metrics. Such configurations put much weight on local costs, which creates prohibitive interference levels.

As discussed in Section 1.1.1 of Chapter 1, it is intuitive that the best operational setting should depend on the network density: for networks that are spatially dense, it makes sense to give priority to interference mitigation. In contrast, for sparse networks where the nodes have few or no neighbors, it is advantageous to give more priority to local preferences. This is illustrated in Figure 2.6, where we plot the total capacity when the spatial density of the network varies, for several values of c[7]. As expected, $c = 0$ performs

[7]For this particular experiment, in order to vary the spatial density, we do not simulate one BSS per cell of the grid. Instead, we draw the coordinates of each AP uniformly at random in the 1000m ×1000m

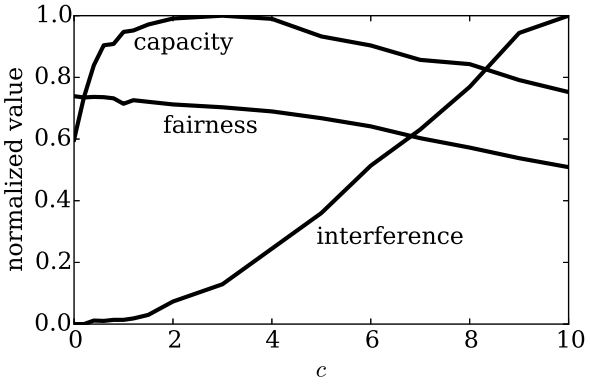

Figure 2.5 – Capacity, fairness and interference as functions of the penalty weight c.

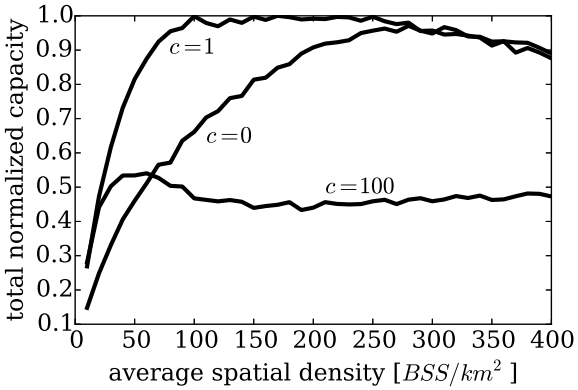

Figure 2.6 – Evolution of the total network capacity as a function of the spatial node density, for several values of the penalty weight c.

the best on dense networks, and a large c is best for sparse networks. However, a small but non-zero value of c obtains the best performance in all regimes. This is remarkable, as it implies that a single energy function (using a fixed parameter c) enables the algorithm to operate in the best regime of the interference-capacity trade-off, irrespective of the spatial node density. Intuitively, the fact that c should be small but positive means that the most of the weight should be put on the first term of Equation (2.3) (interference minimization), but that it is also important to give a non-zero weight to the second term (maximization of spectrum usage).

2.4.4 Performance

We now evaluate the three metrics as functions of the number of iterations of SAW executed by the BSSs. Figure 2.7 considers two cases, with 6 or 11 channels available (the

area, and each client is randomly placed in the disc of radius $R = 100$m centered at its AP.

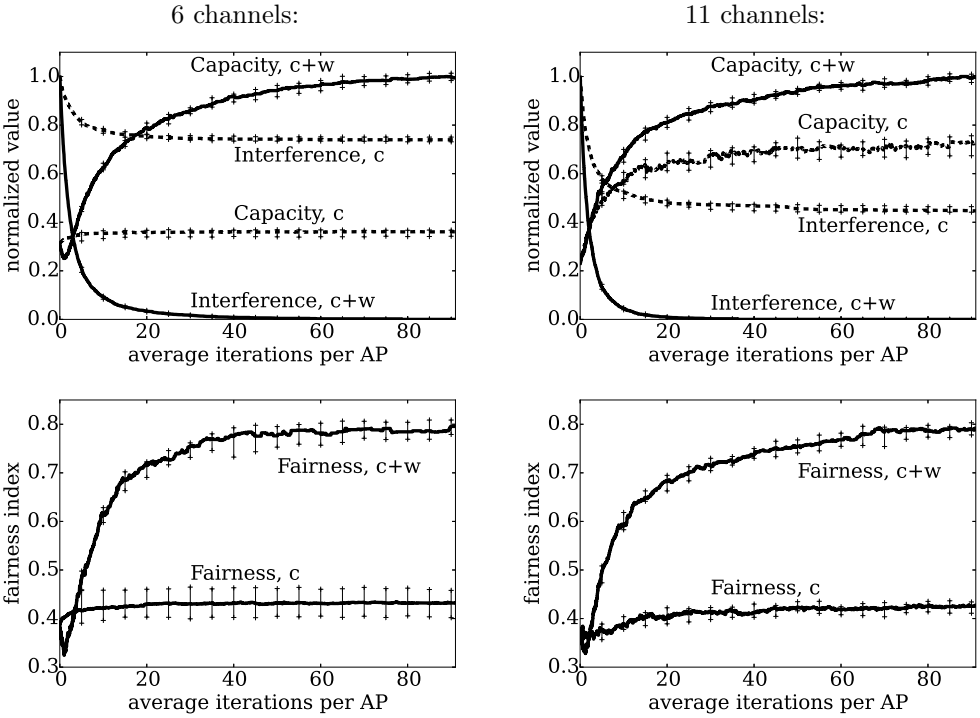

Figure 2.7 – Capacity, interference and fairness as functions of the number of iterations. We show the values obtained when SAW tunes both the channel center-frequency and bandwidth (denoted "c+w"), and when it tunes only the center frequency (denoted "c").

latter is equivalent to the 2.4 GHz spectrum case depicted in Figure 1.3). In addition, we compare with a case where SAW only tunes the center frequency (and not the bandwidth). We make the following observations:

- By tuning both the center frequency and the bandwidth, SAW drastically improves all three metrics. Interference is completely mitigated with 11 channels and nearly mitigated with 6 channels. The capacity is multiplied by a factor 2 to 4, compared to random channel allocations.

- Jointly tuning the center frequency and the bandwidth offers drastic improvements compared to center frequency only, especially when the available spectrum is scarce. In this case, the BSSs can naturally switch to smaller bandwidth and operate in interference-free configurations.

- SAW quickly finds efficient allocations (even though the exact convergence is asymptotic). Besides, an iteration of SAW only involves the assessment of two configurations and is inexpensive to realize in practice.

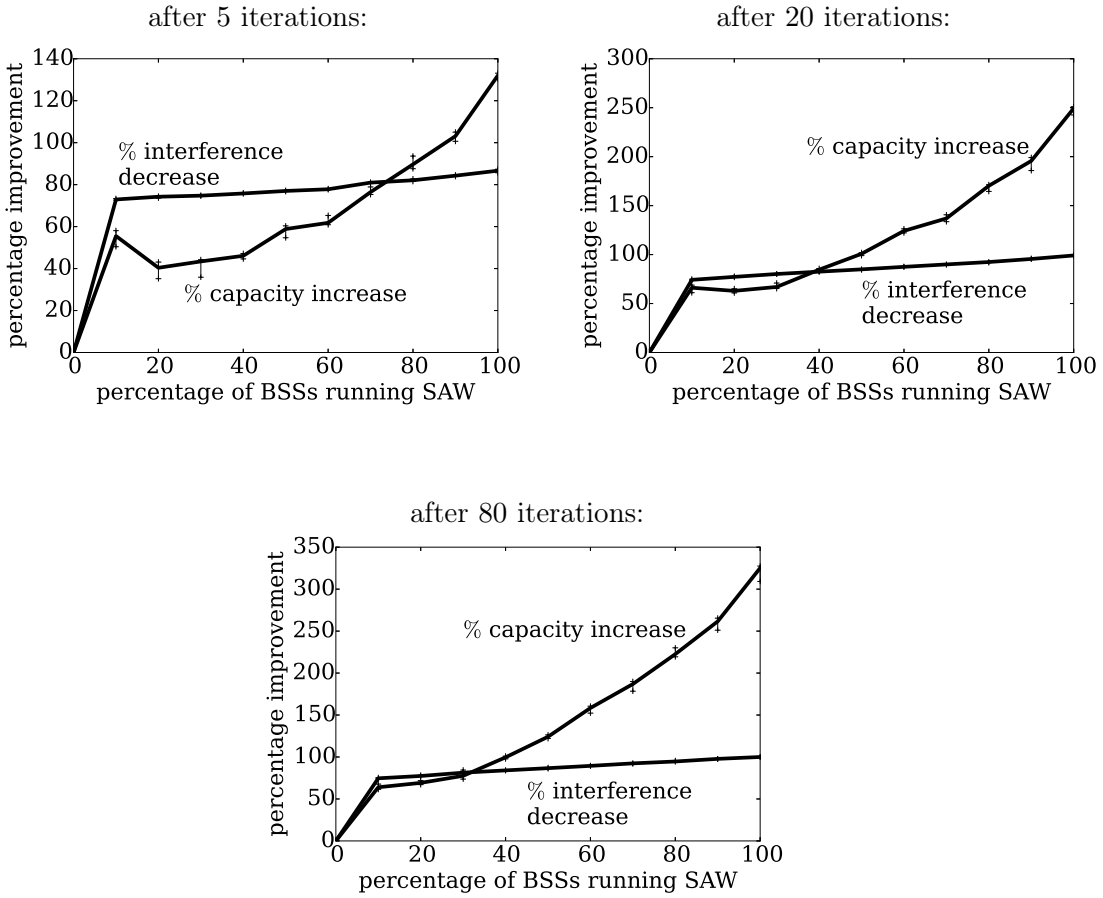

Figure 2.8 – Percentage of improvement (compared to random allocations of fixed bandwidth channels), as a function of the percentage of BSSs using SAW.

2.4.5 Influence of the Fraction of BSSs Running SAW

We evaluate scenarios where SAW runs on randomly chosen subsets of BSSs. We then compute capacity increase and interference decrease observed *by the BSSs running SAW*, compared to the initial random allocations of fixed bandwidth channels. Figure 2.8 shows results when the fraction of BSSs running SAW varies from 0% to 100%, after an average of 5, 20 and 80 iterations per AP. The capacity always increases for the BSSs running SAW. Note that after 5 iterations, this capacity gain is not monotonic with respect to the fraction of BSSs running SAW. We attribute this to the larger convergence time due to the competition between an increased number of BSSs running SAW. Waiting for more iterations enables the APs to explore more configurations and attenuates this effect. Nevertheless, even a small percentage of BSSs running SAW quickly produces a significant capacity increase, which gives users incentives for incremental deployments.

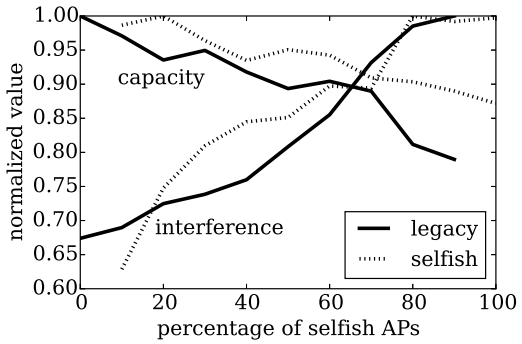

Figure 2.9 – Average capacity and interference (observed by each AP), for both selfish and legacy APs.

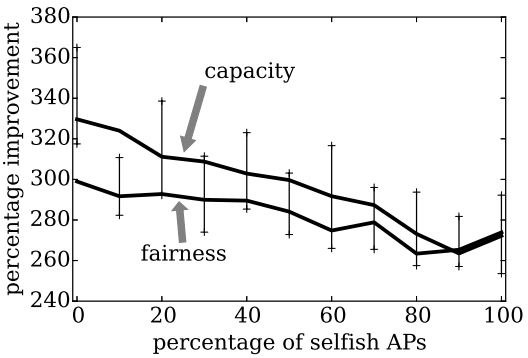

Figure 2.10 – Improvement (compared to random allocations of fixed-width channels) of capacity and fairness as functions of the proportion of selfish APs in the network.

2.4.6 Selfish APs

In lines 7 and 8 of Algorithm 1, when deciding on whether to adopt a new configuration or not, an AP of a BSS A considers not only the interference received from its neighbors ($I_A(B)$), but also the interference that itself creates on its neighbors ($I_B(A)$). Note that in general, due to the asymmetric nature of wireless interference, these two interference terms need not be equal. With this way of sampling configurations, the APs take into account the interference that they produce on their neighbors, which might go against their immediate best interest. This "equity" is necessary for establishing the convergence of the algorithm in Section 2.3.1, but it could be harmed if some APs do not behave socially (i.e., according to the legacy algorithm). We now consider such a scenario, where some subset(s) of APs run a selfish version of the algorithm. These selfish APs do not take the interference created on their neighbors into account and, for these APs, the first term of lines 7 and 8 of Algorithm 1 is replaced by $\sum_{B \in \mathcal{N}_A} I_A(B)$.

Figure 2.9 shows the average capacity and interference experienced by the two categories of BSSs – with selfish and legacy APs – for varying proportions of APs running the selfish version of the algorithm. As expected, the maximum capacity is obtained when

all nodes behave according to the legacy algorithm. For both selfish and legacy APs, the capacity decreases with the proportion of selfish APs, due to increased interference levels. In general, the selfish APs obtain a slightly better capacity than the legacy APs. However, the performance loss due to the presence of selfish APs is not drastic and, even when 100% of the APs are selfish, they still achieve 87% of the capacity achieved when all APs are legacy. This is confirmed in Figure 2.10, where we plot the percentage of improvement of capacity and fairness obtained with our algorithm (compared to random allocations of fixed-width channels), with a varying fraction of selfish APs. Although the presence of selfish APs slightly decreases performance, the improvements still remain above 2.6× for both capacity and fairness.

We explain these large gains in the presence of selfish APs by observing that although the interference levels experienced by two neighboring links, l and k, are asymmetric, they are often strongly correlated. Therefore, if (the AP of) link k seeks to selfishly minimize its own interference level, it is likely to also decrease the interference that it produces on l. As a consequence, the price of anarchy remains limited in these cases, and the algorithm shows some robustness to the presence of selfish APs. A similar conclusion was reached in [EPT07] in the context of Gaussian interference games.

2.5 Testbed Results

2.5.1 Implementation Description

We now show some results obtained on the 802.11 testbed described in Appendix A. We use 21 nodes of the testbed, which form 10 BSSs spread over the second floor of the EPFL BC building (see Figure 2.11). We perform the experiments using 802.11g and the bandwidths of 5, 10 and 20 MHz available with the open-source Atheros ath9k driver. As shown on Figure A.2 of the appendix, the cutoff values match well the widths of the channels, but there is leakage on adjacent channels. There is also a 3 dB gain when the bandwidth is divided by two. Therefore, when implementing the link-interference computation using Equation (2.1), we consider the transmit and receive masks as perfect bandpass filters whose cutoff frequencies match the channel bandwidths, with an extra 2.5 MHz on each side. We empirically observe that this margin is enough to alleviate adjacent-channel interference. We use the `debugfs` files described in Appendix A to reconfigure the center frequencies and bandwidths in a few tens of milliseconds. We give more details on these timings in Section 2.5.3. Finally, we performed all experiments during the night in the 2.4 GHz spectrum band, using the default rate adaptation mechanism of ath9k (Minstrel). The 5 GHz spectrum contains more channels, but we use the 2.4 GHz spectrum in order to create interference-rich scenarios with overlapping channels, where efficient spectrum assignments are non-trivial. SAW is implemented in userspace using the *Click* modular router [KMC+00]. We created four Click elements that, in total, consist of about 2500 lines of C++ code. We also modified the Click radiotap parsing elements to be able to

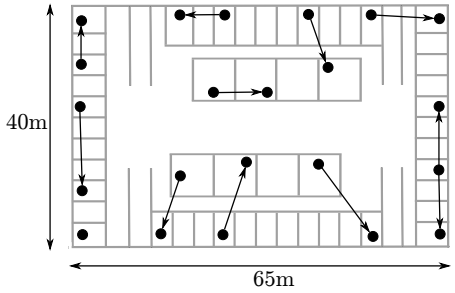

Figure 2.11 – Map of the nodes used for the experiments. The BSSs are shown with arrows directed from the APs to their clients.

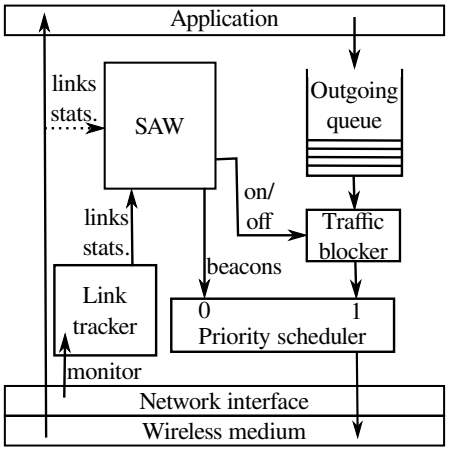

Figure 2.12 – Implementation of SAW at an access point.

collect link and airtime statistics. A schematic view of the role of each element within the networking stack of an AP is shown in Figure 2.12. The core logic of the algorithm is fed by link statistics that come from a link-tracker module and optionally from the clients of the BSS (for the client-aware version of SAW). The link-tracker module is connected to a monitor interface that captures frames sent by the AP and neighboring BSSs. When performing micro-sensing, SAW temporarily blocks outgoing traffic, in order to reduce packet losses. Control traffic between AP and clients (switch announcements, scanning requests and scanning replies) is prioritized over data traffic in order to increase the accuracy of the scheduled switching times.

All BSSs use $\text{cost}_A(b_A) = 1/b_A$, the temperature is set to $T = 0.1$ and the mean wake-up period is $\lambda = 4$ minutes. Such a value offers a good trade-off between stability and reactivity to, for instance, the apparition or disappearance of a neighboring network. The interval between two micro-sensing is set to 500 ms, and each band is sensed five times.

2.5.2 Performance of SAW

We performed several experiments on four scenarios: with UDP or TCP traffic, and with the client-agnostic or client-aware versions of SAW. The traffic is backlogged, which represents a frequent use case where all the capacity is used, for instance when several clients download simultaneously from the Internet. All BSSs start in channel 6 with a bandwidth of 20 MHz (which represents a worst case in terms of spectrum usage). As a benchmark, we use a centralized channel assignment based on graph-coloring[8]. Specifically, we build an inter-BSS interference graph by having all the APs broadcast one packet (of size 1000B), each second during one hour. Two BSSs are neighbors if one of their APs receives at least P% of the beacons sent by the other AP. Then, using the DSATUR graph-coloring algorithm [Bré79], we take the largest value of P such that this graph is 3-colorable. Finally, we use the corresponding coloration to assign one of the three non-overlapping channels (channels 1, 6 and 11) to each BSS. This procedure is centralized and is a reasonable upper-bound of what can possibly be achieved with an unplanned deployment.

Figure 2.13 shows the average sum and the standard deviations (over 20 independent runs) of the throughputs achieved by each link, for dowlink traffic (from APs to their clients). Figure 2.14 shows the results for uplink traffic. We also show the average obtained with the benchmark. In each scenario, SAW starts at 600 seconds. The client-aware version performs slightly better, both for UDP and TCP traffic. In general, SAW settles for spectrum assignments that perform similar or better than centralized graph-coloring. The extra gain is due to the fact that SAW adapts both the frequency and bandwidth of the channel. In these experiments, much of the gain already comes after one or two iterations of SAW per BSS (iterations happen every 240 seconds on average), and the algorithm settles to efficient allocations after approximately three iterations per BSS on average. We emphasize that these results are obtained by using a completely decentralized and online implementation.

This increase in network capacity does not come at the cost of fairness. In particular, it is not obtained by starving some of the BSSs for the benefit of others. This appears clearly in Figure 2.15, where we show the evolution of the average Jain's fairness index of the throughput achieved by all the BSSs over time, for the first scenario (UDP traffic with the client-agnostic version of SAW). Fairness in the remaining scenarios showed similar trends.

[8]Note that with $11 \cdot 3$ channel-width combinations and 10 BSSs, the state space has size 33^{10}. An exhaustive search for the "real" best configuration is therefore impossible.

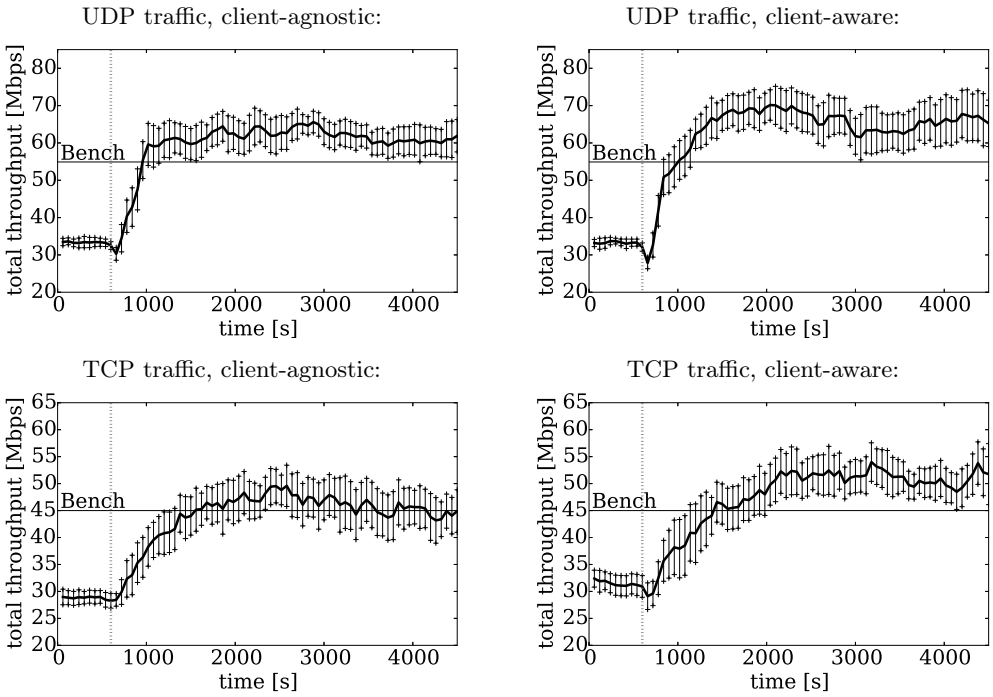

Figure 2.13 – Sum of the link throughputs obtained by the 10 BSSs with downlink traffic. SAW is started at 600 seconds. The "Bench" line is the average throughput obtained with a centralized graph coloring approach that uses the 3 non-overlapping channels with a width of 20 MHz.

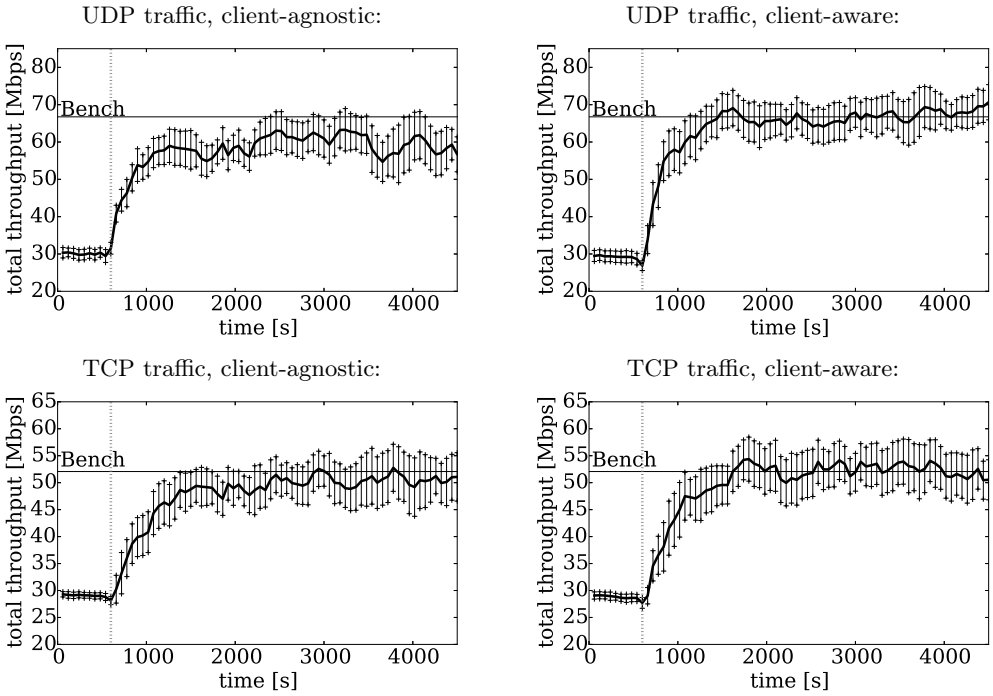

Figure 2.14 – Same experiments as in Figure 2.13, but with uplink traffic.

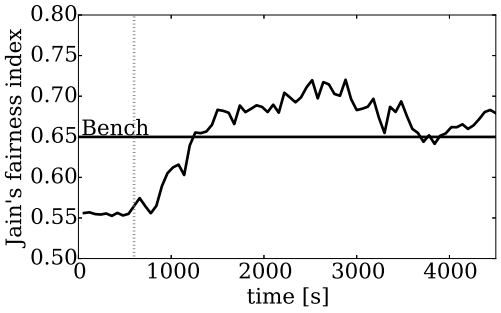

Figure 2.15 – Jain's fairness index for the scenario with UDP downlink traffic and client-agnostic version.

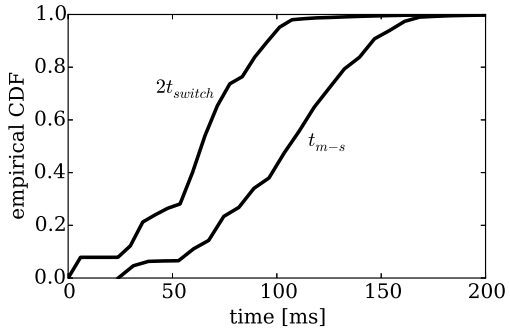

Figure 2.16 – Empirical CDFs of the switching and micro-sensing durations.

2.5.3 Micro-Sensing Evaluation

We now evaluate the potential disturbance produced by the micro-sensing procedures. Because traffic is blocked while the AP (and optionally the clients) perform out-of-band monitoring, frames can experience an additional delay of up to t_{m-s} (see Eq. (2.6)). Figure 2.16 plots the CDFs of $2t_{switch}$ and t_{m-s} during the experiments of Section 2.5.2. Although t_{m-s} typically remains below 150 ms, this could still be non-negligible for delay-sensitive traffic. However, this delay is mostly due to the hardware switching-time, which is relatively high on our cards. Indeed, Atheros and other manufacturers report switching times of 2 ms or less for newer 802.11 chipsets[9]. With such chipsets, the switching overhead becomes negligible, and the additional delay of the micro-sensing procedure can be upper-bounded by about 50 ms. This is low enough to be tolerated by most delay-sensitive applications.

We now show the effect of micro-sensing on TCP traffic. Figure 2.17 shows the throughput of two close-by links, each with fully backlogged TCP traffic. At the beginning of the experiment, both links use channel 1 with a bandwidth of 20 MHz. After 60 seconds, the AP of link 1 (the transmitter of this link) fires its timer and samples a new band

[9]For instance, Atheros reports switching times of 2 ms for its 802.11a/b/g/n AR9390 chipset. See: `http://www.qca.qualcomm.com/wp-content/uploads/2013/11/AR9390.pdf`

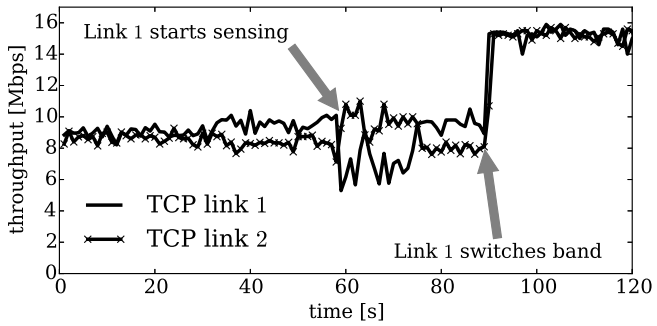

Figure 2.17 – Micro-sensing with TCP traffic.

(channel 11, 20 MHz). From 60 to 75 seconds, the AP of link 1 performs micro-sensing for all the bands that partially overlap with channel 1 or channel 11 (the micro-sensing interval is 500 ms). At 75 seconds, the AP of link 1 decides to switch to the new band. From 75 to 90 seconds, it broadcasts modified beacons containing the time of the scheduled switch, which takes place at 90 seconds. Out-of-band sensing temporarily slightly reduces the TCP throughput. However, the throughput degradation is only marginal, even though our implementation of SAW is in user-level and our hardware has a relatively high switching latency.

2.6 Related Work

The problem of allocating channels *without* considering channel bandwidth has been largely studied in the context of cellular networks (see e.g. [AHK+03]). It is commonly cast as a graph-coloring problem, where an edge corresponds to interference between two cells, and the set of available colors corresponds to the set of channels. Because graph-coloring is NP-complete for general graphs, heuristics are used to solve it (typically, techniques based on the DSATURE algorithm [Bré79]). These techniques have been adapted to 802.11 WLANs as well [VFP09]. The primary drawbacks are that they require a centralized knowledge of the interference graph and usually fail to capture much of the granularity of the interference between any two cells [AIKP08]. Some channel-allocation schemes have been developed specifically for WLANs. [MBB+06] explicitly takes into account interference at the clients of each WLAN. However, it does not provide any optimality guarantee and it requires all the APs to be under a single administrative domain. MAXchop is a distributed algorithm that runs at the APs and computes channel hopping sequences [MSA+06]. Unfortunately, it can get stuck in a local minimum, and there is no guarantee that the allocation patterns minimize interference across the network. In addition, MAXchop is not transparent, as it requires APs to periodically scan all the channels. Scanning can take up to several seconds and heavily disrupts communications. In [RMAQ07], the authors show that accounting for traffic demands when assigning channels can yield better performance. However, their

algorithm is centralized. [KBC+07] proposes a provably optimal distributed channel-assignment algorithm that uses a Gibbs sampler [Bre01]. Because it requires APs to run full channel scans to discover all the channels used by their neighbors, the algorithm is not appropriate for online and decentralized operation. These scans are necessary to compute the so-called partition function of the Gibbs measure used by APs to choose a new channel. Gibbs samplers have been used for distributed resource allocation in different contexts [MPB07, BPK+10, BMS11], and the utility-optimal algorithm proposed in Chapter 4 is also based on Gibbs sampling.

A distributed algorithm for channel assignment is presented in [LCBM12]; it does not require communication between access points, as in our work. The approach is based on decentralized constraint satisfaction [DBL13], and it provably solves the graph coloring problem in a distributed way if the number of available orthogonal channels is at least equal to the chromatic number of the underlying interference graph. Graph coloring only accounts for the presence or absence of interferers on a given band, irrespectively of the actual interference level. Furthermore, it does not capture the interference-capacity tradeoff, and it is not appropriate for selecting operating bandwidths (as the coloring a graph can be made easy by having each BSS select a small bandwidth, which is definitely inefficient in general).

Compared to the above works, SAW provably converges towards the stationary distribution of a Markov random field, but compared to [KBC+07], the costly computation of the Gibbs partition function can be avoided by using a Metropolis sampler (see Section 2.3). SAW is also traffic-aware, in the sense that it explicitly accounts for the airtime consumed by each link when computing the interference. But most importantly, SAW allocates bandwidths jointly with center frequencies, which none of the channel-allocation techniques does: these techniques solve a fundamentally different problem, which consists in maximizing the separation between the channels used by neighboring nodes. Because the channel bandwidth directly affects the experienced capacity, this goal cannot be considered in isolation in our case.

Recent work has shown that the channel bandwidth has quite an impact on interference and overall performance [CMM+08]. Shortly after [CMM+08], the work in [MCW+08] formulated frequency and channel-width assignment as an integer linear program and proposes efficient centralized heuristics. More recently, [RSBC11] proposed a centralized spectrum assignment algorithm and gives useful information on the trade-offs involved when tuning channel center frequencies and bandwidths. Here again, both [MCW+08] and [RSBC11] target enterprise networks, as they rely on the presence of a centralized coordinator. Such a coordinator does not exist for domestic WLAN deployments.

The problem of spectrum allocation has also been studied in the context of cognitive radios for white-space networks [YBC+07, BCM+09]. In particular, [YBC+07] considers the problem of efficiently packing *time-spectrum blocks*. The authors propose a distributed

algorithm, but it requires a dedicated control channel. Such a channel is not available in the context of unplanned WLAN deployments.

2.7 Summary

In this first chapter, we have presented SAW, a decentralized algorithm that finds efficient variable-width spectrum configurations for WLANs. We have thoroughly validated its performance with testbed experiments and simulations. The spectrum-allocation problem is formulated as the global optimization of an energy function, composed of neighbor interactions (capturing interference) and local bandwidth preferences (capturing capacity). When the network conditions do not change, SAW converges towards global minima of this function. In real dynamic settings, SAW constantly adapts spectrum usage. We have identified simple energy functions that enable the algorithm to solve the interference-capacity trade-off, irrespectively of the network spatial density. Due to its underlying Metropolis formulation, where only one new configuration is sampled at a time, the number of measurements required by SAW scales nicely with the total number of available channels and bandwidths. This property enables SAW to operate in a fully decentralized way and target domestic network deployments. In the next two chapters, we explore a different point of the design space and focus our attention on enterprise networks; we will see that it is possible to target more sophisticated optimization criteria when some amount of collaboration is allowed between neighboring APs.

3 Performance Prediction for Arbitrary Configurations

3.1 Introduction

In Chapter 2, we designed a spectrum assignment algorithm that optimizes an explicit balance between interference and spectrum usage. Although the resulting algorithm is simple and efficient, its optimization objective (specified by Equation (2.3)) does not relate directly to the performance achievable in each BSS – instead, it uses the sum of spectral overlaps and bandwidths as proxies for assessing the quality of each configuration. The main reason is that the actual achievable performance of 802.11 networks (especially those using variable bandwidths) is difficult to predict, because it is a very intricate function of spectral configurations and other network properties (such as the traffic loads of neighboring interferers).

In this chapter, we take a step back from the spectrum-assignment problem, and focus our attention on the problem of predicting performance for arbitrary spectrum configurations. In the next chapter, we will come back to the spectrum-assignment problem and observe that having a way to predict performance helps us refine the spectrum-assignment optimization objective. In particular, we will see that using an optimization criterion expressed in terms of performance (instead of using proxies such as interference) is helpful in practice to find configurations that achieve different spectrum-sharing objectives (in terms of efficiency and fairness). Due to the complexity of the task, in this chapter and the next, we do not target fully decentralized methods, rather we consider scenarios where the APs belong to a single administrative entity (e.g., as is the case for enterprise networks). We discuss possible extensions to separate administrative entities when appropriate.

In general, 802.11 networks exhibit several performance intricacies due to complex interactions between the MAC and PHY layers; these intricacies manifest themselves in frequency, spatial and time domains. For example, using a wide bandwidth creates interference in the frequency domain, but using a narrow bandwidth increases packet-

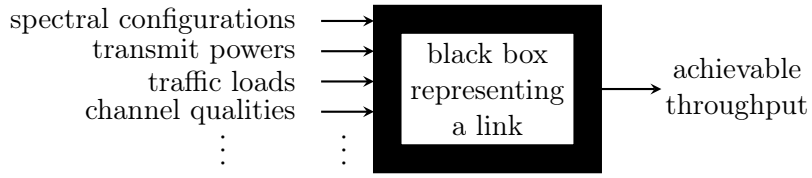

Figure 3.1 – Black box representation of a link. It takes various configuration and topological features related to a given link and its neighbors as inputs, and it outputs a throughput.

transmission times, which can create more interference in the time domain (due to the rate-anomaly problem of MAC layers based on CSMA/CA [HRBSD03][1]). In addition, as mentioned in Chapter 1, a narrow bandwidth packs more Watts per Hertz for a fixed transmit power, which improves the transmission range [CMM+08], but also increases interference in spatial domain.

Existing performance models for 802.11 networks, such as the one proposed by Bianchi [Bia00], usually adopt *explicit* and *bottom-up* approaches; they model the actual mechanics of the protocol (for example, the backoff procedure of the MAC layer in [Bia00]) in order to compute throughput figures. Unfortunately, it is difficult to use such models to capture heterogeneous physical-layer configurations, such as variable channel widths or variable transmit powers. In contrast, textbook models based on the SINR (signal to interference-plus-noise ratio) can be used to capture some of the phenomena occurring at the physical layer[2]. The main drawback of SINR models, however, is that they are not meant to capture 802.11 performance. In particular, they do not take the MAC layer into account and, as we will observe, they do not capture the actual performance of interfering links when CSMA/CA is employed.

In this chapter we observe that, if there is interest in predicting performance for arbitrary settings, it can be more efficient to learn *implicit* and *top-down* models directly from a set of observed measurements. We treat Wi-Fi links as black boxes with potentially unknown internal mechanics (see Figure 3.1). Such a black box takes some parameters as inputs (such as the spectral configurations of a Wi-Fi link and its neighbors, as well as topological features such as current measurements of channel qualities), and it outputs a throughput value. In this setting, our goal is to use a limited set of real-world measurements to find a function providing an accurate mapping between inputs and outputs, both of which might have never been previously observed. In particular, we do not attempt to seed a pre-existing model (such as SINR-based or Markov-based) with

[1] The rate-anomaly problem is an effect due to the CSMA/CA time-sharing mechanism. With this mechanism, contending stations have comparable probabilities to attempt a transmission. However, the stations using a low physical rate need more time to transmit their frames, which penalizes the other stations.

[2] To some extent, we used such a model when defining interference in Equation (2.1) of Chapter 2, as it is based on the model of [MSBA06] for partially overlapping channels, which quantifies interference in terms of interfering power in the SINR.

measurements; our approach is rather to *learn the model itself* from a limited set of measurements.

Constructing useful black boxes from measurements is difficult for two main reasons. First, they must capture a fair level of complexity; the cross-layer relationships between the various input parameters and the resulting throughput are usually complex, multi-modal, nonlinear and noisy. Second, it is in general infeasible to simply measure the link performance for each possible combination of inputs. Instead of conducting exhaustive measurements, we observe that a statistical representation of these black boxes can be learned by observing a limited number of input/output combinations. Using supervised machine learning techniques, it is possible to generalize the observations made on this limited subset of measurements and to still capture the complex relationships between the inputs. We build such implicit models using real-world measurements and we test them systematically, by asking them to predict the throughput for links and configurations that have never been observed during the initial measurement phase. We observe that our learned black boxes improve prediction accuracy over models based on the SINR (which is usually the preferred metric for allocating physical-layer resources such as spectrum or transmit power). In particular, we will see in the next chapter that such black boxes are instrumental (and more useful than SINR models) for capturing intricate interference patterns and finding efficient spectrum and transmit-power configurations.

The remainder of this chapter is organized as follows. In Section 3.2, we explain our approach through a few illustrative examples. In Section 3.3, we present our method to learn black box performance models. We evaluate the accuracy and generalization of our models in Section 3.4. We discuss the limitations of our approach in Section 3.5. Finally, we present some related work in Section 3.6 and summarize our main findings in Section 3.7.

3.2 Motivation

In this section, we first detail some of the complexities inherent to the problem of allocating variable-width spectrum chunks to wireless nodes. In particular, we show why the performance achieved by a link depends in a highly complex fashion on spectrum configurations adopted by this link and its neighbors. Second, we give an example where SINR-based models – the prevailing class of models for adapting PHY layer parameters – fail to capture the actual performance of 802.11 networks.

3.2.1 An Introductory Two-Link Example

Consider a simple setup with two Wi-Fi links l and k (which can be composed of two access points and two clients), shown in Figure 3.2(a). The two access points have

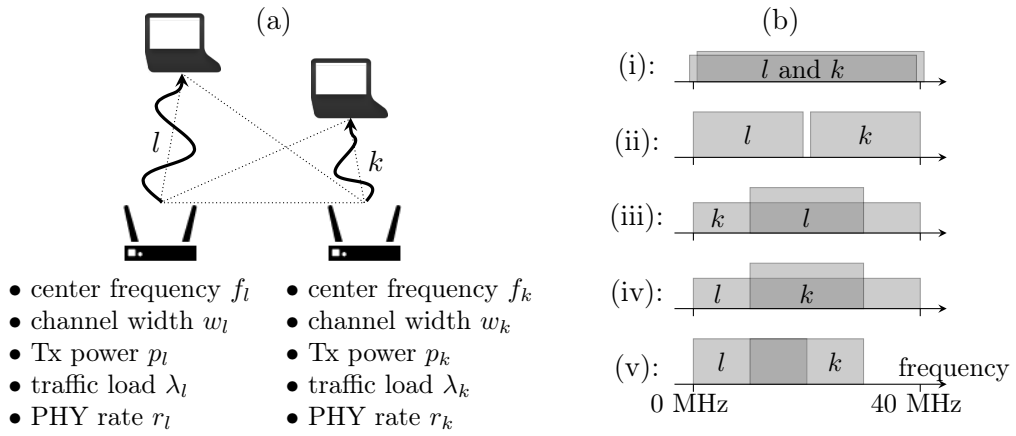

Figure 3.2 – Two interfering links (a), and some possible spectral configurations (b). In this chapter, we propose a method to learn models that can predict the performance of the links in such arbitrary configurations.

arbitrary traffic loads (e.g., due to arbitrary exogenous arrivals). Consider now our problem of allocating a combination of channel center-frequencies and channel widths to these two links. Assuming that the two possible channel widths are 20 MHz and 40 MHz (as in 802.11n), and that there is 40 MHz of total spectrum available, we list some possible allocations on Figure 3.2(b) (see [RSBC11] for an extensive treatment of such an example). Efficiently selecting configurations would clearly benefit from a model which could, for each possible combination of configurations, predict the throughput achievable by each link. A model explicitly designed for this task should take at least the following qualitative aspects into account:

- If the links are sufficiently far apart in space, the spectrum can be re-used, and both links can use a bandwidth of 40 MHz (allocation (i)).

- If the two links are physically close to each other, allocation (i) results in transmission arbitration in the time domain. Instead, it might be more efficient to opt for allocation (ii) to use orthogonal bands and to reduce the time spent in backoff. However, it is not clear *a priori* when allocation (ii) starts outperforming allocation (i). The actual performance improvements depend on the various channel gains, traffic loads, backoff times, adjacent-channel interference, etc.

- If link l has a poor channel quality, it can be beneficial to use allocation (iii), as using a narrow bandwidth increases the SNR of link l. Yet, using a narrow bandwidth has also the effect of increasing the time required to transmit a packet, which exacerbates the rate anomaly suffered by 802.11 [HRBSD03] and can potentially decrease the overall efficiency. In addition, allocation (iii) can also be attractive if link l has a low traffic load (and thus does not create much interference on k).

- A similar reasoning can be made for configuration (iv). This allocation can be the most efficient if, for instance, link k has a low traffic load. In this case, it does not need much spectrum and does not create much contention.

- Finally, depending on the traffic loads and the other surrounding networks, using allocation (v) with partially overlapping channels has the potential to create more efficient spectral re-use patterns [MSBA06], as it frees a part of the total available spectrum. However, in practice, it is in general difficult to predict when partially overlapping channels are beneficial [DGVB+11].

Clearly, as also noted in [RSBC11], performance depends in a highly complex way on the actual topology, channel qualities, spectral configurations, etc. It is especially hard to predict in quantitative terms when a given configuration outperfoms the others. The complexity is further increased if the nodes can adapt their transmit powers; although adapting transmit powers can potentially improve spectral re-use [BEKF07], it is rarely used in practice as the impact is difficult to predict [MPB07].

The difficulty of predicting performance in the presence of complex interference patterns limits the vast majority of works proposing models or optimizations for the PHY layer to using SINR-based models (see for instance the Algorithm presented in Chapter 2, as well as other works proposing spectrum assignment methods [RSBC11, MSBA06, MPB07]). However, SINR models are not meant to capture 802.11 performance and, as we will see now, they can fail to capture important performance patterns.

3.2.2 An Example Where SINR Models Are Inappropriate

We now consider a real testbed example, again with two interfering links l and k. In this case, both links use 20 MHz of channel bandwidth, with the same center frequency (i.e., we consider a simpler setup with no spectral separation). The two links send saturated UDP traffic with packets of 1500 B, and they both use 802.11n in the 5.8 GHz band (void of external interference), using the same 2×2 MIMO configuration. Link l has a fixed transmit power set to 12 dBm, and link k varies its transmit power from 3 dBm to 21 dBm. We measure the throughput obtained by l for two different pairs of links (l, k) on our indoor testbed (we give more details on our testbed in Appendix A). For comparison, we also compute the information-theoretic capacity c_l of link l as

$$c_l = \text{constant} \cdot \log_2 (1 + \text{SINR}_l), \tag{3.1}$$

where the constant factor accounts for the bandwidth and MIMO configuration, and where SINR_l denotes the SINR of link l. On such a two-link setup, the SINR is given by

$$\text{SINR}_l = \frac{P_{l \leftarrow l}}{N_0 + P_{l \leftarrow k}}, \tag{3.2}$$

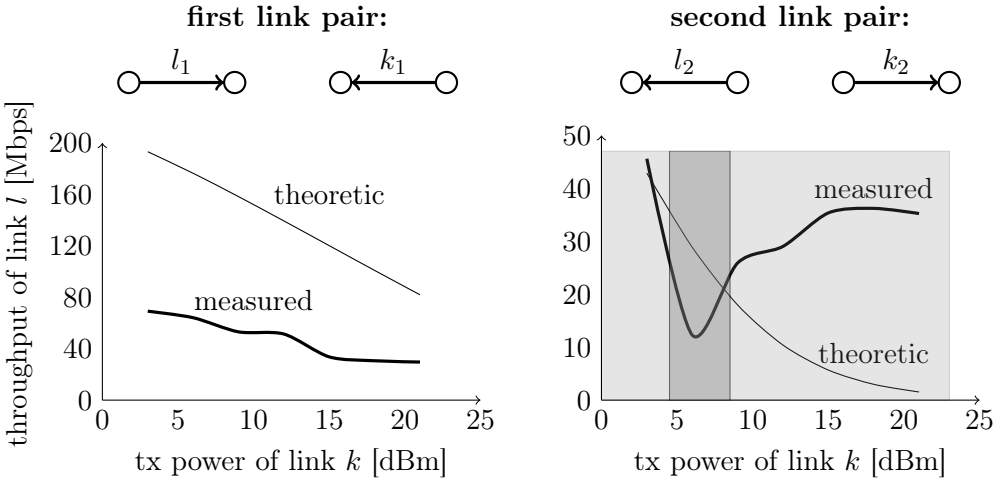

Figure 3.3 – Measured throughput and theoretical capacity of l, when k varies its transmit power. The results are shown for two different pairs of links (l_1, k_1) and (l_2, k_2) from our testbed.

where $P_{l \leftarrow l}$ (respectively, $P_{l \leftarrow k}$) denotes the received power at the receiver of l (as measured by our NICs) from the transmitter of l (respectively, from the transmitter of k), and where N_0 is the background noise (also reported by our NICs).

We show both the measured throughput and the theoretic capacity for the two link pairs in Figure 3.3. The (schematized) topologies are shown at the top of the figure. For the first link pair, the throughput obtained by l decreases by about 50% when k increases its transmit power. This is due to an increased likelihood of collision at l's receiver and carrier-sensing activation at l's transmitter, as k increases its effective interference range. This qualitative trend is captured by the theoretical capacity, which decreases when $P_{l \leftarrow k}$ increases. Note that, in this case, the magnitude of the theoretical capacity is much higher than the actual throughput of the link. This is expected, because the the theoretical capacity does not account for the overhead of the MAC layer. In fact, it seems that in this case, the theoretical capacity can provide a reasonably accurate characterization of the throughput, if it is scaled appropriately.

However, the situation is very different (and more surprising at first sight) for the second link pair. Here, we can decompose the measured performance in three distinct regimes (represented by three shaded regions in the figure). When k's transmit power is low, the links are nearly independent and l suffers little interference from k. When k's transmit power grows to intermediate values, k starts interfering with l. In this case, l carrier-senses k, and interference mitigation is done in the time domain via CSMA/CA. However, a closer inspection of packets reveals that *link k itself* does not have a good channel quality (as it uses only an intermediate transmit power), which forces it to use relatively robust (and slow) modulations. As a result, in this intermediate phase,

k consumes a significant fraction of time transmitting its packets, which reduces l's throughput (due to the rate anomaly). Finally, when k uses a large transmit power, it also uses faster modulations, which has the apparently paradoxical effect of increasing l's throughput.

In this second example, the information-theoretic formulation for the capacity does not capture all these cross-layer and multi-modal effects that are specific to the MAC layer of the 802.11 protocol. Instead, it shows a monotonic dependency on transmit power stemming from its monotonic relationship with the SINR. Overall, the information-theoretic capacity treats the case of Gaussian channels subject to *constant* and *white noise* interference, with no intent to model 802.11 networks. In fact, in the cases where a time-sharing scheme such as CSMA/CA is employed, links often have the opportunity to transmit alone on the channel, thus without observing any interference at all during their transmission[3].

From these two simple examples, we observe that the theoretical capacity (i) might have a significantly different magnitude than the observed throughput, and (ii) might not capture the non-monotonic, multi-modal complex behaviors due to cross-layer interactions occurring when links are interfering (note that this observation holds for *any* model that is monotonic in the SINR). Despite these problems, and despite the low predictive value on throughput given by the wireless community to SINR models, these models are still currently the models of choice for allocating resources at the PHY layer, due to their generality. By adapting judiciously the power values in the SINR Equation (3.2), it is possible to use variable transmit powers (as we just did), but also partially overlapping channels [MSBA06] and variable bandwidths [RSBC11] as inputs of SINR models. In addition, a large body of theoretic literature on optimal resource allocation also relies on SINR models in various contexts [EPT07, HBH06, BMS11, QZC10, HP12, MPB07, KBC+07].

3.3 Learning Performance Models

3.3.1 Approach

In the previous section, we observed that, even in the best cases where SINR-based models correctly capture the correct trend of the throughput, they can still suffer from a scaling problem, whereby they predict a capacity that has a significantly different magnitude than the observed throughput. Therefore, a natural step for improving the accuracy is to seed (or *fit*) some parameters in SINR-based models (for instance, a factor controlling the magnitude of the prediction) to the observations of actual measurements. The approach of seeding a model with measurements has been taken in [RMR+06, LQZ+08, RSBC11] and others (see Section 3.6 for a discussion).

[3]This is also the reason the actual throughput might be largely above the predicted capacity.

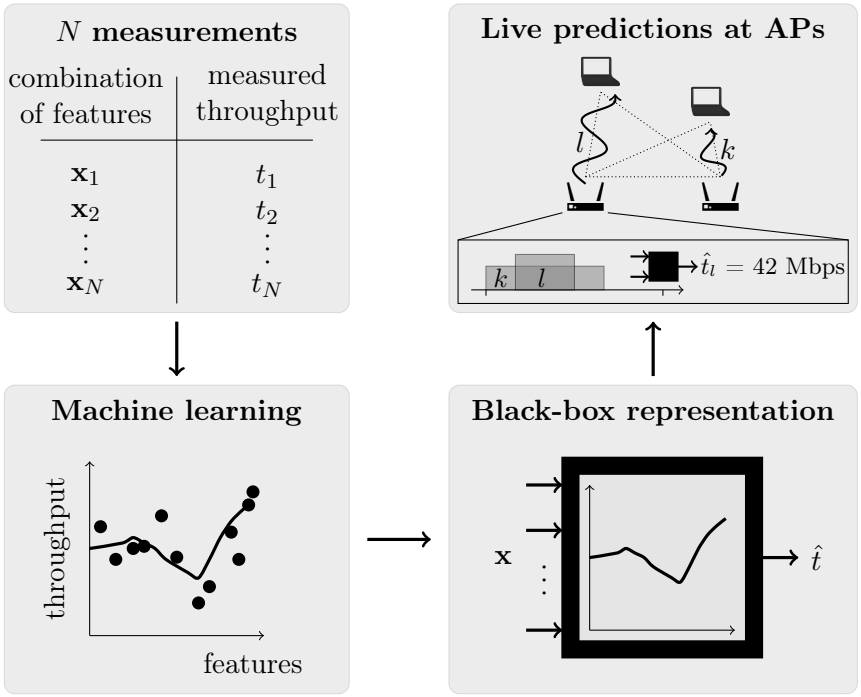

Figure 3.4 – Overview of our learning setup.

In this chapter, we also use an initial measurement phase. However, we take a step back and treat the problem of fitting a model to measurement as a general regression problem. Contrary to prior approaches, we do not try to fit or to seed a previously existing model (such as, for instance, SINR-based or Markov-based). Instead, we rely on the measurements in order to directly learn the model *itself*, using supervised machine learning. Our main reason for doing so is that supervised machine learning proposes rich models for regression (in terms of the complexity of the dependencies that they can capture), which have efficient fitting (or learning) procedures. Our overall approach is summarized in Figure 3.4, and it consists of four main steps:

1. **Measurement phase:** This phase consists in performing N short-duration controlled experiments. Considering again the black-box representation of Figure 3.1 (although generalized for more than two links), each experiment consists in measuring the throughput of a given link l, for one particular combination of inputs (which we call *features*). This phase is relatively short; we observe in Section 3.4.4 that it is possible to "learn" our entire indoor testbed of 22 nodes with reasonable accuracy in less than 6 hours.

2. **Learning phase:** Once the measurements are obtained, this phase consists in finding a function that maps the features to observed throughputs. The function should be d-dimensional if there are d features, and it should approximate the throughput well on the measured data points. However, to be useful, it must

not overfit existing measurements that are intrinsically noisy. Instead, it should generalize to unseen combinations of input features (which can potentially relate to unseen nodes and links). Supervised machine-learning provides the natural tools to address this challenge.

3. **Black-box representation:** Once a function has been found that is both accurate and generalizable, we can discard the measurements and use the function itself to obtain throughput predictions.

4. **Usage in resource allocation (Chapter 4):** This step is the operational phase. If we target distributed resource-allocation, the black box can be used as an oracle by the access points themselves. For instance, in the two-links example of Figure 3.2, if l collaborates with k to acquire some of the features at a certain time instant, it can produce throughput predictions for various configurations (and thus choose efficient configurations without probing).

Notably, we observe in Section 3.4.3 that learned models continue to be useful in new or unseen environments, and that the training procedure does not need to be repeated when new wireless links come and go. We detail our procedure in the remainder of this section.

3.3.2 Feature Selection

Consider a link l, for which we want to predict saturated throughput (i.e., under saturated traffic load[4]) for arbitrary spectrum and transmit-power configurations, given a set $\mathcal{N}_l$ of K neighboring links with arbitrary conditions, configurations and traffic loads. Such a scenario is shown in Figure 3.5 for $K = 2$. The features must include factors that affect the performance and are measurable by the transmitter of l and its immediate neighbors. We selected the following list of features, because they all have an immediate effect on performance (see e.g., [BEKF07, CMM+08, MSBA06, HRBSD03]):

- The power received by each node of l from every transmitting node, and the power received by every other node, from the transmitter of l. These quantities are denoted $P_1, \ldots, P_{11}$ in Figure 3.5 (assuming downlink traffic, from the APs to their clients). They depend on the transmit powers and the various channel gains, and they can be easily measured online by commodity hardware using the RSSI (received signal strength indicator). There are $5K + 1$ such power quantities in general.

- The channel widths used by l and by the links in $\mathcal{N}_l$. There are $K + 1$ such values.

[4]We target saturated throughput because it is the maximum achievable throughput in a given configuration. In particular, we assume that if throughput t is achievable in a given setting, then any throughput $t' < t$ is also achievable in the same setting. Note that this concerns the final predicted throughput of link l, but our models accounts for neighboring links with arbitrary traffic loads.

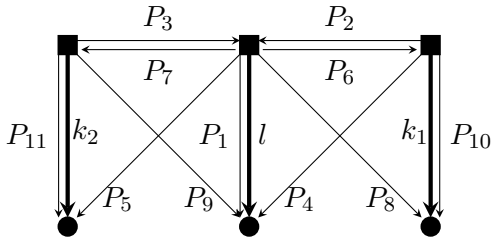

Figure 3.5 – Throughput prediction setting with a link l and two neighboring links $\mathcal{N}_l = \{k_1, k_2\}$. We wish to predict the throughput that l could potentially obtain, given the various received powers, as well as the physical rates, channel widths, center frequencies, and traffic loads on k_1 and k_2.

- The spectral separations between the center frequency used by l, and the center frequencies used by all the other links in $\mathcal{N}_l$. There are K such values.
- The K average traffic loads of the links in $\mathcal{N}_l$.
- The physical rates (determined from the MCS index of 802.11n) used on each link in $\mathcal{N}_l$. These are used as features, because it is known that the modulation and coding employed for transmission have a strong effect on the performance of neighboring links [HRBSD03]. There are again K such values.

Adding up the above-mentioned features, we have access to $d := 9K + 2$ quantities to estimate the throughput that link l can achieve in the presence of K interferers. Note that this list of features is not an exhaustive list of the factors affecting performance that can be known or measured by the APs. For instance, we could make it more complete by including the packet sizes, higher order statistics to describe the traffic loads of interferers (instead of the mean only), or more detailed PHY layer information (e.g., to capture non-802.11 interference, multipath effects or frequency-selective fading[5]). Including more features could further increase the predictive power and generality of the learned models. However, the features selected here already enable us to build useful models and have the advantage of being easy to acquire with commodity hardware.

3.3.3 Measurement Phase

The initial measurement phase consists of N measurements with different combinations of features. Some of the features can be directly controlled (namely, the channel widths, spectral separations and traffic loads) and others cannot (the received powers depend both on the transmit powers and channel gains, and the physical rates depend on the

[5] [HHSW10] shows that considering channel measurements at the OFDM subcarrier level provides substantially more information on channel quality than crude RSSI. Unfortunately, such measurements are not available on our wireless cards.

auto-rate mechanism used by the APs). Each of the N measurements consists of two sub-experiments. We first perform an experiment during which link l is silent, in order to obtain a corresponding vector $\mathbf{x} \in \mathbb{R}^d$ of features (some of which are controlled, others are measured). We then repeat the experiment with l sending saturated traffic, and measure its throughput t_l. Our goal is to expose the learning procedure to as wide a variety of situations as possible. To this end, we apply the following sampling procedure for each of the N data points.

We use 802.11n for all measurements in this chapter. We start by selecting a link l uniformly at random among all the links formed by all the nodes of the network. We then sample K random interfering links, where K itself is randomly drawn between 0 and max_K, and max_K denotes a fixed upper bound on K. For l and the K links in $\mathcal{N}_l$, we sample transmit powers and spectral configurations uniformly at random from the set of configurations that do produce some interference (i.e., such that each link in $\mathcal{N}_l$ uses a band at least adjacent or partially overlapping with l). Finally, for each link k in $\mathcal{N}_l$, we sample a traffic load in the interval $(0, h(w_k)/K]$, where $h(w_k)$ is equal to the maximum throughput achievable on an isolated link using bandwidth w_k. We take $h(20 \text{ MHz}) = 80$ Mbps and $h(40 \text{ MHz}) = 130$ Mbps in our training procedure, in line with the maximum achievable throughput of our 802.11n nodes. Our goal is to predict performance for *arbitrary* interfering loads, and sampling the loads in this way enables us to expose the learning procedure to different environments with both light and heavy contention. In particular, we measured that the offered loads of the nodes in $\mathcal{N}_l$ was above capacity (i.e., saturated) in about 54% of the experiments (mainly due to inter-neighbors interference). The remaining experiments consist of non-saturated conditions.

Once the configurations are chosen, we perform the first experiment with only the K interfering links active. During this experiment, we measure the average physical rates used by each of the K links in $\mathcal{N}_l$, and we group all the above-mentioned features in a vector $\mathbf{x}_i$. In order to vary K between 0 and max_K while keeping features vectors of fixed dimension d, we append $9(max\_K - K)$ default "flag" values to $\mathbf{x}_i$, using -110 dBm for all the power values, and setting all the remaining features to zero[6]. We then perform the second experiment in the same conditions, but with link l sending saturated traffic, and we measure its achieved throughput; this represents our target value. Each of the two sub-experiments constituting each of the N data points needs only to last a few seconds (in order to measure average physical rates and throughput), and the whole procedure is easily automated.

[6]The current number of interfering links K is thus an *implicit* feature, encoded by the presence/absence of flag values.

3.3.4 Learning

Let us write $\{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\} \subset \mathbb{R}^d \times \mathbb{R}$ for our set of measurements. Our goal is now to find a function $f : \mathbb{R}^d \to \mathbb{R}$ that maps $\mathbf{x}_i$ to a value close to the target t_i for each measurement i. This is an instance of a regression problem, and we learn the function f directly from the observed data.

Several techniques exist for learning functions from data. We do not go into the details, but rather we present the overall characteristics of the different approaches that we consider for this task, and we refer the reader to reference textbooks for details (see e.g., [Bis06, HTF08]). We consider the following regression techniques.

- **Regression tree:** This technique fits a binary tree to the data. Each feature vector corresponds to a path in the tree (from the root to a leaf), and each leaf corresponds to a (discretized) throughput value. The resulting model is elegant, because it yields predictions that can be evaluated by a sequence of "if-else" clauses on the features[7]. However, fitting an optimal tree is a NP-hard problem, and the obtained trees are usually sub-optimal. It also produces hard decision thresholds, which can affect generalization and accuracy.

- **Gradient Boosted Regression Trees (GBRT):** This technique combines the predictions of M regression trees. Given a feature vector $\mathbf{x}$, the throughput is predicted as

$$\hat{t} = f(\mathbf{x}) = \sum_{m=1}^{M} \pi_m h_m(\mathbf{x}).$$

  In the above expression, $h_m(\mathbf{x})$ denotes the prediction of the m-th tree, and the π_m's are the weighting coefficients (learned with gradient boosting [HTF08]). The number of trees M, as well as their depth, is obtained by cross-validation (see [Bis06]). Using several trees has the potential to largely improve the predictive power, compared to a single tree, however as we will see, it might still be subject to potential overfitting.

- **Support Vector Regression (SVR):** For a feature vector $\mathbf{x}$, this method outputs a predicted throughput given by

$$\hat{t} = f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b,$$

  where the α_i's and b are the fitted parameters (obtained by solving a convex minimization problem that accounts both for regression error and overfitting). The function $k(\cdot, \cdot)$ is a so-called *kernel function*. We generate our own SVR models using a common kernel function known as the radial basis function (RBF), specified by

[7] For instance, on a simplistic tree of depth 2, a regression path could look like: "if received power $\leq X$ and frequency offset $> Y$, then predict Z".

$k(\mathbf{x}_i, \mathbf{x}) = \exp(-\gamma\|\mathbf{x} - \mathbf{x}_i\|^2)$, where γ is a parameter obtained by cross-validation. Usually, most of the α_i's are equal to zero, and therefore SVR requires only a fraction of the initial measurements to be stored. Furthermore, this technique has a high descriptive power, and it can efficiently prevent overfitting (see [SS04] for more details).

- **SINR-based model:** As a comparison to pure machine-learning techniques, we also fit SINR-based models to our measurements. In particular, we consider the following variant to Equation (3.1) for computing the theoretical capacity c_l of link l:

$$c_l = \Gamma \cdot w_l \cdot \log_2(1 + \mathrm{SINR}_l), \tag{3.3}$$

where Γ is a constant that is fitted to measurements (using minimization of least square error), in order to correct for the scaling problem mentioned in Section 3.2. In addition, we also use the approach of Chapter 2 proposed in [MSBA06] in order to account for partially overlapping channels; specifically, we scale each power value appearing in the SINR Equation (3.2) by an appropriate value that accounts for the spectral overlap, assuming perfect bandpass filters. To the best of our knowledge, these models are the only existing models that can produce performance predictions at such levels of generality (i.e., taking into account arbitrary spectral configurations with variable widths and variable transmit powers).

3.4 Accuracy of Performance Predictions

In this section, we evaluate the accuracy and generalization of the different learning strategies in various conditions.

3.4.1 Experimental Setup and Methodology

Experiment Setup

We use 22 nodes of the testbed described in Appendix A, which are shown in Figure 3.6. The experiments are conducted using 802.11n and the default Minstrel autorate algorithm. We employ 20 and 40 MHz channel widths, 2×2 MIMO, and 10 different transmit power values in the set $\{3\text{dBm}, 5\text{dBm}\ , \ldots, 21\text{dBm}\}$. We use the 5.735-5.835 GHz band, which comprises a total of 100 MHz of spectrum.

Methodology

Our objective is to test the models with unknown combinations of features. As such, we only predict throughputs for data points that do not appear in the N measurements used for learning (or training). To this end, we always split our total set of measurements into

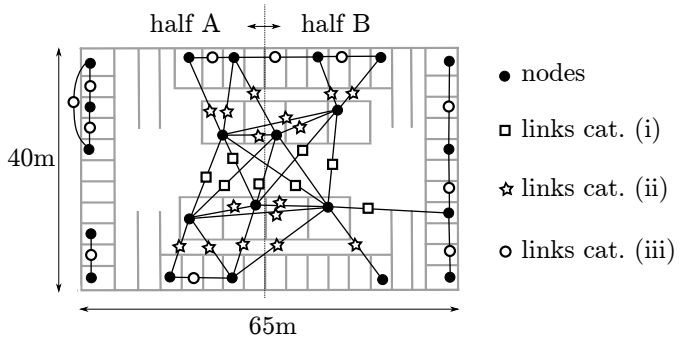

Figure 3.6 – Layout of the 22-nodes used for the experiments. We also show the different link categories and the two halves of the testbed used in the experiments of Section 3.4.3.

a *training set* and a *test set*. The training set consists in the actual N measurements used for learning the models and their parameters, whereas the test set is used only once, for measuring the final accuracy.

We gathered a trace of 8900 measurements[8], with $max\_K = 3$. This set is voluntarily larger than what is actually needed, in order to enable us to test the effect of the number of measurements N on the models quality.

To evaluate the goodness of the regressions for the various models, we use the *coefficient of determination*[9] R^2. If we have a test set with n target throughput measurements $t_1, \ldots, t_n$ and a given model predicts the throughputs $\hat{t}_1, \ldots, \hat{t}_n$, then the coefficient of determination is given by

$$R^2 = 1 - \frac{\sum_i (t_i - \hat{t}_i)^2}{\sum_i (t_i - \bar{t})^2},$$

where $\bar{t}$ is the average throughput, given by $\bar{t} = \frac{1}{n}\sum_i t_i$. Concretely, the R^2-score quantifies how well a predictor does, compared to the simplest baseline strategy that always predicts the mean throughput. It is equal to 1 if there is a perfect match between predicted and measured throughputs. Whereas, it can be negative if a strategy would do better by always predicting the mean throughput. In addition to the R^2-score, we also compute the root mean squared error (RMSE), defined as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_i (t_i - \hat{t}_i)^2}.$$

We used the Python machine learning package `scikit-learn` [PVGa11] to learn the various models.

[8] Our dataset is publicly available: `http://www.hrzn.ch/data/lw-data.zip`
[9] `http://en.wikipedia.org/wiki/Coefficient_of_determination`

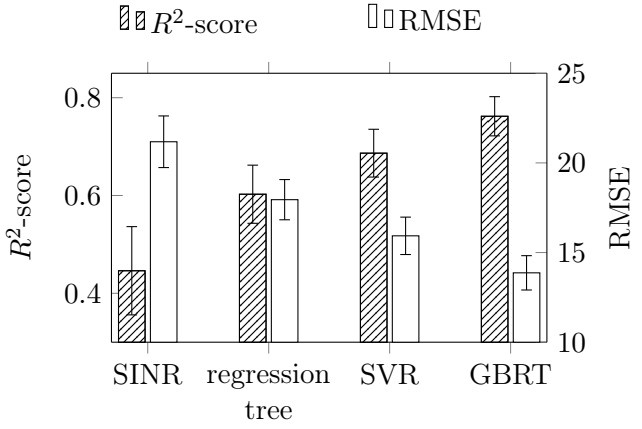

Figure 3.7 – Summary of prediction performance for various models.

3.4.2 Prediction Accuracy

In order to compare the accuracy of the different classes of models, we perform 50 consecutive splits of our measurements in training and test sets (i.e., 50-fold cross validation [Bis06]). For each split, we evaluate the R^2-score and RMSE, and we show the average and standard deviations in Figure 3.7 for each class of model. In addition, we also show the detailed distribution of prediction errors in Figure 3.8 for models based on SVR and GBRT.

It appears clearly that the learned models, in particular those based on SVR and GBRT, perform significantly better than the SINR-based models. In terms of R^2-score, learned SVR and GBRT models improve the prediction accuracy by 54% and 71%, respectively, compared to SINR models (which, we recall, are the predominant class of models capturing such things as overlapping channels). In terms of error distribution, 90% of the errors made by learned models are between -25 Mbps and 25 Mbps, whereas 90% of the errors made by SINR-based models are between -35 Mbps and 36 Mbps. The higher accuracy of the learned models is remarkable; it demonstrates that, as far as performance prediction is concerned, learning abstract models that come from the machine learning domain – without any knowledge of 802.11 networks – can be more efficient than trying to fit (or seed) pre-existing SINR models.

In order to visualize the actual predictions in detail, we also show a scatter plot of the predicted throughputs, against the actual measured throughputs, in Figure 3.9. We show both the predictions obtained by the SINR model and the learned SVR model (for reasons that will be clear soon, we show the results for SVR instead of GBRT, even though GBRT performs better in this particular setting). On these plots, the closer the points are to the diagonal, the better the prediction accuracy. Clearly, here too, SVR models perform much better and produce fewer outlying predictions than SINR models.

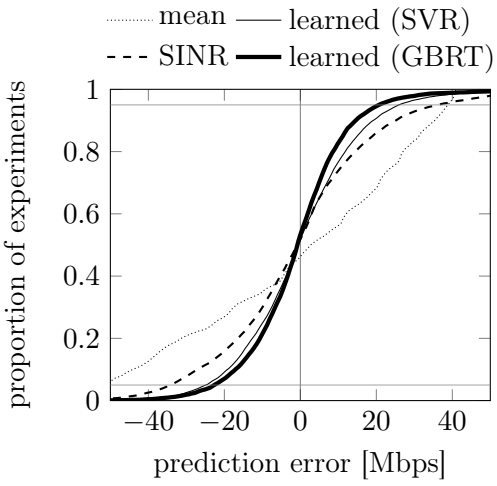

Figure 3.8 – Empirical CDF of prediction errors. The "mean" model represents the errors obtained by a baseline predictor that always predicts the mean throughput of the training set.

Note that obtaining *perfect* predictions is impossible here, because both the measured features and the throughput are highly noisy variables, measured with commodity hardware. To illustrate this, we examine in more detail the features corresponding to the worst prediction obtained by both models (shown by an arrow on the plots – incidentally, this is the same point for both models). This point corresponds to a link l subject to no (controlled) interference (i.e., $K = 0$), with an apparently good channel quality (the measured RSSI is -59 dBm in this case), and using a bandwidth of 40 MHz, supposedly yielding the largest capacity. Yet, despite these features, the measured throughput was low. We can only speculate about the causes for this discrepancy; it could have been an especially unfavorable conjunction of high noise, both in the measurements of channel quality (which might have been worse than measured) and/or obtained throughput (which might have been temporarily altered by some higher layers' factors). In any case, this example, although relatively extreme, illustrates the limits of throughput predictability with imperfect information.

3.4.3 Generalization

Due to the split between the training set and test set, the previous results address cases where throughputs predictions are produced for unseen combinations of *features*. We now attempt to push our models further, by making them predict throughputs for unseen *links*, potentially belonging to different environments.

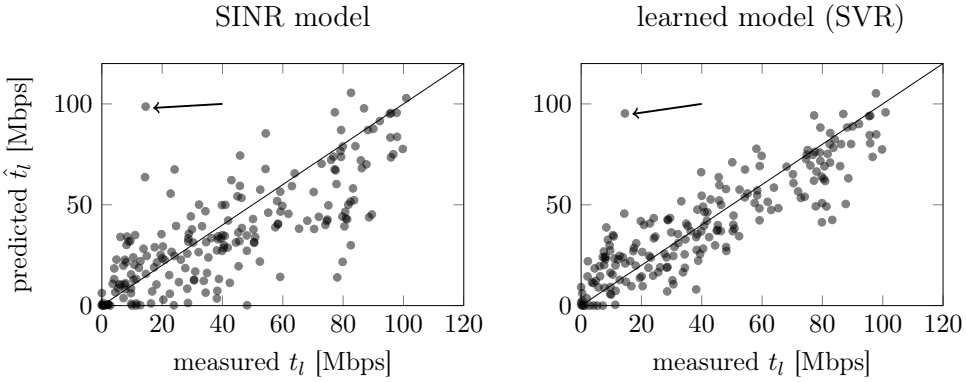

Figure 3.9 – Predicted versus measured throughput, for SINR and a learned model, on a test set of 200 points.

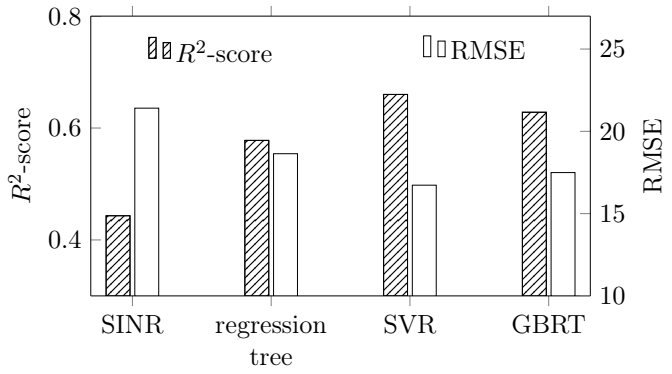

Figure 3.10 – Prediction accuracy on links that have never been observed during the learning phase.

Predictions for Unseen Links

For each possible link l, we remove both l and its reverse link (obtained by inverting the transmitter and the receiver of l) from the training set. We then produce throughput predictions for each data point that contains l (or its reverse link), and show the results in Figure 3.10. Compared with Figure 3.7, some models (especially those based on regression trees) see their accuracy slightly decreased. However, the models learned with SVR still perform remarkably well; in terms of R^2-score, their accuracy is reduced by less than 4%, and they still improve the accuracy by 49%, compared to SINR-based models.

Different Environments

We now manually divide the links present in our trace in three distinct categories, depending on the type of attenuation that they experience. The categories are shown in Figure 3.6, and they correspond to the following link division: (i) links that traverse mostly empty space, (ii) links that traverse sparsely spaced walls and (iii) links that traverse densely spaced walls.

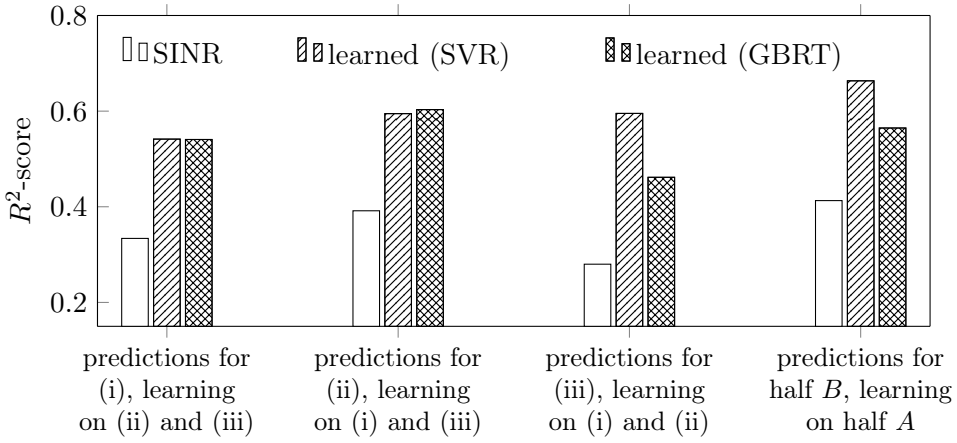

Figure 3.11 – Prediction accuracy for never-observed groups of links. Even in the difficult cases, learned models largely outperform SINR models.

For each category, we remove all the links (and their respective reverse links) belonging to this category from the training set. We then build the test set so as to predict throughput for links belonging *only* to this category. The goal of this experiment is to test prediction accuracy in the worst possible conditions: each model is learned on links that operate in conditions radically different than the conditions prevailing during the actual predictions. In addition to the three link categories (i)-(iii), we also split our testbed in two halves (also shown in Figure 3.6). Here too, we test the prediction accuracy when learning the models on the first half A of the testbed, and testing them on the second half B. The resulting accuracies are shown in Figure 3.11.

Even in these difficult cases, the learned models based on SVR show a graceful degradation and keep a relatively high accuracy (with R^2-scores always larger than 0.54). When producing predictions for the half B with models learned on the half A, models based on SVR even obtain similar accuracies as when learning using the full testbed. This enables us to draw some conclusion on the extent to which our method generalizes. Even when learning models on a different part of the testbed, or using radically different links, abstract models based on machine learning still have far more predictive power than measurement-seeded models based on SINR. Note that such an ability to generalize to drastically different conditions constitutes a promising avenue for extending the applicability of learned models to networks that do not belong to a single administrative entity, and where obtaining the initial measurements is difficult.

3.4.4 How Much Learning Is Needed?

Finally, we measure the accuracy as a function of the training set size N. For different values of N, we learn models using N experiments sampled at random from our complete experiment trace. We then predict the throughput for all the other experiments, and

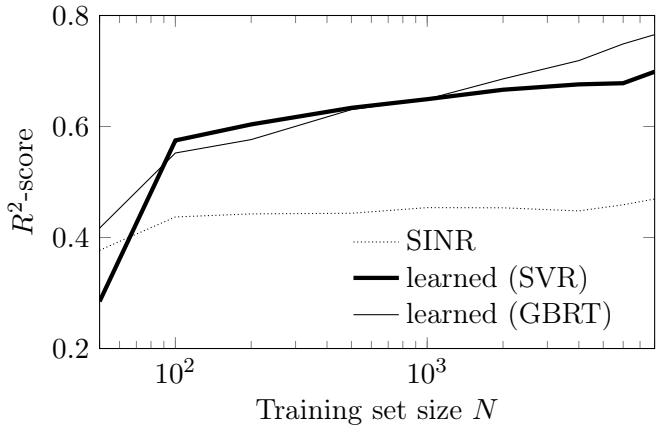

Figure 3.12 – Prediction accuracy as a function of number of measurements N (note the logarithmic scale of the x-axis).

measure the R^2-score. The results are shown in Figure 3.12. Using $N = 100$ training experiments is enough to obtain better accuracy than SINR models, and $N = 1000$ experiments already yield good predictive accuracies. If each experiment consists in 10 seconds of measurements (which is the duration that we employed), this means that an efficient performance model for an entire building-scale network such as ours can be learned in less than 6 hours.

3.5 Limitations and Discussion

We evaluate our learned models in static conditions, a setting for which throughput prediction is somewhat easier (compared to say, high mobility with fast fading, short channel coherence times, etc). This is because, in this thesis, we deliberately restrict ourselves to using features easily accessible on commodity hardware (e.g., RSSI measurements). Such features are only meaningful on relatively coarse timescales (typically seconds) and cannot capture such fast-changing phenomena. In this sense, our learned models suffer from the same timescale limitations as any model (including SINR) that uses similar measurements. Extending such a learning framework to using features that operate at shorter timescales is an interesting avenue for future work.

Notably, using features that operate at relatively coarse timescales already enables our learned models to be useful in practice. In particular, it fits well the needs of the algorithm described in Chapter 4, which uses such a black box in order to re-evaluate the spectrum configuration of each AP every few minutes on average[10].

[10] At faster timescales, the overhead of switching to different spectrum bands on commodity hardware would exceed the benefits of employing efficient spectrum allocations.

3.6 Related Work

802.11 networks have been extensively studied in the literature. The performance and fairness of the MAC layer and its CSMA/CA procedure are well captured by several theoretical models [Bia00, DDT07, MDL07, DDT09]. Unfortunately, these models do not take PHY layer diversity into account and are agnostic to spectral configurations – we can therefore not use them to select spectral configurations. Several papers also propose measurement-based approaches to model performance and interference in 802.11 networks. In particular, [RMR+06, KGD07, QZW+07, LQZ+08] propose to conduct initial measurement campaigns (where the number of measurements is typically a function of the number of nodes present in the network), in order to fit various performance models. [RMR+06] fits a model based on the SINR in order to estimate the packet loss probability, whereas [KGD07], [QZW+07] and [LQZ+08] use measurements-based Markov chain models to predict the capacity and/or interference of 802.11 networks. Here too, the models are agnostic to the spectral configurations of the nodes, and they are designed to work when the links operate on a same channel, with a fixed channel width. In this chapter, we also use an initial measurement phase. However, the important difference with other approaches is that we are not constrained to any particular model, rather we employ supervised machine learning to obtain *any* suitable model that captures both PHY and MAC layer complexities together. This enables us to address more general interference scenarios, where nodes employ CSMA/CA with arbitrary traffic loads and heterogeneous spectral configurations.

[HHSW10] observes that measurements at the OFDM subcarrier level largely improves the accuracy of performance prediction. Unfortunately, the method does not take interference into account, and it cannot be used to make performance predictions when several links operate at the same time.

Finally, a few papers propose to use machine learning techniques in the context of wireless networks. [DH09] discusses the use of k-NN for link adaptation and [CHSO07] proposes an architecture for cognitive radios with learning abilities. However, these works do not attempt to predict performance. To the best of our knowledge, ours is the first work using machine learning to predict actual Wi-Fi performance.

3.7 Summary

We have investigated and validated a new approach for predicting the performance of Wi-Fi networks. Rather than manually fitting complex models to capture complex dependencies, we have shown that it is possible to directly learn the models *themselves*, from a limited set of observed measurements. This approach bypasses the usual modeling process, which requires both deep knowledge and tedious analysis, and yet often yields models that are either too restricted or too inaccurate. We observe that abstract black-box

models built using supervised machine-learning techniques – without any deep knowledge of the complex interference dynamics of 802.11 networks – can largely outperform the dominant class of SINR-based models. Further, we have shown that these models still work when they have to predict performance for links that have never been observed during the learning phase. In addition, their performance degrades gracefully when predicting performance in different environments that have not been observed during learning (such as free space and dense walls). This observation offers interesting perspectives for extending the usage of learned models to different administrative entities.

In the next chapter, we will use one such model as an oracle in a new distributed utility-optimal resource allocation algorithm. We will observe that this algorithm adapts well to various optimization criteria, and that our learned model is instrumental for achieving good performance.

4 A Utility-Optimal Collaborative Spectrum-Assignment Algorithm

4.1 Introduction

In this chapter, we come back to the spectrum-assignment problem. We take a direct approach in our optimization procedure; instead of optimizing proxies, such as interference and spectrum usage (as in Chapter 2, where we minimize the function given by Equation (2.3)), we use the models built in Chapter 3 in order to optimize a function of the final performance. The algorithm proposed in this chapter allocates channel center-frequencies, bandwidths and the access points' transmit powers. Adapting the transmit power is notably challenging, as it is difficult to predict the effect that it has on performance (see [MPB07] and Section 3.2.2 of the previous chapter). Allocating variable transmit powers can be seen as a form of "generalized" spectrum assignment task, where the access is modulated not only in frequency, but also in space.

Our goal is to design an algorithm that is *utility-optimal*, in the sense that it maximizes a sum of user-defined *utility functions* that are functions of the final performance obtained in any given configuration. This setting relies on the "direct approach" of performance prediction, and it enables our algorithm to optimize an arbitrary balance between efficiency and fairness (where the balance is defined in terms of the utility achieved on each link). In this context, our algorithm uses some collaboration between neighboring APs for two reasons. First, when predicting the achievable performance for a given configuration, the APs need to obtain some information about their neighbors. This information includes the current spectrum assignment, the current traffic load, and the various channel gains that each neighbor observes from the nodes in the BSS of the querying AP. These constitute some of the inputs of the performance model. Second, in order to target utility-optimal configurations, the APs also need to weigh the effect (in terms of utility) that each potential configuration is predicted to have on its neighbors.

In order to implement collaboration in a distributed setting, we devise a neighbor-discovery mechanism, whereby the APs meet on pre-defined channels and exchange their

public (wired) IP address. When the APs know the IP address of their neighbors, they use their backbone connection for the actual collaboration. We implement our complete algorithm (with neighbor discovery) and observe on our testbed that the optimization objective, defined by the utility functions employed by the APs, effectively drives the spectrum allocations in the desired regimes. Furthermore, we also show that the algorithm naturally load balances the amount of consumed spectrum (in frequency and spatial domains), as a function of the traffic loads and fairness objectives of the various APs.

We organize the remainder of this chapter as follows. In Section 4.2, we first present our model and notations. We then present the algorithm itself in Section 4.3, and we describe its implementation in Section 4.4. We show the results of testbed experiments in Section 4.5. Finally, we summarize the chapter in Section 4.6.

4.2 Model

We use a BSS-centric model similar to the one presented in Section 2.2. There are however some slight differences between the two models, and we therefore present all the notations used in this chapter for completeness. We consider a set of L links and a set $\mathcal{A}$ of access points (APs). Here again, each link l is composed of one transmitter and one receiver that operate on the same frequency band (i.e., using the same channel center-frequency and bandwidth). In this chapter we focus on downlink traffic, in line with asymmetric domestic Internet connections, where downlink traffic largely dominates [SdDF+11]. As such, we use the terms transmitter and AP interchangeably. Further, we write $l \in A$ if AP A is serving the receiver of link l. Let $\mathcal{F}, \mathcal{B}$ and $\mathcal{P}$ denote the finite sets of available channel center-frequencies, bandwidths and transmit powers, respectively. For convenience, we define $\mathcal{C} := \mathcal{F} \times \mathcal{B} \times \mathcal{P}$ to be the set of possible configurations. For a link l, we denote by $f_l \in \mathcal{F}$ its channel center-frequency, $b_l \in \mathcal{B}$ its bandwidth, and $p_l \in \mathcal{P}$ the transmit power used by the transmitter of l. All the links sharing a common AP must use the same configuration of center frequency, bandwidth and transmit power. We thus define $\mathcal{S} \subseteq \mathcal{C}^{|\mathcal{A}|}$ the corresponding set of feasible configurations that satisfy this constraint. We write $c_A \in \mathcal{C}$ for the spectrum configuration of AP A. We extend this notation to sets, and we write $c_\mathcal{D} \in \mathcal{C}^{|\mathcal{D}|}$ for the joint configuration of a set $\mathcal{D} \subseteq \mathcal{A}$ of APs.

We say that two APs are *neighbors* if they are close enough to interfere with each other (i.e., if there exists a joint configuration $\langle f_l, b_l, p_l, f_k, b_k, p_k \rangle$ such that some of the links served by the two APs interfere with each other when operating with this configuration). We write $\mathcal{N}_A$ for the set of neighbors of AP A, and we use the terms neighbor and interferer interchangeably. This neighborhood relationship defines an undirected neighbor graph $G(V, E)$, where $V = \mathcal{A}$ and $(A, B) \in E$ if and only if APs A and B interfere. Here too, our interference definition implies mutual neighborhood relationships, i.e., $B \in \mathcal{N}_A \Leftrightarrow A \in \mathcal{N}_B$ for any two APs A and B.

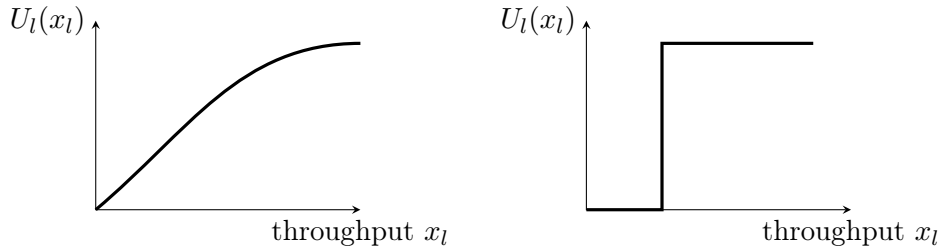

Figure 4.1 – Two examples of utility functions $U(\cdot)$. x_l is the throughput achieved on link l. The function on the left can for instance correspond to best effort traffic with diminishing returns, while the step function on the right implements a tight throughput requirement, e.g., for voice transmission.

Our goal is to optimize the performance of such a network. We use link utility, a function of received throughput, as a measure of performance. In particular, we consider the utility of link l as given by a function $U_l : \mathbb{R} \to \mathbb{R}$. We thus define $U_l(x_l)$ as the utility of link l when the throughput achieved on link l is x_l. The utility is an idealized representation of the *satisfaction* the link's user receives as a function of the actual (application-layer) throughput achieved. The function U_l might depend on the type of traffic and/or the load flowing through l. We illustrate the shape of some possible utility functions in Figure 4.1. In our testbed experiments of Section 4.5, we focus on a popular family of concave utility functions that depend on a parameter $\alpha \in \mathbb{R}$ and are defined as

$$U_l(x_l; \alpha) = (1 - \alpha)^{-1} x_l^{1-\alpha}. \tag{4.1}$$

These utility functions are known as the α-*fairness* functions, and they have been defined in [MW00]. When $\alpha = 0$, we have $U_l(x_l) = x_l$ and maximizing the sum of the utilities amounts to maximizing the total throughput of the network. When α increases, the concavity of the function increases, and maximizing the sum of utilities consists in finding configurations for which the links achieve increasingly similar performances (thereby increasing fairness). We assume that both the function $U_l(\cdot)$ and the traffic demand (or load) of l are known to the AP of l (the traffic load is easy to measure at the AP). Although we consider α-fairness utility functions in our evaluations, our algorithm can accommodate general utility functions; in particular they need not be concave or continuous.

Our objective is to maximize the global sum of utilities, which corresponds to solving the optimization problem

$$\mathsf{maximize} \quad \sum_l U_l(x_l) \quad \mathsf{over} \quad \mathcal{S}, \tag{4.2}$$

where the sum is taken over all the L links. The dependency on the spectral configurations is captured through the x_l variables. In particular, as explained in Chapter 3, the

throughput achieved on a link l is a complex function of the spectral configurations, wireless conditions, and traffic loads of its neighbors. In the next section, we present our algorithm for the distributed allocation of spectrum configurations.

4.3 Spectrum-Allocation Algorithm

In this section, we assume that each AP is able to communicate with its neighbors using an out-of-band channel, thus allowing for the exchange of messages regarding configurations and estimated utilities. We defer the detailed description of how to enable this neighbor discovery and communication in practice to Section 4.4, where we describe our protocol design and implementation.

4.3.1 Algorithm Description

As explained in Chapter 1, the optimization problem (4.2) is NP-complete (as it is more general than a graph-coloring problem). Therefore, we follow an approach similar to that in Chapter 2, and we devise an iterative approach that converges arbitrarily close to optimal configurations when the traffic loads and channel conditions are steady. Our distributed procedure is described in Algorithm 2, from the point of view of an AP A. Each AP keeps an exponential timer of mean wake-up time $1/\lambda$ that is independent of other APs. The algorithm is run when the timer fires. In this case, AP A first estimates its potential throughput (its achievable throughput when sending saturated traffic) for each configuration $c_A \in \mathcal{C}$, using some information about its neighbors. This information consists of the neighboring configurations $c_{\mathcal{N}_A}$, the physical rates currently used by the neighbors, their traffic loads, as well as the channel gains measured during the neighbor-discovery procedure. This information is exchanged on demand, only once per algorithm iteration, among neighboring APs (not shown in Algorithm 2). It is required by AP A in order to estimate its potential throughput at line 8 of the algorithm, using one of the model built in Chapter 3.

AP A then calculates $U^A(c_A)$, the sum of utilities of all links served by A, based on these estimated throughputs. It then sends a query request (line 11) to all its neighbors for their own estimated utilities for each configuration c_A (used by A). Let us now consider how this query is treated when AP A itself receives such a query, as shown starting in line 18. When AP A receives a query message from a neighbor B, it estimates the potential throughput for each of its links, for each possible configuration $c_B \in \mathcal{C}$ of B. Here again, this operation requires that A queries its own neighbors (some of which may be two hops away from B) about their current configurations, physical rates, traffic loads and measured channel gains (not shown in Algorithm 2). AP A then computes $U^A(c_B)$ as the sum of utilities of all links $l \in A$ for each $c_B \in \mathcal{C}$, and sends these values to B.

When AP A receives replies from its neighbors, these values are stored in a matrix $\boldsymbol{U}_A^{\text{nb}}$, where $\boldsymbol{U}_A^{\text{nb}}(B, c_A)$ denotes the estimated utility of neighbor B when A uses configuration c_A. AP A then uses these values to compute $U_A^{\text{nb}}(c_A)$, the sum of neighbors' utilities when A uses configuration c_A, and $\widetilde{U}^A(c_A)$, the total estimate of the neighborhood utility, including A. This value is then used in line 16 to draw a random configuration according to the Gibbs distribution. Such a random sampling procedure converges to states that are arbitrarily close to the global optimum of Problem (4.2), as explained in the next section.

Algorithm 2: Generalized spectrum allocation at AP A

1 **Initialization:**
2 Set the temperature $T > 0$
3 Start with a random generalized configuration c_A
4 Setup an exponential timer with mean wake-up time $1/\lambda$
5 **When the timer fires:**
6 **for** *each* $c_A \in \mathcal{C}$ **do**
7 **for** *each link* $l \in A$ **do**
8 estimate potential throughput $\hat{x}_l(c_A \cup c_{\mathcal{N}_A})$
9 compute $U^A(c_A) = \sum_{l \in A} U_l\left(\hat{x}_l(c_A \cup c_{\mathcal{N}_A})\right)$
10 **for** *each neighbor* $B \in \mathcal{N}_A$ **do**
11 send a `query_u` message to B
12 Upon reception of a `reply_u` message from neighbor B, fill matrix $\boldsymbol{U}_A^{\text{nb}}$, where $\boldsymbol{U}_A^{\text{nb}}(B, c_A)$ is the estimate utility of neighbor B when A uses configuration c_A
13 **for** *each* $c_A \in \mathcal{C}$ **do**
14 compute $U_A^{\text{nb}}(c_A) = \sum_{B \in \mathcal{N}_A} \boldsymbol{U}_A^{\text{nb}}(B, c_A)$
15 compute $\widetilde{U}^A(c_A) = U^A(c_A) + U_A^{\text{nb}}(c_A)$
16 draw a new configuration $c_A \in \mathcal{C}$ at random, with probability

$$\mathbb{P}(c_A) = \frac{\exp\left(\widetilde{U}^A(c_A)/T\right)}{\sum_{c'_A \in \mathcal{C}} \exp\left(\widetilde{U}^A(c'_A)/T\right)}$$

17 reschedule the timer
18 **Upon reception of a** `query_u` **message from a neighbor** B**:**
19 **for** *each* $c_B \in \mathcal{C}$ **do**
20 **for** *each link* $l \in A$ **do**
21 estimate potential throughput $\hat{x}_l(c_A \cup c_B \cup c_{\mathcal{N}_A \setminus B})$
22 compute $U^A(c_B) = \sum_{l \in A} U_l\left(\hat{x}_l(c_A \cup c_B \cup c_{\mathcal{N}_A \setminus B})\right)$
23 send a `reply_u` message that contains the set $\{U^A(c_B)\ \forall\ c_B\}$ to B

4.3.2 Configuration Sampling

We now show that our algorithm converges to utility-optimal allocations. The reasoning is very similar to that of Theorem 1 of Chapter 2 and it relies on the standard framework of Gibbs sampling. We therefore give only a brief outline of the convergence guarantees (rigorous convergence analyses can be found for similar settings, see for instance [BMS11]). Denote by $C_n \in \mathcal{S}$ the global state of the network at the n-th iteration of the algorithm. Each AP independently selects a configuration at the ticks of an exponential timer, based only on the current information that it receives from its neighbors. C_n is thus a Markov chain, whose transitions from one state to the next are specified by the Gibbs distribution in line 16 that depends only on the sum of utilities in a given neighborhood. The transition probabilities are such that C_n is reversible and thus converges in distribution, at geometric speed, to the stationary distribution with measure π given by

$$\pi(C) \propto \exp\left(\frac{\sum_l U_l(x_l)}{T}\right) \tag{4.3}$$

for a given global state $C \in \mathcal{S}$. This distribution has the same form as the limiting distribution of Theorem 1, and it is also parameterized by a temperature parameter T. Here too, for sufficiently low T, the distribution assigns arbitrarily large probabilities to the states that are the global optima of Problem (4.2). To observe that Equation (4.3) is indeed the stationary distribution of the chain, we can use a similar method as in [BMS11] and consider an augmented graph $G'(V, E')$, which is the original neighborhood graph $G(V, E)$ with an edge added between two nodes that are two hops away from each other. We can then apply the Gibbs-Markov equivalence [Bre01] on this augmented graph, where the sum of potential functions over the cliques of the graph corresponds to the sum of utilities in the AP neighborhoods (that is, $\widetilde{U}_A(c_A)$ for AP A). Each clique of the graph G' thus separately contributes to achieving convergence to states where the global sum of utilities is maximized.

4.3.3 Selfishness

In order to appeal to the Gibbs-Markov equivalence and to obtain convergence guarantees for arbitrary utility functions, we consider the sum of utilities in the AP's neighborhood for taking a spectrum allocation decision (in line 15). This implicitly embeds a notion of equity in how the neighbors' utility is taken into consideration, and it assumes that all the APs are willing to collaborate and follow the protocol.

In order to study situations where the APs might depart from the protocol, we now introduce a selfishness parameter, β, that controls this equity by letting each AP weigh its own utility and those of its neighbors. Note that this is orthogonal to the notion of fairness among links implied by the α-fairness utility functions. In a general manner, β should be seen as a weight given to the utilities of the various APs. We thus redefine

$\widetilde{U}^A(c_A)$ (computed in line 15) as a weighted sum:

$$\widetilde{U}^A(c_A) = 2\left(\beta U^A(c_A) + (1-\beta)U_A^{\mathrm{nb}}(c_A)\right), \tag{4.4}$$

where $0 \leq \beta \leq 1$. When $\beta = 0.5$, Equation (4.4) boils down to our utility-optimal mechanism (i.e., in this case $\widetilde{U}^A(c_A)$ is equal to the value computed in line 15 of Algorithm 2). When $\beta \neq 0.5$, the behavior differs from that defined in Algorithm 2. For $\beta \to 1$, the AP puts more weight on its own utility $U^A(c_A)$ and is thus increasingly selfish. Conversely, for $\beta \to 0$, the AP puts more weight on its neighbors' aggregated utility and is thus increasingly altruistic. With $\beta \neq 0.5$, we can no longer claim reversibility of the Markov chain C_n and the convergence to globally optimal states is not guaranteed. Nevertheless, considering different values for β enables us to investigate experimentally the effect of selfish APs on the achieved global throughput and on convergence. We evaluate our algorithm with different β values in Section 4.5.

4.4 Implementation

In this section we briefly describe the implementation of our algorithm, including the accompanying neighbor-discovery protocols required for the local collaboration.

4.4.1 Overall Description

We implemented the distributed algorithm, throughput prediction module, as well as the neighbor AP and client discovery procedures (described below) in about 3500 lines of C++ code, mainly in the form of new elements of the Click modular router [KMC+00] in userspace. This code runs at the APs. The clients are only modified to react to changes of spectral configurations selected at their APs; these changes are announced with dedicated beacons. To ensure the accuracy of time-dependent functions, all the control packets sent by the algorithm on the wireless medium (i.e., the neighbor-discovery frames, and the frames sent to clients for discovery and configuration changes) are prioritized over best-effort traffic and use a different MAC queue (similar to the technique employed by SAW in Section 2.5). For the throughput predictions (in lines 8 and 21 of Algorithm 2), we wrote the code to generate predictions that use a learned model based on SVR (as presented in Chapter 3).

4.4.2 Neighbor Discovery

Our spectrum-assignment algorithm is collaborative and thus requires information exchange between neighboring APs that do not necessarily operate on the same frequency band. Furthermore, at a given AP, the models for throughput prediction described in Section 3 need to know the channel gains between this AP and its neighboring APs

and their clients. We therefore design two accompanying protocols for discovering the neighboring APs and their clients, and for measuring the corresponding channel gains.

Neighboring Access Point Discovery

This protocol is a rendez-vous based procedure. The APs periodically meet on a predetermined frequency band and each AP broadcasts a dedicated packet containing its public (WAN) IP address. Each AP uses the default narrowest bandwidth (20MHz in our 802.11n implementation) and the maximum transmit power to ensure that a maximum of neighboring APs receive this transmission. The neighboring APs' IP addresses are then used to contact them over the wireline network to exchange further information (e.g., when announcing client discovery as described below, or when querying for utilities or configurations as described in overall in Algorithm 2). If the APs belong to a single administrative domain, they can use their local backbone network for collaboration. In contrast, if the APs belong to different administrative entities, they can use their Internet connection for collaboration.

In addition to the IP address, each receiving AP records the received signal strength (RSS) from its neighbors. The APs are assumed to have synchronized clocks by using a protocol such as the network time protocol (NTP), and the meeting time is determined using a modulo of the Unix time (in seconds) with the number of available rendez-vous channels. Similarly, the channel center-frequency for each meeting is determined by a modulo of the time with the channel index, which results in an iteration over available channels. This iteration is useful because it keeps the APs from getting stuck in a dedicated channel that may be poorly suited for this IP exchange because of factors such as heavy external traffic load.

Similarly to the micro-sensing operation described in Section 2.3.2 of Chapter 2, switching to the rendez-vous channel temporarily prevents the APs from sending or receiving traffic to/from their own clients. For this reason, on the one hand, the discovery periods need to be kept as short as possible, so as not to disrupt any time-sensitive traffic. On the other hand, this duration must be sufficiently long to let the APs to meet despite synchronization errors[1]. The time required to send the discovery frames is negligible (< 1 ms, even when using low physical rates and a narrow bandwidth), and so is the time required to switch back and forth to the discovery band using recent wireless chips[2]. However, in the Internet, it has been observed that most NTP-synchronized clocks are within 21 ms from reference time, and all are within 29 ms on average [MTH97].

[1] We send one discovery frame at the beginning of the period and one at the end, to ensure success in case of minimal overlap.

[2] For example, as mentioned in Chapter 2, Atheros reports switching times of 2 ms for its 802.11a/b/g/n AR9390 chipset. See: `http://www.qca.qualcomm.com/wp-content/uploads/2013/11/AR9390.pdf`. In particular, in this chapter, we ignore the fact that our own hardware has relatively long switching times, and rather focus on the ideal performance that can be obtained with recent wireless chips.

Therefore, the discovery duration should be governed by the synchronization accuracy. In our implementation, we use a discovery duration of 50 ms, which ensures a good discovery probability, even for the worst cases where two nodes both have a time offset of $20 - 30$ ms. Such a duration is also compatible with most delay-sensitive traffic. But, if such a duration happens to be detrimental to QoS, the APs are still free to skip the discovery operations scheduled while they are serving delay-sensitive traffic. This would come at the price of delaying the convergence process of the algorithm, but would not otherwise affect performance. Finally, although our reasoning follows a worst-case scenario where the APs are synchronized remotely over the Internet, let us mention that we could expect in practice that nearby neighboring APs have relatively low RTTs. The time synchronization can then be expected to be more accurate; the ideal scenario being when the APs use a local network for collaboration, in which case they can obtain a much tighter synchronization and further reduce the time spent out of their operating band.

Neighboring Client Discovery

We propose a protocol that enables the APs to measure the channel gains between themselves and the neighboring clients, without modifying the client devices. The protocol works as follows. Directly following an AP discovery procedure, each AP announces the time for its own next *client-discovery session* to all its neighboring APs. This client-discovery session consists of the AP sending an empty unicast frame to each of its clients. Contrary to rendez-vous based AP discovery, the AP uses its current operating spectral band (channel center-frequency and bandwidth) for this procedure so that it does not require client-side modification. This band information is therefore included with the time announcement. All neighboring APs that have tuned to this spectral band at the announced time can then observe the corresponding MAC-layer ACK packets sent by the clients and record the corresponding RSS values. Note that the neighboring APs do not need encryption information in order to measure these packets' RSSs with a virtual interface in monitoring mode. The frequency of the client-discovery sessions is a configurable parameter. In our implementation, we choose a random time between every second and every third AP discovery session. The moving averages of the RSS values recorded during these two discovery procedures are used by the AP and its neighbors to compute the various channel gains shown in Figure 3.5 of Chapter 3, which in turn are used as inputs of the throughput prediction module. In this context, filtering with moving averages mitigates the effect of noisy RSS measurements on throughput prediction.

4.5 Testbed Evaluation

In this section, we first describe our experimental settings and methodology, and we then analyze the results obtained on our testbed.

4.5.1 Experimental Settings and Methodology

We employ the testbed described in Appendix A with 802.11n. We use 20 and 40 MHz channel bandwidths (which are the legacy bandwidths with 802.11n), 2×2 MIMO, and 10 different transmit power values in the set $\{3\text{dBm}, 5\text{dBm}, \ldots, 21\text{dBm}\}$. We make our experiments in the 5.735-5.835 GHz band, which comprises a total of 100 MHz of spectrum (namely, five channels of 20 MHz). We randomly select between 8 and 10 AP-client pairs among the 22 nodes of our testbed. Each pair starts in a random configuration of center frequency, bandwidth and transmit power. We conduct our throughput measurements using UDP traffic generated by `iperf`, with packets of 1500 B. After 600 seconds, we start our algorithm at each AP for 3000 seconds. Unless otherwise stated, the results shown on the plots are the averages and standard deviations obtained over 10 such runs. For the algorithm execution we set temperature $T = 0.01$ and the average wake-up time $\lambda = 600$ seconds. We use the α-fairness utility functions defined by Equation (4.1) with $\alpha \in \{0, 1, 4\}$, thus yielding the following link utility functions:

- $\alpha = 0$. In this case, we have $U_l(x_l) = x_l$. When all links use this utility function, the optimization problem (4.2) boils down to the maximization of the sum of throughputs, irrespective of other considerations such as fairness.
- $\alpha = 1$. In this case, we have $U_l(x_l) = \log(x_l)$. Using this function is equivalent to maximizing proportional fairness. The goal of such an optimization objective is to provide a trade-off between efficiency and fairness by allocating more resources to links with larger potential throughput.
- $\alpha = 4$. In this case, we have $U_l(x_l) = -x_l^{-3}/3$. This function represents a compromise between proportional fairness and max-min fairness. Increasing α corresponds to increasing the weight put on fairness in the optimization objective. Therefore, this function is the fairest of the three functions that we consider.

4.5.2 Comparison with [KBC+07]

We benchmark our algorithm against the one proposed in [KBC+07]. This algorithm uses a Gibbs sampler to find configurations that minimize the overall interference. As its original version samples channel center-frequencies only, we "augment" it to sample bandwidths and transmit powers as follows. We modulate the power received by a node a from a node b by (i) the transmit power used by b and (ii) the overlap between a's receive spectrum mask and b's transmit spectrum mask (see [MSBA06]), assuming perfect band-pass filters. Overall, this benchmark is useful for comparing how our algorithm performs in comparison to an algorithm that considers only the unilateral objective of minimizing interference (and not maximizing capacity – see Section 1.1.1). We run the algorithm [KBC+07] (with our augmented metric) offline, using the whole testbed channel gains matrix in input, for 1000 iterations. The resulting allocations are denoted K+, and

are run for 1000 seconds in our testbed. This is repeated 10 times to obtain confidence intervals.

4.5.3 Results

We now present our experimental results. We first evaluate our algorithm in terms of convergence and performance. Next, we test the generality of the optimization criteria by studying the performance, in terms of throughput and fairness, using our different utility functions with the different values of α. Finally, we explore the interplay of the various parameters by examining the selected spectrum allocations for different utility functions, selfishness parameters, and traffic loads.

Convergence and Performance

Figure 4.2 shows the temporal evolution of the average total sum of throughputs, when all links use $\alpha = 0$, as well as the average sum for K+. We observe that K+'s single criterion that consists in reducing interference does not result in significant throughput gain over random configurations in this case. This is because the K+ algorithm favors narrow channel widths and lower transmit powers in the case of channel overlap, which might not be beneficial in general. In contrast, we observe encouraging results for our algorithm. After it is started, the system converges to total throughputs that are about 30% higher than those of K+ or when random configurations are used. In addition, convergence happens fast, on average in less than two iterations of the algorithm per AP[3].

It is difficult to compare these results against those obtained by SAW in Chapter 2. SAW was tested with 802.11g in the 2.4 GHz spectrum band that contains only 3 orthogonal 20 MHz bands. In contrast, the present results are obtained using 802.11n on a spectrum band that contains 5 orthogonal bands of 20 MHz[4]. Therefore, using random spectrum-allocations performs better on average in the present case (as random allocations are less likely cause interference when more spectrum is available).

Selfishness

We designed our algorithm assuming collaborative APs, but it is informative to investigate the potential effect that APs behaving selfishly or altruistically (i.e., using $\beta \neq 0.5$ in Equation 4.4) might have on the overall performance. To this end, we consider three values

[3]The first iteration happens on average at 1200 seconds, and the second iteration on average at 1800 seconds.

[4] Unfortunately, we were not able to implement the micro-sensing procedure described in Chapter 2 for 802.11n, and therefore we cannot directly compare the present results with those obtained with SAW. Furthermore, SAW does not allocate transmit powers and its optimization objective does not immediately extend to this additional parameter.

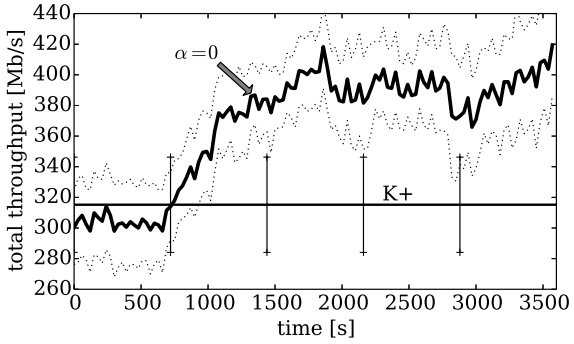

Figure 4.2 – Sum of all links throughputs as a function of the time. Our algorithm starts at 600 seconds and runs with $\alpha = 0$. The K+ line denotes the average total throughput obtained with the augmented algorithm of [KBC+07] described in Section 4.5.2.

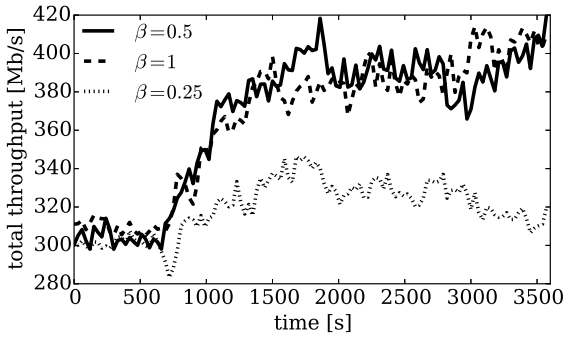

Figure 4.3 – Sum of throughputs for different values of the selfishness parameter β. In this setting, the sum of throughput is not affected when each AP is completely selfish ($\beta = 1$), whereas altruistic behavior ($\beta = 0.25$) has a negative effect onto global performance.

of the selfishness parameter β defined in Section 4.3.3. We use $\beta = 0.5$ (the default value for which our algorithm is shown to be utility-optimal), $\beta = 1$ (completely selfish APs) and $\beta = 0.25$ (where each AP is more altruistic and puts more weight on maximizing its neighbors' utilities than its own). Figure 4.3 shows the temporal evolution of the sum of throughputs for the different levels of selfishness. We observe that networks with altruistic APs achieve limited overall throughput improvements compared to when they adopt higher values of β. Altruistic APs appear indeed to under utilize the available spectrum, which penalizes the overall throughput. However, there is no noticeable difference beyond certain values of β, suggesting that selfishness does not decrease the overall throughput in this case. This result is similar to those observed for SAW in Section 2.4.6: Even if the APs behave selfishly, they still tend to employ configurations that do not significantly penalize their neighbors; because mitigating interference is often mutually beneficial.

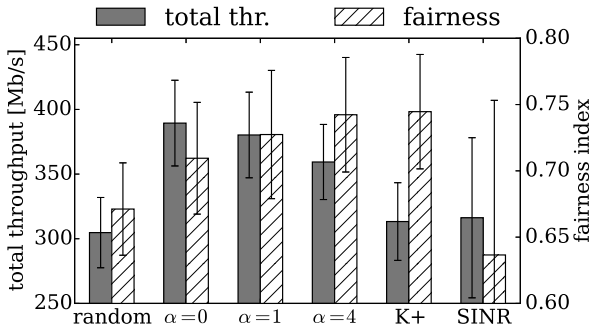

Figure 4.4 – Throughput and fairness for α-fairness, with $\alpha = 0$, $\alpha = 1$ and $\alpha = 4$. We also show the results for random configurations, the K+ algorithm, and our algorithm with $\alpha = 0$, but using an measurement-fitted SINR model instead of a learned model for estimating throughput.

Comparison of Utility Functions

We now compare the performance obtained with various utility functions. We conduct experiments where all the links use $\alpha = 0$, $\alpha = 1$ and $\alpha = 4$. We measure the resulting fairness using Jain's index, by computing

$$\frac{(\sum_l x_l)^2}{L \cdot \sum_l (x_l)^2},$$

where x_l is the measured throughput obtained by link l. Figure 4.4 shows the average throughput and fairness obtained with the various utility functions. In order to evaluate the system in steady state, the statistics are obtained over the last 1000 seconds of each test run. We observe that all utility functions perform well both in terms of fairness and throughput, in comparison to K+ or random configurations. Notably, our algorithm adapts well to the various utility functions: $\alpha = 0$ provides the greatest throughput, and increasing α improves fairness. In line with theoretical expectations, $\alpha = 4$ provides slightly better fairness and lower throughput than $\alpha = 1$. To the best of our knowledge, this is the first time that utility-driven optimization is employed for spectrum allocation in a real network.

In this resource-allocation context, it turns out that our throughput prediction technique, introduced in Chapter 3, is instrumental in achieving good performance. In Figure 4.4 we also plot the throughput and fairness for $\alpha = 0$, obtained when using a measurement-seeded SINR model (denoted "SINR"). Our algorithm with the learned throughput prediction module significantly outperforms the instance running with the SINR model, both in terms of throughput and fairness.

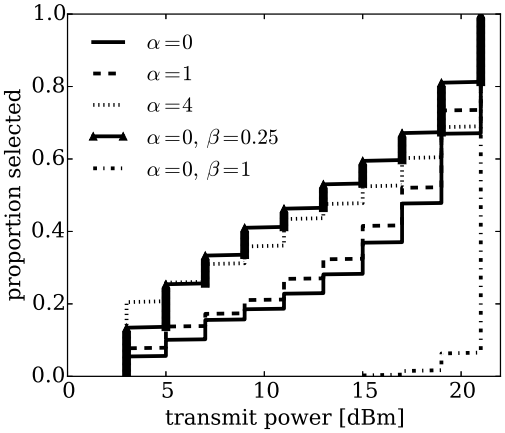

Figure 4.5 – CDF of transmit powers sampled with different utility functions, and different degrees of selfishness (when not indicated, β is set to $\beta = 0.5$). The APs use low transmit powers when the utility functions privilege fairness or when they behave altruistically.

Selected Configurations

We now examine the configurations that are selected by our algorithm under different scenarios. In Figure 4.5, we show the CDF of the transmit powers selected by our algorithm, for the different utility functions and two different values of the selfishness parameter β. As could be expected, altruistic APs (using $\beta = 0.25$) tend to use low transmit powers, whereas selfish APs (using $\beta = 1$) almost always use the maximum transmit power. More notably, it turns out that this trend is also present when the APs change their fairness objectives: fair utility functions tend to use lower transmit powers for a higher fraction of the time (see e.g., $\alpha = 4$ vs. $\alpha = 0$). Overall, these results indicate that the selected transmit powers are directly determined by the fairness objectives, even though our algorithm optimizes high-level functions of performance (which is itself an intricate function of the employed configurations).

For studying the effect of traffic load, we perform experiments where each link has a traffic load randomly chosen between 10 and 80 Mbps. In these experiments we take the traffic load into account by using the utility functions $U_l(x_l) = \min\{x_l/\text{load}_l, 1\}$, where load_l is the load of link l. In addition, we also consider situations where all links have a load of 100 Mbps, which corresponds to a backlogged situation. In this particular case, the employed utility function is equivalent to maximizing throughput (that is, using $\alpha = 0$). In Figure 4.6, we show the proportion of time that our algorithm chooses to use a 40 MHz channel bandwidth (on the left y-axis), and the average transmit power in dBm (on the right y-axis), as a function of traffic load at the AP that makes these choices. We observe that APs with heavier loads tend to use wider channel widths and larger transmit powers on average. This demonstrates a natural load balancing across

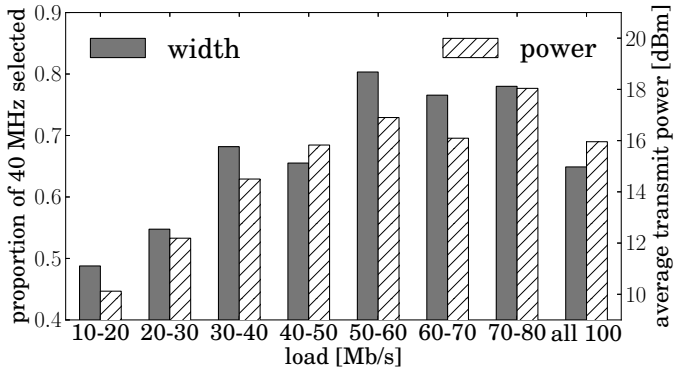

Figure 4.6 – Proportion of time that a bandwidth of 40 MHz is sampled and the average sampled transmit power, as a function of the AP traffic load.

these parameters: the APs with heavier traffic loads naturally consume more spectrum, both in frequency and spatial domains. This is indeed a desirable feature [MCW+08], and it confirms the ability of the throughput prediction modules built in Chapter 3 to suitably adapt to traffic load.

In contrast, when *all* APs generate 100 Mbps of load (last column, labeled "all 100" in Figure 4.6), the APs lower their resource consumption in spatial and frequency domains, compared to cases with heterogeneous loads. This is because, in these cases, heavily-loaded APs must share resources with other heavily-loaded APs. Indeed, instead of competing with each other, these APs collaborate to share the spectrum equitably. We thus deduce that the various utilities, fairness objectives and traffic loads directly impact the spectrum allocation patterns. In particular, both transmit power and channel bandwidths are used to load-balance the spectrum usage as a function of the optimization criteria.

4.6 Summary

In this chapter, we have presented an algorithm for the joint allocation of channel center-frequency, bandwidth and transmit power. Contrary to the algorithm of Chapter 2, this algorithm optimizes explicit formulations of the final performance. As a result, it accepts a wide range of optimization criteria (specified in terms of utility functions) that, in turn, directly determine the aggressiveness and efficiency of the configuration chosen by the nodes. In particular, the channel bandwidth and transmit power both determine the intensity of spectrum access, in frequency and spatial domains, respectively. These parameters are transparently adapted to provide natural load-balancing among access points subject to different traffic loads. As the effect of the various parameters and traffic loads on final performance is very intricate, the models that we built in Chapter 3 appear to be instrumental in achieving good performance and load-balancing properties.

In order to implement the direct performance-driven optimization proposed in this chapter, the APs need to exchange some control messages with their physical neighbors. In order to make it practical, we have proposed an accompanying AP- and client-discovery protocol, whereby neighboring APs can learn their mutual public IP addresses and use their backbone wired connection as a control channel. This approach is sufficiently general to be implemented by networks belonging to separate administrative entities that have access to an Internet connection. As such, it can represent a useful building block for other kinds of collaborative tasks requiring control traffic.

5 Scheduling in Time and Frequency Domains

5.1 Introduction

In the previous chapters, we have considered scenarios where the APs repeatedly re-evaluate the spectrum band on which they operate. This constitutes a first step in order to depart from the traditional "fully reserved" view of spectrum assignment, which is typically used in today's networks, and where a channel is chosen once and for all. However, due to practical constraints, the algorithms presented in the previous chapters still change their operating band only relatively infrequently – typically every few minutes –, compared to the much shorter timescales at which traffic dynamics might vary. In this chapter, we consider the spectrum-assignment problem in the context of *flexible channelization*, whereby the spectrum consumed by each station can be adapted on a per-frame basis. Recently, significant progress has been made in system design, showing that flexible channelization is feasible in practice [RSBC11, TFZ+10, YHC+10, CRB+12, CLL+12, YKQ13]. Compared to the algorithms presented in the previous chapters, flexible channelization further departs from the reserved view of spectrum management. In this chapter, we view the spectrum-assignment problem as a *scheduling* problem; the stations have to decide at what *time* and on which *frequency band* each frame should be sent.

Flexible channelization has the potential to provide several important advantages. First, adapting the spectrum consumed by each frame obviously removes the need for other spectrum-assignment algorithms acting at slower timescales. Second, adding frequency-domain decisions to the scheduling process can mitigate the severe time-domain overheads of 802.11, which are exacerbated by recent PHY layers. Third, as already mentioned, modulating the spectrum on a per-frame basis departs from the usual static channel assignment perspective and enables spectrum-allocation schemes to finely adapt to instantaneous traffic loads.

Despite important promises in terms of performance improvements, finding efficient schedules in time and frequency domains is difficult. Similarly to the algorithms presented

in the previous chapters, it requires that the stations reach some level of coordination in order to efficiently balance a minimization of interference with a maximization of spectrum usage (see Section 1.1.1). This is challenging, because for each frame the stations need to choose "time-spectrum blocks" that (i) do not overlap (to avoid interference) and (ii) consume as much of the available spectrum as possible (to maximize performance). For this reason, to the best of our knowledge, all schemes for flexible channelization that have been proposed so far rely on different forms of explicit signaling, synchronization, spectrum scanning or central control, in order to coordinate neighboring stations and efficiently organize transmissions (see e.g., [MCW+08, TFZ+10, FZD15] – so do more traditional spectrum assignment schemes operating at slower timescales, such as those proposed in the previous chapters).

In this chapter, we propose a new approach for scheduling packets in time and frequency, which is completely decentralized and requires no synchronization, explicit signaling, control traffic, nor spectrum scans. We propose TF-CSMA/CA (CSMA/CA in time and frequency domains), which is an extension of the time-domain CSMA/CA backoff mechanism of 802.11 to the frequency domain. In addition to the contention window and backoff counter used by 802.11 in the time domain, TF-CSMA/CA also adjusts dynamically the channel bandwidth and center frequency used for each frame, both of which determine the spectral-domain behavior. When a station is involved in a collision, it hops to another spectrum band and (with a certain probability) decreases both its time-domain aggressiveness and its (average) spectrum consumption. In contrast, when a station experiences a successful transmission, it sticks to its current spectrum band with a large probability, and it increases its (average) spectrum consumption with a small probability.

TF-CSMA/CA respects the design and engineering principles of 802.11: it is a purely random access mechanism that adapts its time-spectrum aggressiveness based only on transmission outcomes (collisions or successes) and carrier sensing. Although the proposed additional decision rules are relatively simple to describe, we will see that they produce non-trivial self-organizing behaviors, whereby stations avoid interference while efficiently using the available spectrum in both time and frequency domains.

Compared to time-domain random access, TF-CSMA/CA provides several important advantages. First, it drastically reduces the inefficiencies caused by the recent PHY layers of 802.11n and 802.11ac. As already mentioned in Chapter 1, these amendments can deliver up to multi-gigabit raw transmission rates at the physical layer, by using techniques such as MU-MIMO, aggressive modulations, and larger channel bandwidths (up to 40 MHz for 802.11n and up to 160 MHz for 802.11ac). Although these techniques drastically reduce the time required to transmit a frame, they also increase correspondingly the time-domain overheads due to backoff, acknowledgments, PHY-layer headers, and other MAC overheads (as we will explore in detail in the next section). To mitigate this, 802.11n and 802.11ac amendments have the ability to use frame-aggregation mechanisms

to increase transmission durations. The sizes of the aggregated frames can reach up to 65 kB for 802.11n and up to 4.5 MB for 802.11ac [Gas13]. Although heavy aggregation has the potential to increase efficiency, it does not help applications producing chatty traffic, or real-time traffic such as video, VoIP or gaming, which cannot afford to wait for large buffers to fill up. In contrast, TF-CSMA/CA drastically reduces these inefficiencies, by (i) reducing the channel width in case of interference (thus reducing the fraction of time consumed by overheads, as reducing the bandwidth increases the transmission duration while maintaining the same overheads) and (ii) being much more aggressive in the time domain (it is able to use minimum contention windows as low as $CW_{min} = 2$ while maintaining excellent fairness and small collision probabilities, compared to $CW_{min} = 16$ with current 802.11).

In addition to improving efficiency, TF-CSMA/CA also serves to solve the spectrum-assignment problem addressed in previous chapters at slower timescales. In this context, note that 802.11ac is capable of using different channel widths of 20, 40, 80 and 160 MHz and can decide to use channel bonding on a per-frame basis. However, this decision amounts only to deciding whether to employ or not the non-primary channel, and it offers only limited additional flexibility because the primary channel remains fixed. In fact, 802.11ac is known to require very careful spectrum planning in order to manage interference when large channel widths are employed [Gas13]. TF-CSMA/CA finds interference-free schedules and spectrum allocations directly at the MAC layer, as determined by instantaneous traffic loads. To the best of our knowledge, TF-CSMA/CA is the first mechanism to extend the backoff mechanism of 802.11 to the frequency domain.

We organize the remainder of this chapter as follows. In the next section, we give some background on the time-domain CSMA/CA mechanism used by 802.11, and explore some tradeoffs involved with packet scheduling. We present TF-CSMA/CA in Section 5.3. In Section 5.4, we analyze TF-CSMA/CA and show that it converges to interference-free spectrum allocations. Then, in Section 5.5, we use packet-level simulation to evaluate the performance of TF-CSMA/CA both in terms of throughput and short-term fairness, in a wide variety of settings. Finally, we present related work in Section 5.6 and give concluding remarks in Section 5.7.

5.2 Background and Motivation

5.2.1 The IEEE 802.11 Distributed Coordination Function

To arbitrate transmissions and avoid collisions, 802.11 specifies a distributed coordination function (DCF) based on CSMA/CA. When a station receives a new packet for transmission from the upper layer, it selects a *backoff counter* BC uniformly at random in $\{0, \ldots, CW - 1\}$, where CW denotes the contention window and is initially set to a minimum value CW_{min}. The backoff mechanism employs a discrete time scale; for each

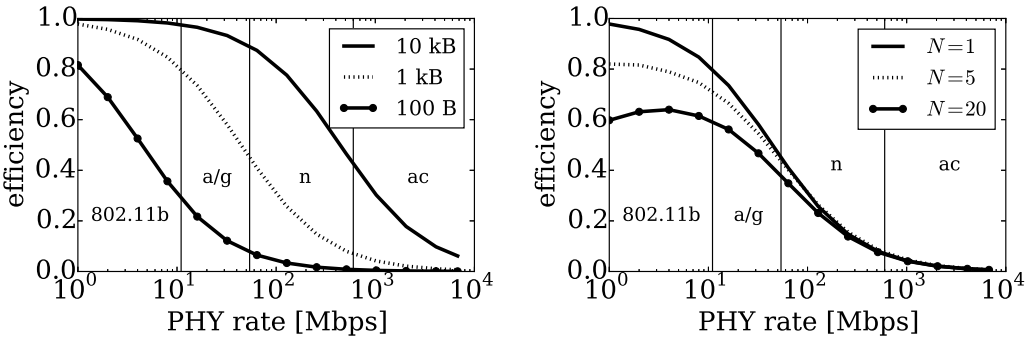

Figure 5.1 – Normalized throughput of the 802.11 DCF as a function of the PHY rate, for several frame sizes (left) and different numbers of stations N (right). The results on the left subplot are computed analytically for $N = 1$. The results on the right subplot are obtained by simulation with frames of 1 kB. We also show the rates employed by 802.11b/a/g/n/ac (delimited by vertical lines).

time slot during which the medium is sensed to be idle (i.e., below the carrier-sensing threshold), the station decreases its backoff counter BC by 1. When the medium is sensed busy, the station freezes its backoff counter until the medium is sensed idle again for a duration equal to DIFS (DCF Interframe Space). The station transmits when the backoff counter reaches 0. If the destination station successfully receives the frame, it waits for a duration equal to SIFS (Short Interframe Space) and replies with an ACK. If there is a collision (detected by a missing ACK), this is interpreted as contention and the transmitting station reduces its aggressiveness by doubling CW (up to a CW_{max} value). It then repeats the process.

The time-slot duration has to last long enough for reliable carrier sensing to be performed (i.e., measure the energy level), for the RF front-end to be switched from receiving to transmitting, and for possible propagation delays to be accounted for. It appears that these durations are mostly incompressible; the 802.11a/g/n/ac amendments have been using time-slot durations given by $t_{\text{slot}} = 9$ μs for more than a decade. Similarly, SIFS needs to account for the time required for the incoming frame to be processed and for the mode of the RF front-end to be switched, in order to transmit the ACK. 802.11a/n/ac use SIFS durations given by $t_{\text{SIFS}} = 16$ μs. These time constraints also propagate to DIFS, which is set to SIFS + 2 time slots and is equal to $t_{\text{DIFS}} = 34$ μs for 802.11a/n/ac. Finally, each frame starts with the transmission of a PHY preamble, which is required to detect and to decode frame transmissions, as well as to set the spectrum and modulation parameters. In total, 802.11ac uses PHY preambles lasting for durations of $t_{\text{PHY}} = 44$ μs [Gas13][1].

Let us define the *(normalized) throughput* (or *efficiency*) of a medium access control protocol as the product of (i) the fraction of time and (ii) the fraction of spectrum that

[1] We neglect the overhead of MAC headers, as those are transmitted at much higher rates.

are used for successful transmission of payload traffic. For example, a protocol that successfully transmits payload bits on 50% of the spectrum for 50% of the time has a normalized throughput of 25%. As 802.11 uses 100% of its channel, its efficiency is only determined by its time-domain operation. To analyze the efficiency of 802.11 as a function of the PHY rate, we can adopt a simple analytical model like the one proposed by Tan et al. [TFZ+10]. When there is only one transmitting station (and thus no collision), the average value of BC, which we denote by $\overline{BC}$, is given by $\overline{BC} = (CW_{min} - 1)/2$. We can thus easily compute the normalized throughput as

$$\text{efficiency}_{802.11} = \frac{t_{\text{data}}}{t_{\text{DIFS}} + \overline{BC}t_{\text{slot}} + t_{\text{PHY}} + t_{\text{data}} + t_{\text{SIFS}} + t_{\text{ACK}}},$$

where t_{data} denotes the time required to transmit the payload and t_{ACK} is the total time required to send the ACK. In Figure 5.1 (left), we show this throughput for different packet sizes as a function of the physical data rate[2]. Although faster transmission rates reduce the total time required for transmitting a frame, they exacerbate the time-domain overheads explained above: when sending 1 kB frames with a PHY rate of 600 Mbps (the maximum rate achievable with 802.11n, but well below the rates achievable with 802.11ac), the efficiency is below 10%. This is also true when the number N of contending stations is larger, as shown in Figure 5.1 (right) using simulation results (we give more details on our simulator in Section 5.5).

5.2.2 Improving Efficiency

We now present two techniques for improving efficiency, which are used by TF-CSMA/CA.

Reducing Backoff Durations

Current 802.11 amendments use $CW_{min} = 16$. One obvious solution for improving efficiency is to reduce the overhead due to the backoff process, by employing smaller contention windows (i.e., smaller CW_{min} values). Of course, there are good reasons for employing a reasonably large CW_{min}. If the stations transmit too aggressively, they can increase the collision probability (harming the overall efficiency) and even cause starvation. To illustrate this, consider the example depicted on Figure 5.2 with $N = 2$ stations using $CW_{min} = 2$. Such small contention windows cause a high number of collisions during the initial transmission attempts, and then they create a bi-stability effect, whereby one of the two stations monopolizes the channel for long durations.

This bi-stability effect induces poor short-term fairness (i.e., fairness evaluated on short time horizons), as some stations might starve for long durations before successfully sending a packet. In order to quantify this short-term (un)fairness and starvation effect,

[2]We use $t_{\text{ACK}} = 19\mu$s.

	Station u		Station v	
	BC	CW	BC	CW
	1	2	1	2
	0	2	0	2
collision	1	4	2	4
	0	4	1	4
station u transmits	1	2	1	4
	0	2	0	4
collision	2	4	3	8
	1	4	2	8
	0	4	1	8
station u transmits	1	2	1	8
	0	2	0	8
collision	3	4	13	16
	⋮	⋮	⋮	⋮

time

Figure 5.2 – Example of starvation with $CW_{min} = 2$. Repeated collisions lead to the rapid increase of the contention window of station v. There is a large number of collisions during the initial attempts, but station v becomes less and less likely to attempt a transmission. On average, it can thus take a long time until v experiences a successful transmission. When this happens, the situation might reverse and u might in turn have to wait a long time before experiencing a successful transmission.

we define the *inter-transmission time* ITX as the time duration between two successive successful transmissions of a given station. A scheduling algorithm is perfectly short-term fair (and prevents starvation) if ITX is constant and equal for all transmissions. We can therefore use the standard deviation of ITX (over all inter-transmissions of all stations) as a measure of unfairness, which we call σ_{ITX}. The larger σ_{ITX} is, the less the protocol is short-term fair, and the more likely it is for the stations to experience starvation[3]. In Figure 5.3, we show σ_{ITX}, as well as the normalized throughput, for several values of CW_{min} with 802.11 and $N = 5$ stations. With the default CW_{min}, 802.11 gets a throughput of about 6.7%. The throughput can be increased to more than 10% by decreasing CW_{min}. The cases with small CW_{min}, however, correspond to situations where a station monopolizes the medium for long durations (indicated by large σ_{ITX} values)[4]. In the extreme cases where $CW_{min} < 4$, some stations could not experience any successful transmission at all during the whole simulation time (which is set to one second in this case).

[3]Our measure of short-term unfairness follows closely the proposals made in previous studies on MAC layer short-term fairness; they also use inter-transmission times [BSDG+04]. Note that short-term fairness implies long-term fairness (which is measured in terms of the throughput received over long time intervals), whereas the converse is not true.

[4]We use $CW_{max} = 1024$ in these experiments. Reducing CW_{max} together with CW_{min} (i.e., by using a fixed number of backoff stages) increases collision rates and produces worse throughputs than the default configuration.

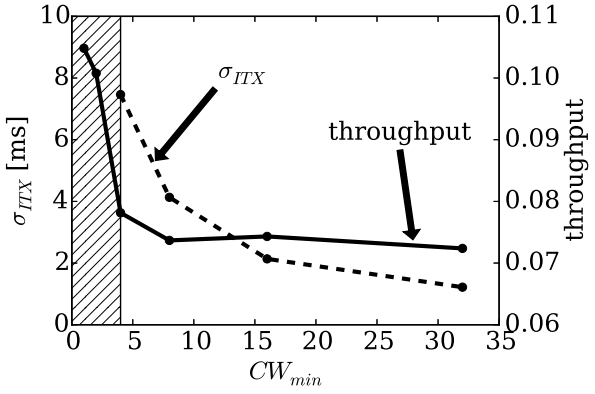

Figure 5.3 – Variation of the inter-transmission durations and throughput as functions of CW_{min}, for 802.11 with $N = 5$ stations sending frames of 1 kB, using a PHY rate of 600 Mbps. The hatched region ($CW_{min} < 4$) corresponds to complete starvation, where some stations could not experience a single successful transmission over the whole simulation time.

Using Narrow Channels for Multiple Stations

Even with dangerously small CW_{min} and backoff durations, 802.11 still obtains relatively low efficiencies (about 10% for the example shown in Figure 5.3). A solution to further improve efficiency is to reduce channel bandwidths; narrow channels require longer durations to send a given number of payload symbols and thus amortize the time-domain overheads. This idea was previously proposed by Chintalapudi et al. [CRB+12] and others. Note, however, that for a single station, simply dividing a wide-band channel into several narrow-band channels to send several longer frames effectively requires buffering more payload bits. In this respect, it is equivalent to performing aggregation on the original wide-band channel. However, when multiple stations compete for access, it is possible to increase efficiency by having each station transmit in parallel on different narrow bands (without requiring more payload to be buffered).

In the remainder of the chapter, we show that it is possible to implement the two above-mentioned solutions (reduction of backoff durations and narrow channels for multiple stations), by extending the contention-resolution process to the frequency domain. Backing off in the frequency domain enables TF-CSMA/CA to use very small CW_{min} values and reach efficiencies much higher than 802.11 (or any other time-domain scheduling mechanism), while maintaining excellent fairness and removing the starvation problem existing for 802.11 with small CW_{min} values. Overall, when $N = 1$, the efficiency gain comes only from a reduction in backoff duration. When $N > 1$, the gain comes from a combination of reduced backoff durations and reduced overheads over narrower bandwidths. Interestingly, we will see that when $N > 1$, the stations naturally converge to operating points where they use an average amount of spectrum proportional to $1/N$ – without knowing the number of stations N.

5.3 Scheduling in the Time and Frequency Domains

We now present TF-CSMA/CA. We start by introducing some necessary notations, and then present the algorithm itself.

5.3.1 Settings and Notations

We assume that the stations use a flexible baseband design such as the one proposed in [YKQ13], which lets the receivers detect the center frequency and bandwidth used by incoming transmissions (e.g., using PHY-layer preambles) and process frames accordingly. We focus on the case where the stations use contiguous chunks of spectrum (i.e., without fragmentation), which is simpler in terms of system design. Hence, with TF-CSMA/CA, in addition to its contention window CW and backoff counter BC, each station also maintains its current center frequency CF and bandwidth BW. These parameters are the spectrum parameters used at any point in time for packet transmissions and carrier sensing. To describe spectrum constraints, we denote by $\mathcal{CF}_{BW}$ the set of center frequencies that can be used with a given bandwidth BW (for example, in the 5.17-5.33 GHz band, we can have $\mathcal{CF}_{160\text{ MHz}} = \{5.25\text{ GHz}\}$ and $\mathcal{CF}_{80\text{ MHz}} = \{5.21\text{ GHz}, 5.29\text{ GHz}\}$, etc.). We write BW_{min} and BW_{max} for the minimum and maximum available bandwidths, respectively (e.g., in 802.11ac settings, we can have $BW_{min} = 20$ MHz and $BW_{max} = 160$ MHz). For simplicity of exposition, we assume throughout the paper that bandwidths are powers of 2, so that switching to the next larger (resp. next smaller) bandwidth is obtained by multiplying (resp. dividing) the current bandwidth by 2 (in a similar way as CW for 802.11). Finally, TF-CSMA/CA employs a value of CW_{min} that depends on the current bandwidth BW, and which we denote CW_{min}^{BW}.

5.3.2 Description of TF-CSMA/CA

TF-CSMA/CA is based on the following two observations:

- *Reaction to collisions in the frequency domain:* In the presence of contention, the stations should preferably separate their transmissions in the frequency domain. This is because orthogonal transmissions in the frequency domain enable simultaneous transmission of packets, and narrow bands reduce the time-domain overheads mentioned in Section 5.2. Therefore, upon experiencing a collision, a station should seek another spectrum band by changing its center frequency. In addition, frequent collisions should be interpreted as a signal that the station is using too much spectrum and should thus reduce its channel bandwidth to be able to find a free spectrum band.
- *Reaction to successes in the frequency domain:* Repeated successes indicate that a station operates alone in its spectrum band. The station should thus remain in this

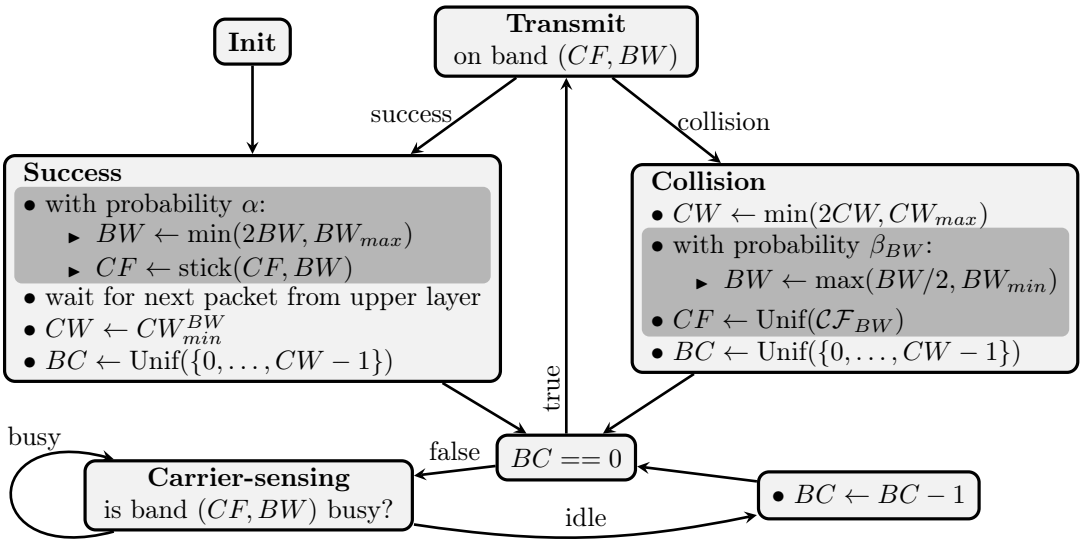

Figure 5.4 – Finite state machine of TF-CSMA/CA, as running in a station. In addition to the contention window CW and backof counter BC maintained by 802.11, TF-CSMA/CA also maintains the current center frequency (CF) and channel bandwidth (BW). The changes with respect to 802.11 are highlighted in dark gray. The function $\text{stick}(CF, BW)$ returns the center-frequency in $\mathcal{CF}_{BW}$ that is the closest to CF (breaking ties uniformly at random).

band or, with a small probability, try to increase its bandwidth in order to check if it is possible to use more spectrum.

We show the operation of TF-CSMA/CA at a single station as a finite-state machine in Figure 5.4. The stations start in any arbitrary combination of center frequency and bandwidth. The time-domain backoff mechanism is strictly equivalent to that of 802.11. Upon receiving a data packet from the upper layer, the station draws BC uniformly at random in $\{0, \ldots, CW_{min}^{BW} - 1\}$. It then performs carrier sensing on the current band that is specified by the tuple (CF, BW). For each slot during which the band is sensed idle, the station decreases BC by 1 (the time slots have the same duration as for 802.11). When BC reaches 0, the station attempts a transmission. If the destination station successfully receives the frame, it sends an ACK on the same band (CF, BW) after a SIFS duration (not shown on Figure 5.4). If the transmission collides (as detected by a missing ACK), the station doubles its contention window CW. If the transmission succeeds, the station sets CW to CW_{min}^{BW}.

The differences compared to 802.11 are shown in dark gray on Figure 5.4 and consist of the following additional actions. If a collision occurs, the station re-selects a new center frequency CF uniformly at random. In addition, it divides BW by 2 with a probability β_{BW}, which depends on the current bandwidth. In contrast, in the event of a successful transmission, the station doubles BW with a probability α. In this case, if BW changes

because of a successful transmission, the station also re-selects a new CF that is as close as possible to its current CF. This action is represented by the "stick" function in Figure 5.4: the function stick(CF, BW) simply returns the center frequency in $\mathcal{CF}_{BW}$ that is the closest to CF (breaking ties uniformly at random).

Note that the parameters BW and CF play roles in the frequency domain that are similar to CW and BC in the time domain. BW determines aggressiveness in the frequency domain, similarly to CW in the time domain. Likewise, CF and BC determine the localizations of the resource chunks consumed in the frequency and time domains, respectively.

5.3.3 Time-Domain Behavior and Configuration of CW_{min}

As we will see in Sections 5.4 and 5.5, the stations running TF-CSMA/CA converge to using non-overlapping spectrum bands that are well spread over the entire available spectrum. Although TF-CSMA/CA uses the same time-domain mechanism as 802.11, the fact that it can self-organize in the spectral domain makes it possible to configure the time-domain backoff mechanism in a more efficient way.

When the stations use large bandwidths, TF-CSMA/CA attempts to separate their transmissions in the frequency domain, by reducing their bandwidth and letting them transmit on orthogonal subbands. As a result, contention can be resolved entirely in the frequency domain and the stations operating with large bandwidths can be much more aggressive in the time domain (i.e., employ very short backoff durations) without risking to starve other stations. In contrast, when the stations already use narrow bandwidths (for example, if there are many stations using orthogonal bands with the minimum bandwidth BW_{min}), some stations may have to share some spectrum bands. Therefore, in this case, the stations should also use the time domain to separate their transmissions (i.e., employ reasonably long backoff durations – note, however, that the time spent in backoff represents a smaller overhead when using small bandwidths).

Overall, the importance of the time domain in the contention-resolution process should thus depend on the bandwidth. In particular, CW_{min}^{BW} should be a decreasing sequence of BW. In this paper, we propose to use CW_{min} values given by

$$CW_{min}^{BW} = \left\lceil \frac{16}{BW/BW_{min}} \right\rceil .$$

This sequence is such that $CW_{min}^{BW_{min}} = 16$, which corresponds to the current default CW_{min} employed by 802.11. In 802.11ac settings, the corresponding sequence is $CW_{min}^{20\text{ MHz}} = 16$, $CW_{min}^{40\text{ MHz}} = 8$, $CW_{min}^{80\text{ MHz}} = 4$ and $CW_{min}^{160\text{ MHz}} = 2$.

5.3.4 Mechanism for Adapting Contention Bandwidth

TF-CSMA/CA, as described, above uses the spectrum efficiently, but it can create problematic situations in terms of short-term fairness. When several stations transmit simultaneously on orthogonal narrow bands, it is possible that a given wider band, which contains some of these narrow bands, rarely becomes entirely free. Thus, if a station is contending on this wider band, it might have to freeze its backoff counter for long durations. To avoid this undesirable situation, TF-CSMA/CA uses the following additional mechanism (not shown in Figure 5.4), which incurs no performance penalty but improves short-term fairness.

Bandwidth Adaptation after Carrier Sensing: Each station halves its bandwidth BW with a small probability $\epsilon \ll 1$ after having sensed the medium busy due to a transmission by another station.

Although this mechanism is simple and requires no additional state, it ensures that each station waits on average no more than $1/\epsilon$ transmissions from other stations before reducing the bandwidth on which it contends[5]. It is useful when there are many stations, as it ensures that each station adapts the amount of spectrum on which it contends, without actually experiencing a collision (or waiting for one).

5.4 Analysis and Configuration

In this section, we first introduce a Markov chain model to study the spectral self-organization of TF-CSMA/CA. The main purpose of this analysis is to show that a simple frequency-domain scheduling scheme based on random access such as TF-CSMA/CA can exhibit self-organization. We conclude from the analysis that if the parameter α is small enough, the stations spend the vast majority of their time in states without interference. Then, in Section 5.4.2, we use the results of the analysis, as well as arguments related to the transient regime of TF-CSMA/CA, to configure the parameters of the algorithm, namely β_{BW}, α and ϵ. In particular, the arguments related to the transient regime consider the tradeoff between exploration (i.e., converging quickly and thus adapting to variable traffic) and exploitation (i.e., remaining in good states as much as possible to optimize performance).

[5]This way of adapting the state of the random-access mechanism as a function of the observed contention – without actually experiencing a collision – is similar to the mechanism introduced by the *deferral counter* in the IEEE 1901 MAC layer used for PLC communications [VBHT14]. The mechanism proposed here is simpler in the sense that it does not introduce any additional state.

5.4.1 Steady-State Model of Spectrum Consumption

Let $C := BW_{max}/BW_{min}$ be the number of smallest orthogonal subbands. For simplicity of exposition, we restrict our analysis to the case where $N = C$. For these values, there exists exactly one state without interference[6]. We first detail our Markov chain model and provide an example where $N = 2$ and $C = 2$ and then we extend our results to general N. We consider the case where the N stations belong to a single contention domain, and we assume that the channel quality is sufficiently high so that packet losses are due only to collisions. Without loss of generality, we set $BW_{min} = 1$ and $BW_{max} = C$. In addition, we make a modeling assumption similar to the decoupling assumption introduced by Bianchi in the time domain [Bia00], and we assume that every station attempts a transmission with a fixed probability p at any given time slot. Let $n_i, 1 \leq i \leq C$, denote the number of nodes using a band that overlaps with the i-th subband of width 1. We build a discrete-time Markov chain whose states represent all the possible patterns according to which the N stations can occupy the spectrum. Precisely, each state belongs to the set $\mathcal{S} := \{n_i : 1 \leq i \leq C,\ 0 \leq n_i \leq N\}$[7]. With TF-CSMA/CA, the stations change their spectral configuration after a transmission attempt with probability α (in case of success) or β_{BW} (in case of collision). Therefore, the transitions of the Markov chain from one state to the next occur upon a transmission attempt by any one of the stations (following the assumption of geometric backoff durations).

Example with Two Stations and Two Subbands

It is helpful to first consider the case with two stations and two subbands, as the states can be easily enumerated. In this case there are two bandwidths: one bandwidth corresponding to using all the band (i.e., $BW = BW_{max}$) and the other corresponding to half of the band (i.e., $BW = BW_{max}/2$). The Markov chain is represented in Figure 5.5. There are four possible states, denoted A, B, C and D: They correspond to the different combinations of spectrum occupation (the spectrum configurations of the two stations are represented by segments in Figure 5.5). As there are only two bandwidths, we only need one β_{BW}, because the stations can only decrease the bandwidth when $BW = BW_{max}$; hence, we define $\beta := \beta_{BW_{max}}$. The transition probabilities are easy to obtain from the reaction of TF-CSMA/CA to successes and collisions. For example, the transition probabilities from A to B and from A to D are $\frac{1}{2}p\beta^2$, because the other station (the one that does not trigger the state transition) has to transmit (which happens with probability p) and the two stations have to independently choose to reduce their bandwidth (with probability β^2).

[6] The case $N < C$ corresponds to an easier problem, in terms of finding interference-free assignments, and it can be treated similarly. Note that there does not exist a state without interference when $N > C$. Yet, we will see in Section 5.5 that TF-CSMA/CA performs extremely well for all N.

[7] Note that $\mathcal{S}$ describes the set of all possible states, also if stations could fragment their spectrum arbitrarily among the C subbands. When the stations do not fragment their spectrum (as is the case for TF-CSMA/CA), the possible spectral patterns belong to a subset of $\mathcal{S}$.

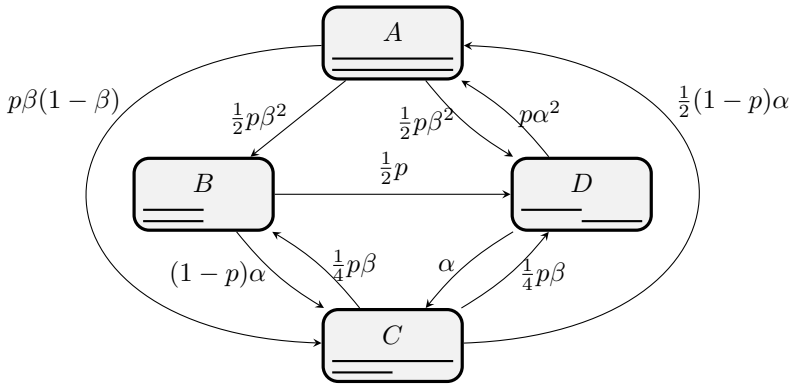

Figure 5.5 – Markov chain for the case of two stations and two bandwidths. The states are denoted A, B, C and D and the spectrum settings for the two stations within each state are shown by segments. We do not show self-transition probabilities as they would provide redundant information.

In the case under study, the most desirable state is D because there is no frequency-domain interference and the whole spectrum is used in this state. The following theorem shows that, if α is small enough, TF-CSMA/CA spends an arbitrarily large fraction of time in state D.

Theorem 2. *Let π_i be the stationary distribution of state $i \in \{A, B, C, D\}$. We have*

$$\pi_D \underset{\alpha \downarrow 0}{\longrightarrow} 1. \tag{5.1}$$

Proof. Using the balance equation for D, we get

$$\pi_D = \pi_D(1 - \alpha - p\alpha^2) + \pi_A \frac{1}{2} p\beta^2 + \pi_B \frac{1}{2} p + \pi_C \frac{1}{4} p\beta.$$

Let us define $\beta' := \min\{\frac{1}{4}p\beta, \frac{1}{2}p\beta^2\}$. We have

$$(\alpha + p\alpha^2)\pi_D \geq \sum_{i \in \{A,B,C\}} \pi_i \beta',$$

and thus

$$\frac{\pi_D}{\sum_{i \in \{A,B,C\}} \pi_i} \geq \frac{\beta'}{\alpha + p\alpha^2},$$

which in turn implies (5.1). □

Theorem 2 also holds if $\alpha = 0$, in which case D trivially becomes an absorbing state. However, in this case the chain is no longer ergodic and, for general configurations of N and C, it might remain "stuck" in absorbing states that avoid interference but under-utilize the spectrum. For this reason, TF-CSMA/CA employs a small but non-zero

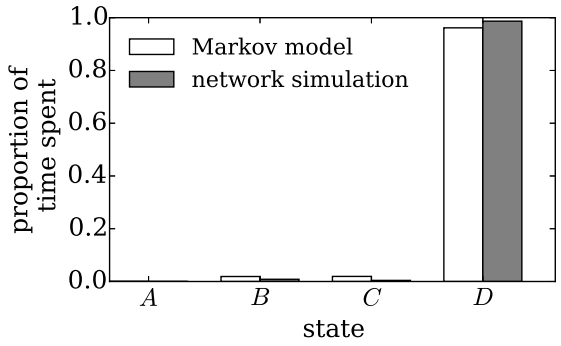

Figure 5.6 – Proportion of time spent in states A, B, C and D for the Markov chain of Figure 5.5 and a packet-level simulation of TF-CSMA/CA, with the settings $p = 0.05$, $\alpha = 10^{-3}$ and $\beta = 1$.

value of α (we elaborate further on this point in Section 5.4.2 as well as in the simulations of Section 5.5). In Figure 5.6, we show the fraction of time spent in the states A, B, C and D by the Markov chain of Figure 5.5, as well as by packet-level simulations of TF-CSMA/CA (see Section 5.5 for more details on our simulation settings). Although our proposed Markov model makes simplifying assumptions, it correctly captures the tendency of TF-CSMA/CA to spend the vast majority of the time in the best possible state in this scenario.

N Stations and N subbands

We now extend Theorem 2 to the general case of N subbands (with $C = N$).

Theorem 3. *Let $s^* \in \mathcal{S}$ be the (unique) interference-free state. We have*

$$\pi_{s^*} \underset{\alpha \downarrow 0}{\longrightarrow} 1. \tag{5.2}$$

Proof. Let us denote the bandwidth used by a station u in state $s \in \mathcal{S}$ by BW_u^s. We define

$$\mathcal{S}_1 := \left\{ s : \max_{u \in \{1,\dots,N\}} \{BW_u^s\} \leq 2 \right\} \setminus \{s^*\},$$

which is the set of states that are one transition away from s^*.

For any two states s and s', let $P_{s \to s'}$ denote the transition probability from s to s'. Now, when the network is in state s^* and a station transmits, there could be a random number, say k, of other stations transmitting at the same time, and k follows a binomial distribution of parameters $N - 1$ and p. Then, the network remains in state s^* if and only if none of the $k + 1$ transmitting stations decides to double its bandwidth. Therefore,

the probability of staying in state s^* is

$$\begin{aligned}
P_{s^*\to s^*} &= \sum_{k=0}^{N-1} \binom{N-1}{k} p^k (1-p)^{N-1-k} (1-\alpha)^{k+1} \\
&= (1-\alpha) \sum_{k=0}^{N-1} \binom{N-1}{k} (p(1-\alpha))^k (1-p)^{N-1-k} \\
&= (1-\alpha)(p(1-\alpha)+1-p)^{N-1} \\
&= (1-\alpha)(1-p\alpha)^{N-1} \\
&\geq (1-\alpha)^N \\
&\geq 1-N\alpha.
\end{aligned}$$

We can therefore use the balance equation for s^* to obtain

$$\pi_{s^*} \geq \pi_{s^*}(1-N\alpha) + \sum_{s\in\mathcal{S}_1} \pi_s P_{s\to s^*}.$$

Let $\beta_{min} := \min_{BW}\{\beta_{BW}\}$. It is easy to see that $P_{s\to s^*} \geq C^{-N} p^{N-1} (\beta_{min})^N$ for any state s in $\mathcal{S}_1$. We thus have

$$\pi_{s^*} \geq \pi_{s^*}(1-N\alpha) + \sum_{s\in\mathcal{S}_1} \pi_s C^{-N} p^{N-1} (\beta_{min})^N,$$

from which we obtain

$$\frac{\pi_{s^*}}{\sum_{s\in\mathcal{S}_1} \pi_s} \geq \frac{p^{N-1}(\beta_{min})^N}{C^N N\alpha}$$

and thus, for any state $s \in \mathcal{S}_1$,

$$\pi_s \leq A(N\alpha)\pi_{s^*}, \tag{5.3}$$

with $A = C^N/(p^{N-1}(\beta_{min})^N)$.

We now need to iterate this reasoning over the states that are not in $\mathcal{S}_1$ and need more than one transition to reach s^*. To this end, we extend the definition of $\mathcal{S}_1$ and define $\mathcal{S}_k := \left\{s : \max_{u\in\{1,\dots,N\}} \{BW_u^s\} = 2^k\right\}$, for $k \geq 2$. Now, for any $k \geq 2$ and any state $s_k \in \mathcal{S}_k$, let $\mathcal{N}_{s_k}$ denote the set of stations that use bandwidth 2^k in s_k. Note that $|\mathcal{N}_{s_k}| > 0$ by construction of $\mathcal{S}_k$, and so there exists a state $s_{k-1} \in \mathcal{S}_{k-1}$ that is obtained by halving the bandwidth of the stations in s_k that use bandwidth 2^k (and having them use any valid center frequency). It is again easy to see that $P_{s_k\to s_{k-1}} \geq C^{-N} p^{N-1} (\beta_{min})^N$ and so from the balance equation of s_{k-1}, we obtain

$$\pi_{s_{k-1}} \geq \pi_{s_k} C^{-N} p^{N-1} \beta_{min}^N.$$

We can now iterate this argument k times and combine it with inequality (5.3) in order

to obtain (noting that $k \leq \lceil \log_2(N) \rceil$)

$$\pi_s \leq A^{\lceil \log_2(N) \rceil}(N\alpha)\pi_{s^*}$$

for any possible state $s \in \mathcal{S}$, which concludes the proof. □

Theorem 3 shows that, by setting α sufficiently small, we can ensure that TF-CSMA/CA spends most of the time in the most desirable state. Based on this and other considerations, we next discuss the setting of α, as well as the other parameters of the algorithm.

5.4.2 Parameters Configuration

Let us now give some high-level comments on the setting of the parameters of TF-CSMA/CA, namely α, β_{BW} and ϵ.

Let us start with β_{BW}. A collision indicates that a station uses a band that overlaps with another station. In this case, the station should change its center-frequency and find a new (hopefully non-overlapping) band, and, if it is using a bandwidth that is too large to find a free spectrum band, it needs to reduce it. The average number of collisions needed to reduce BW is given by $1/\beta_{BW}$: this determines the time that a station has to find an interference-free configuration. Therefore, on the one hand, β_{BW} should be sufficiently small so that the stations are given enough time to find an interference-free configuration, if it exists, before reducing their bandwidths. On the other hand, it should not be smaller than needed, as otherwise the stations might lose time looking for an interference-free configuration that does not exist, thus harming the overall convergence of the algorithm.

Hence, in order to find an appropriate setting for β_{BW}, we need to compute the time needed to find an interference-free configuration for a given bandwidth, in situations where the stations should not reduce their bandwidth. This problem is similar to the one addressed in [BKSS13], which analyses the time it takes a balls-into-bins algorithm to find a configuration in which all bins have the same number of balls (in our particular case, one ball). In the algorithm of [BKSS13], each ball samples randomly each bin until it finds an empty one. This is similar to our algorithm when we have N stations that are using subbands of bandwidth equal to BW_{max}/N. In our case, when a station is in a non-empty subband, it detects this through a collision and randomly chooses another subband until it finds a free one. According to the analysis of [BKSS13], the time it takes to find such a configuration is $\mathcal{O}(N)$.

Based on the above reasoning, in our configuration of β_{BW}, a station sets this parameter equal to $c \cdot BW$, where c is some constant. Indeed, following the above rationale, when a station is using a small value of BW it is likely to contend with $\mathcal{O}(1/BW)$ stations and

thus the time needed to find an interference-free configuration will be given by $\mathcal{O}(1/BW)$; hence, in this case we set $\beta_{BW} = \mathcal{O}(BW)$. For the choice of c, we set it such that when a station is using BW_{max}, we have $\beta_{BW_{max}} = 1$ (which is clearly the best configuration for this scenario), which leads to $\beta_{BW} = BW/BW_{max}$.

As for the setting of α, based on the analysis of the previous subsection, we note that it should be set to a small value, so that the stations experiencing successful transmissions tend to remain on the same band. Whereas, setting α to a non-zero value enables the stations to reclaim possibly unused spectrum. Based on our evaluations of Section 5.5, we set $\alpha = 10^{-3}$, as we observe that it performs well in all settings.

Finally, we set ϵ based on the following reasoning. If we set $\epsilon = 1/x$, then a station has to wait on average up to x transmissions before halving the bandwidth on which it contends, which means that it might not be able to transmit during this time. Based on this, we set $\epsilon = 10^{-2}$, so that each station waits on average for no more than 100 transmissions before halving its bandwidth.

5.5 Performance Evaluation

5.5.1 Simulation Settings

We developed an event-driven packet-level simulator in Python. Our simulator is very similar to several other simulators that have been used in the past to model various MAC layers (see e.g., [DDT09]). We simulate TF-CSMA/CA with the same per-frame timing overheads described in Section 5.2 (time slot, SIFS, DIFS, PHY headers and ACK durations). We assume that each station achieves a physical rate proportional to its channel bandwidth, which corresponds to what is observed empirically [CMM+08]. Unless otherwise stated, we use 802.11ac settings and set $BW_{max} = 160$ MHz and $BW_{min} = 20$ MHz (and so the set of bandwidths available is $\{20, 40, 80, 160\}$ MHz). When simulating the 802.11 DCF (i.e., without our extensions to the frequency domain), we use the whole 160 MHz channel and the default configuration $CW_{min} = 16$ and $CW_{max} = 1024$ (i.e., 7 backoff stages). TF-CSMA/CA is also simulated with 7 backoff stages in the time domain. In line with 802.11ac, for each bandwidth BW, we use the set of center frequencies $\mathcal{CF}_{BW}$ such that all available bands of width BW do not overlap [Gas13]. The default parameters, which we use unless otherwise specified, are summarized in Table 5.1. We consider scenarios where all the stations interfere with each other (complete interference graph) and each station always has a packet to send (saturated traffic), except in Section 5.5.4, where we consider scenarios with non-complete interference graphs and non-saturated traffic. Finally, in order to isolate the effects of the random-access mechanism, we assume that there is no error due to channel noise (hence all packet losses happen due to collisions). Unless otherwise stated, we use a physical rate of 600 Mbps and 1 kB frames. Each configuration is evaluated using 10

BW	CW_{min}^{BW}	β_{BW}	PHY rate
160 MHz	2	1	600 Mbps
80 MHz	4	1/2	300 Mbps
40 MHz	8	1/4	150 Mbps
20 MHz	16	-	75 Mbps

Table 5.1 – Default parameters used for simulations.

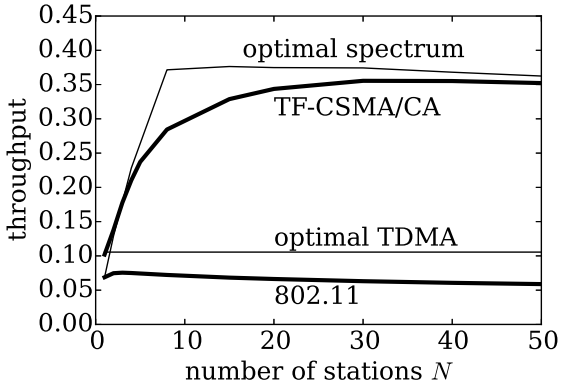

Figure 5.7 – Comparison of TF-CSMA/CA with 802.11, optimal TDMA and 802.11 operating with optimal spectrum assignment.

independent simulation runs lasting at least one second of simulated time (which is much larger than convergence times and typically corresponds to several thousands of transmission attempts).

5.5.2 Efficiency and Fairness

In Figure 5.7, we show the throughput obtained by TF-CSMA/CA as a function of the number of stations N, and we compare it against several other scheduling mechanisms: (i) "802.11 default" denotes 802.11 operating with its default configuration on the single wide band channel; (ii) "optimal TDMA" shows the performance obtained with a perfect TDMA scheme that uses N distinct time slots for a network with N stations (this corresponds, for instance, to the steady-state of the scheme proposed by Fang et al. [FMDL13]) and this is also an upper bound on the performance achievable by any enhancement of 802.11 that does not employ channelization (e.g., [Bia00, PBSA11]); (iii) "optimal spectrum" shows the performance obtained when all stations share the spectrum optimally (i.e., spreading their spectrum as evenly as possible). Obtaining this "optimal spectrum" configuration requires perfect information and is an upper bound of what can be achieved using centralized knowledge for the spectrum assignment.

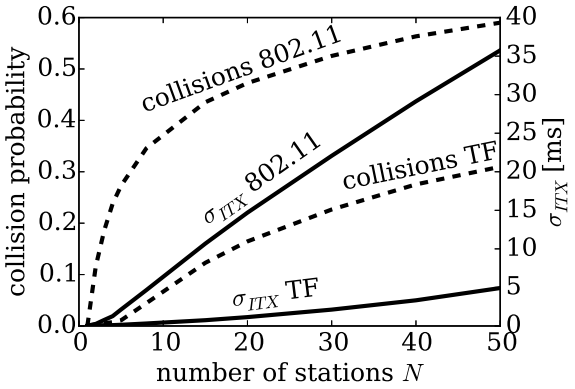

Figure 5.8 – Short-term unfairness σ_{ITX} and collision probability, for 802.11 and TF-CSMA/CA (denoted "TF").

Clearly, even perfect scheduling in the time domain using TDMA is less efficient than a mixture of time and frequency scheduling. Even though TF-CSMA/CA is completely decentralized, its backoff and frequency-repartition mechanism provides significant performance gains compared to time-domain scheduling, and achieves performance close to what can be obtained using a perfect centralized spectrum assignment. For $N = 1$, throughput is increased by roughly 1.5× due to a reduction in backoff durations – the performance in this case is similar to TDMA that sends packets back-to-back. For $N > 1$, splitting transmission onto smaller bandwidths provides important gains (up to more than 5× in this setting).

Importantly, these gains are not obtained at the price of short-term fairness. In Figure 5.8, we show σ_{ITX} and the collision probability obtained by TF-CSMA/CA and 802.11. TF-CSMA/CA achieves significantly better short-term fairness and smaller collision probabilities, which is a direct result of the parallelization of transmissions onto orthogonal subbands.

The gains provided in the frequency domain depend on the PHY rate and frame sizes. In Figure 5.9, we show the performance increase provided by TF-CSMA/CA for various PHY rates and frame sizes. As expected, the gains are the largest for high PHY rates and small packet sizes (i.e., for small overall transmit delays). Note that for the unlikely cases where large frames are transmitted at low PHY rates, TF-CSMA/CA can be slightly less efficient than 802.11. This is because, in these regimes, the efficiency of 802.11 is high, and the small inefficiency introduced by TF-CSMA/CA in the frequency domain is not compensated by significant gains in the time domain. However, such configurations are unlikely to happen in practice, as the oldest Wi-Fi standards with low PHY rates usually do not use frame sizes larger than 1500 B (they do not need to employ aggregation as their efficiency is already satisfactory).

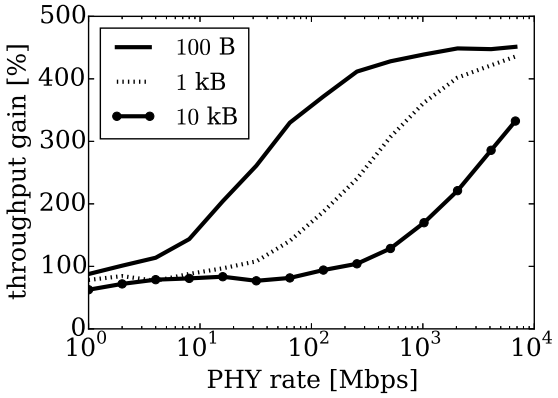

Figure 5.9 – Ratio between the efficiency of TF-CSMA/CA and that of 802.11 for $N = 5$. Larger gains occur for smaller transmission delays.

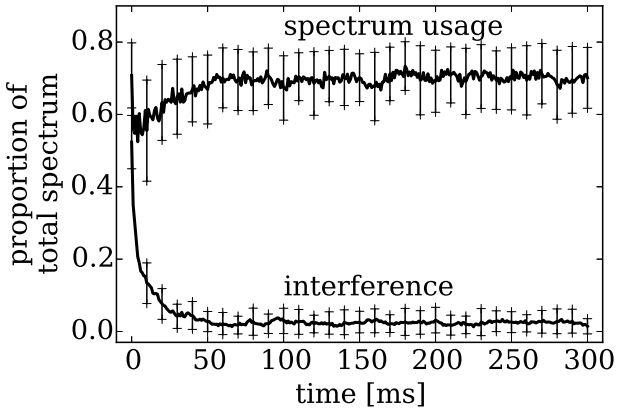

Figure 5.10 – Interference and spectrum usage over time for $N = 5$. TF-CSMA/CA balances well the two conflicting goals of minimizing interference while maximizing spectrum usage. Furthermore, convergence to steady state happens within 50 ms.

5.5.3 Interference and Self-Organization

TF-CSMA/CA trades off a very high time-domain inefficiency for some frequency-domain inefficiency. Furthermore, by adapting their spectrum bands, the stations pursue the two conflicting goals defined in Section 1.1.1: On the one hand, they aim to avoid using bands that are also used by other stations. On the other hand, they also try to use as much spectrum as possible in order to maximize their transmission rates.

To quantify how well TF-CSMA/CA reaches these goals, we define the *interference* as the fraction of the total spectrum that is used by *more than one* station at any given time. Similarly, we define the *spectrum usage* as the fraction of total spectrum used by *at least one* station. In Figure 5.10, we show the interference and spectrum usage over 300 ms of traces (averaged over 100 indepedent simulation runs using windows of 1 ms)

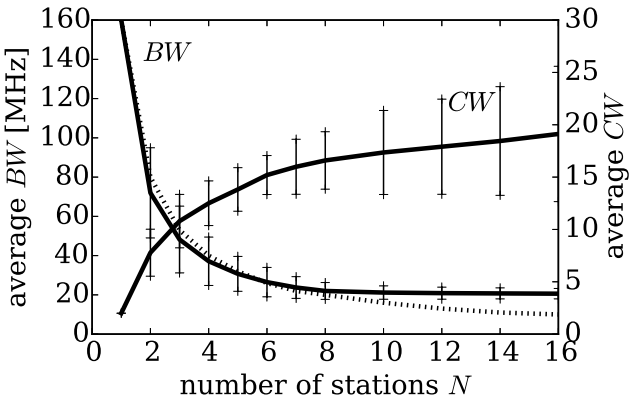

Figure 5.11 – Average values of BW and CW selected by TF-CSMA/CA. The optimal spectrum consumption is given by $160/N$ MHz and shown by the dotted line (partly indistinguishable from the BW curve).

for $N = 5$ stations. All stations start with the same center frequency and bandwidth BW_{max}. Although 5 stations are competing for access, TF-CSMA/CA converges to interference-free spectrum allocations. The nodes spend little transient time using the same spectrum and rapidly self-organize to use the spectrum efficiently; about 70% of the spectrum is used on average. Furthermore, because TF-CSMA/CA acts at the very fast time-scale of packets (re-)transmissions, convergence to steady-state is fast, within about 50 ms – even though the network started in a highly inefficient state in terms of spectrum assignment.

In order to illustrate how resource allocation is performed in time and frequency domains, we show in Figure 5.11 the average BW and CW parameters that are selected by the stations, as a function of N. Ideally, if all stations were to perfectly share the spectrum, each station should converge towards using a bandwidth BW given by BW_{max}/N (shown by the dotted line on Figure 5.11). It turns out that TF-CSMA/CA selects values for BW that are on average very close to optimal. This is remarkable, considering that the stations do not know N[8]. Note that for $N \geq 8$, TF-CSMA/CA uses mostly $BW = 20$ MHz, as this is the minimum available bandwidth.

5.5.4 Dynamic Traffic and Random Topologies

The probability α of doubling BW after a successful transmission responds to a tradeoff between exploration and exploitation. A small α ensures that the stations spend most of their time in states that minimize interference (see Section 5.4). However, a non-zero α is needed to regain available spectrum (for instance, in case the stations that were using

[8]Estimating N is not trivial; for instance, a number of works to find the optimal CW_{min} in 802.11 have designed algorithms to estimate N in some way [Bia00, PBSA11].

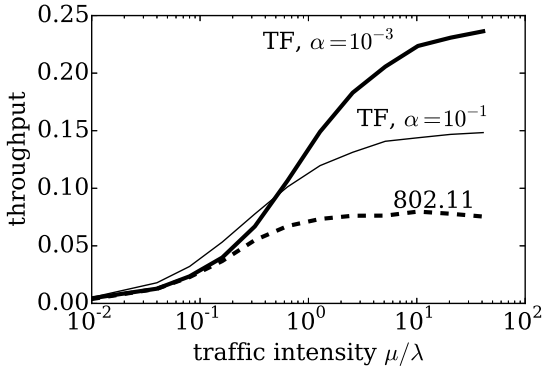

Figure 5.12 – Throughput as a function of average traffic load with $N = 5$. All stations have a fixed average "off" duration set to $1/\mu = 100$ ms and their average "on" duration $1/\lambda$ is varied between 1 ms (the stations are nearly silent) and 4 s (the stations are nearly backlogged).

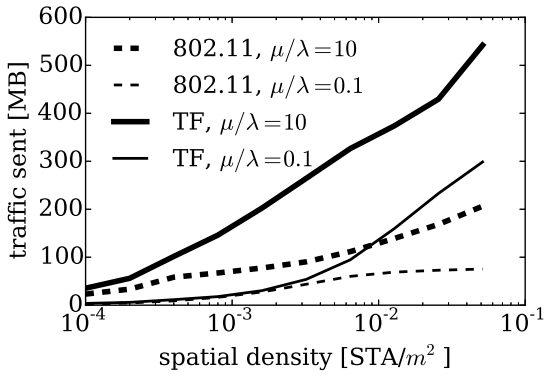

Figure 5.13 – Traffic sent over 5 seconds of simulated time, as a function of the average spatial density of the network topology.

that spectrum have left) and to ensure high utilization of the overall spectrum. We thus expect that a large α should favor situations with high traffic variability, whereas a small α should improve performance in steady state.

To quantify this effect, we introduce random traffic patterns as follows. The packets are generated by an exogenous on/off process at each station. The "on" durations are exponentially distributed with mean $1/\lambda$ and the "off" durations are exponentially distributed with mean $1/\mu$. In addition, the frame sizes are also exponentially distributed with mean 1 kB. In the following experiment, we set $1/\mu = 100$ ms, and we vary $1/\lambda$ between 1 ms and 4 s. In Figure 5.12, we show the throughput obtained in these settings, as a function of the resulting *average traffic intensity* μ/λ. As expected, a large α improves performance when μ/λ is small (bursty traffic). This is because the stations experience little contention and gain from re-using the spectrum more aggressively. However, even

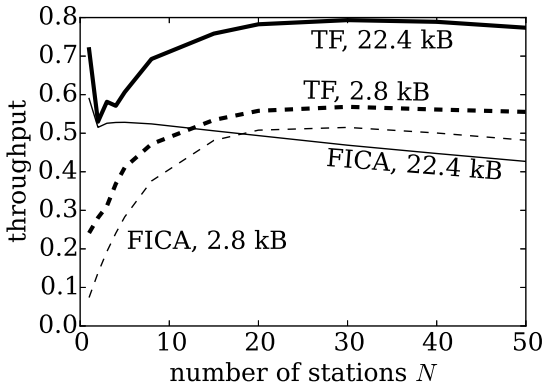

Figure 5.14 – Comparison of TF-CSMA/CA (denoted "TF") and FICA, for two different amounts of payload traffic available in the upper layer's buffer.

when using a relatively small $\alpha = 10^{-3}$ while the traffic is highly dynamic and bursty, TF-CSMA/CA performs at least as well as 802.11 (which uses the whole spectrum band).

This observation still holds when traffic intensity varies not only in time, but also in space. We make an experiment where the N stations are spread uniformly at random on a 100m×100m square and use an interference radius of $R = 30$m (that is, two stations separated by a distance less than R cannot transmit successfully at the same time on overlapping bands). In Figure 5.13, we measure the traffic sent when the number of stations varies between $N = 1$ and $N = 512$, for different average traffic intensities (using the default value $\alpha = 10^{-3}$). When the stations do not suffer from contention, TF-CSMA/CA and 802.11 offer comparable performances. The gain offered by TF-CSMA/CA increases with traffic intensities and spatial densities.

5.5.5 Comparison with FICA

We close this section by providing a comparison with the frequency-domain backoff scheme proposed by FICA [TFZ+10]. With FICA, the spectrum band is divided into several subchannels, and each station can use one or several subchannel(s) (not necessarily contiguous). FICA introduces a form of RTS/CTS signaling, and the transmissions occur in rounds; all transmitting stations have to simultaneously send an RTS signal and the receiving station elects winner(s) for each subband and announces them using a CTS signal. Note that, compared to FICA, TF-CSMA/CA is considerably simpler, as it does not require any signaling or synchronized transmissions.

The authors of FICA recommend splitting payload traffic into 1.6 kB frames to send over each subchannel for a case with a PHY rate of 580 Mbps and 14 subchannels [TFZ+10]. In our case we use only 8 subchannels, hence we scale this threshold correspondingly and configure FICA to send 2.8 kB frames on each subchannel. Note that this means that

FICA might need to access up to 22.4 kB (8×2.8 kB) of payload traffic in the upper layer's buffer, when a station decides to transmit on all subchannels simultaneously. We therefore consider two scenarios that correspond to two different saturation levels: (i) the upper layer's buffer always contains 22.4 kB of payload traffic (in which case, FICA can send up to 8 frames of 2.8 kB simultaneously and TF-CSMA/CA can send a unique frame of 22.4 kB); and (ii) the buffer always contains 2.8 kB of payload traffic (in which case FICA sends a 2.8 kB frame on a unique subchannel and TF-CSMA/CA also sends a 2.8 kB frame on its unique band). For FICA, we use 16 subcarriers in each subchannel for contention resolution and, in the 22.4 kB case, we use the proposed AIMD algorithm for choosing the number of subchannels used for transmission.

We show the results in Figure 5.14. For both saturation levels, TF-CSMA/CA outperforms FICA, even though TF-CSMA/CA does not rely on control traffic or synchronization primitive in order to organize transmissions. This is mainly due to the fact that FICA introduces extra per-frame overheads for the RTS/CTS signaling in order to explicitly organize transmissions. Such coordination is not needed by TF-CSMA/CA, because it provides self-organization in a purely random-access fashion. Note that, for large N, FICA performs better when only 2.8 kB is available in the transmit buffer. This is because, in these regimes, it is nearly always beneficial to use a single subchannel. Note also that, for large frames, TF-CSMA/CA performs significantly better for $N = 1$ compared to $N = 2$. This is because a unique station always uses the full spectrum band (which is efficient in this case), whereas a scenario with more stations can be less efficient due to the randomness of self-organization. In contrast, for larger N values, TF-CSMA/CA stations almost always contend using the smallest bandwidth, which is less challenging in terms of self-organization.

5.6 Related Work

Several recent works have shown the practical feasibility of flexible (or fine-grained) channelization [RSBC11, TFZ+10, CRB+12, YHC+10, CLL+12, YKQ13]. Among these, [RSBC11] and [YKQ13] propose schemes to schedule packets in time and on variable amounts of spectrum, but both algorithms rely on a central controller to take the scheduling decisions. [TFZ+10] proposes an extension of CSMA/CA to frequency domain, but it relies on explicit signaling (similar to RTS/CTS) sent on OFDM subcarriers and is sensitive to synchronization issues. [CRB+12] presents a novel radio design that enables the 802.11 DCF function to run independently on several narrow channels. However, the proposed mechanism to decide the subchannel(s) on which each link should contend needs to measure the residual airtime and number of contenders in all subchannels. Such an approach does not let stations to choose their spectrum on a per-packet basis and is closer to spectrum assignment schemes acting at slower timescales (such as those proposed in previous chapters). In addition, it increases efficiency without requiring buffering only if there are enough stations contending: When there are only one or a few stations, splitting

the wideband in several narrow bands to send several longer frames in parallel requires buffering more payload traffic. In contrast, in these regimes, TF-CSMA/CA increases efficiency by letting the stations be more aggressive in the time domain.

Some works consider the problem of scheduling packets in the context of flexible channelization. [MCW+08] considers the optimization problem of efficiently packing time-spectrum blocks, but the proposed algorithms require centralized control. [MJS+09] considers running independent 802.11 DCF on several subchannels (similar to [CRB+12]), but does not address the problem of deciding how much spectrum should be used by each station. Very recently, [FZD15] proposed an algorithm for scheduling packets in time and frequency domains, but here too the proposed mechanism requires additional control traffic and synchronization (as transmissions occur in synchronized rounds). [KSKL09] proposes a generalization of CSMA/CA to contend on several subchannels with variable intensities in the time domain. However, the stations always use a fixed channel for transmission and do not modulate their access intensity in the spectral domain. In addition, different techniques have been proposed to reduce the time-domain inefficiencies of Wi-Fi [SRCN11, MCRR11, FZZL12]. For example, [SRCN11] shows that it is possible to improve backoff overheads by resolving contention using signaling on OFDM subcarriers in the frequency domain. However, in these cases, the stations always use a fixed channel and, although contention *resolution* can be done in the frequency domain, the stations do not perform *backoff* in the frequency domain. Furthermore, we have seen in Section 5.5 that even completely removing the backoff overhead (using perfect TDMA) provides little gain compared to separating transmissions in the frequency domain.

In contrast to all the above works, TF-CSMA/CA is the first one to adapt both the time and frequency domain consumptions at the scheduling layer, using no explicit signaling but only collisions, successes and carrier sensing as implicit signals.

5.7 Summary

We have proposed TF-CSMA/CA, a scheduling algorithm that adjusts both the time and frequency access intensities in a random-access fashion. In contrast to existing schemes acting in time and frequency domains, TF-CSMA/CA is completely decentralized and reacts only to collisions, successes and carrier sensing. Overall, relying only on transmission outcomes provides a simple and effective way to assign spectrum to stations directly at the MAC layer, in a way that departs from the usual "reservation-based" view of spectrum usage, but that is rather determined by instantaneous traffic loads, just like CSMA/CA in the time domain. We have shown that (i) it provides self-organization in the spectral domain; (ii) although it is completely decentralized, it outperforms perfect time-domain scheduling, and (iii) it provides performance close to what is achievable when a centralized controller directly assigns spectrum to 802.11 nodes in a perfect (but monolithic) fashion.

With TF-CSMA/CA, the stations probabilistically reduce their spectrum usage in case of interference (with a relatively large probability β_{BW}), and they occasionally attempt to regain unused spectrum when there is no interference (with a small probability α). In this sense, TF-CSMA/CA takes a direct approach for balancing a minimization of interference with a maximization of spectrum usage. This is similar to SAW, proposed in Chapter 2, which minimizes the sum of a relatively large interference cost and a small but non-zero spectrum-usage cost. Compared to the algorithms proposed earlier, TF-CSMA/CA is further away from being implementable on off-the-shelf hardware. However, it provides important advantages, perhaps most notably because it acts at a much faster timescale (that of packet-transmission opportunities) and thus provides a greater flexibility.

6 Conclusions

In this thesis, we propose new techniques for finding efficient spectrum allocations in a flexible way. The success of Wi-Fi, together with the tendency of recent physical layers to rely on employing more spectrum, makes it increasingly important to find efficient ways of assigning variable-width channels to concurrent wireless links. In this context, a recurring challenge in finding efficient configurations is the need for balancing interference with capacity.

The algorithms proposed in this dissertation approach this challenge from different perspectives. In Chapter 2, we propose SAW, an algorithm that optimizes an explicit sum of two terms that capture interference and spectrum usage. This algorithm runs at the APs without requiring control traffic and it re-evaluates the spectrum configurations every few minutes. We have seen that it provides important performance gains, irrespective of network density, and even when some APs behave in a selfish manner. In Chapter 3, we observe that the performances obtained by interfering wireless nodes that are subject to different spectrum and traffic conditions exhibit intricate patterns that are due to complex interactions between the MAC and the PHY layers. We predict performances by treating this problem as a pure regression task, and we show that it is possible to learn implicit performance models from a set of real-world measurements. This technique outperforms other measurement-seeded models, and it generalizes well to unseen links and environments. In Chapter 4, we take advantage of these sophisticated performance models to formulate a utility-optimal algorithm for the joint allocation of channel center-frequency, bandwidth and transmit power. In this setting, the APs need to collaborate and exchange control messages in order to predict performances and to weigh the gains in utility provided by each configuration. Hence, we propose an accompanying neighbor-discovery protocol, whereby neighboring APs exchange their public IP address for subsequent collaboration. Using testbed experiments, we observe that it is possible to drive the network to operate in different regimes of efficiency, fairness and load balancing, as determined by the utility functions. Finally, in Chapter 5, we study the spectrum-assignment problem as a packet-scheduling problem. We propose TF-CSMA/CA, an extension of the backoff

mechanism of 802.11 to the frequency domain. With TF-CSMA/CA, the stations seek to avoid interference by reducing their spectrum usage and hopping to a new band when experiencing a collision. Conversely, they also seek to maximize spectrum usage by increasing their bandwidth when experiencing a successful transmission. We show that TF-CSMA/CA provides a form of self-organization. Furthermore, we observe that, even though it acts in a completely decentralized fashion without relying on any form of control traffic, TF-CSMA/CA provides excellent performances, comparable to 802.11 operating with optimal spectrum settings.

Although we have focused on the algorithmic aspects of the spectrum-assignment problem in this thesis, more steps need to be taken in order for such schemes to be become ubiquitous. In particular, standardization bodies should study the potential inclusion of frequency-domain behaviors in future standards, similarly to what was done to describe the backoff mechanism in the time domain.

As for future perspectives, we believe that the most promising directions for improving Wi-Fi performance in the long term will come from an efficient use of flexible-channelization systems. In particular, the tendency of current networks to rely on increasing amounts of statically-assigned spectrum has two undesirable effects. First, it is increasingly shifting the collision-avoidance process to the time domain, which puts more stress on the time-domain backoff procedure (as more and more networks are contending for access). Second, this time-domain operation is itself becoming increasingly inefficient, notably because of the use of wider bandwidths. This tendency contrasts with legacy networks operating on narrower spectrum bands, where transmissions are easier to separate in the frequency domain (at slower timescales), but for which the raw transmission speeds are lower. We therefore think that random-access algorithms performing backoff in both time and frequency domains, such as TF-CSMA/CA, represent promising solutions to achieve the best of both worlds in practice (i.e., maintaining a high efficiency while still accommodating large transmission speeds and small buffers). Hence, some interesting research directions include the characterization of the maximum capacity achievable by such random-access procedures, and how close they can get from being optimal (compared to centralized schemes operating with full information). In addition, a natural next step is to implement spectrum-assignment schemes of the flavor of TF-CSMA/CA on real hardware.

A Wireless Testbed

We built a wireless testbed in order to evaluate the algorithms proposed in Chapters 2 and 4, as well as the capacity estimation technique proposed in Chapter 3. Our testbed comprises 22 Wi-Fi nodes that are spread over the second floor of the EPFL BC building, as shown on Figure A.1.

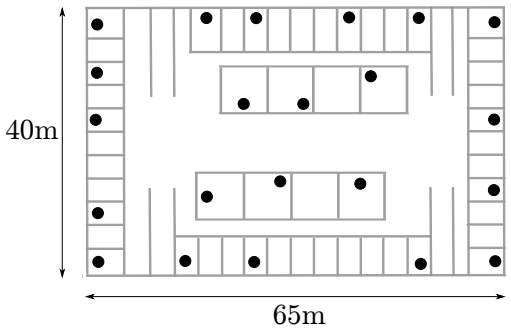

Figure A.1 – Testbed layout.

In this appendix, we provide some technical details about our testbed. Our main purpose is to provide helpful information for other groups who want to build similar testbeds.

A.1 Hardware

The nodes are PCEngines Alix 2D2 boards[1]. These boards have a 500 MHz AMD Geode CPU, 256 MB of RAM, two mini-PCI slots and two Ethernet interfaces. They very conveniently support PXE boot (see next section) and PoE (Power over Ethernet). In addition, they are low-power (about 3-4W when idle, and about 6W of peak consumption without mini-PCI devices) and relatively inexpensive (about 100 CHF for the board at

[1] `http://www.pcengines.ch/alix2d2.htm`

the time of writing). We use one Wistron DNMA92 mini-PCI radio card per board as to serve as the Wi-Fi interface. These cards use the Atheros AR9220 chipset, which provides 802.11a/b/g/n operation in both 2.4 GHz and 5 GHz bands. They have two antennas and support up to 2×2 MIMO with 802.11n, for PHY-layer data rates of up to 300 Mbps. Our boards are equipped with CompactFlash cards of 8 GB. As we will explain in the next section, although we use PXE for booting, the nodes also store a standalone Linux system on their flash cards. Finally, let us also mention that some of our boards are also equipped with an additional mini-PCI card that can be used for power-line communications (which we did not use in this thesis). These cards use the Intellon INT6300 chipset and support HomePlug AV operation with data rates up to 200 Mbps.

A.2 Software

A.2.1 System

The nodes run the OpenWrt Linux distribution[2]. This distribution targets embedded devices and it provides good support of the Alix boards and a large range of wireless devices. For the Wi-Fi cards, we use the open-source ath9k Atheros driver. All our nodes are connected to an Ethernet plug operated by the EPFL IT services, but all the plugs used by our testbed are configured to belong to an isolated VLAN. We use PXE for loading the operating system, so that when booting, the nodes receive the kernel directly through one of their Ethernet interface. In addition, the file systems of the nodes are accessed via NFS. Hence, we use two additional servers: an NFS server stores all the file systems as well as a Linux kernel used by all the nodes, and another server provides DHCP and PXE support. This way of storing the kernel and file systems is very convenient; it ensures that all nodes run the same software, and it makes it possible to re-compile a new version of the operating system (or the kernel) and very quickly distribute it to all nodes, without having to update their individual storage. In addition, it also serves to gather all the files created by all the nodes centrally on the NFS server. This is helpful for processing experiment results (which usually come in the form of log files stored by the nodes) in a centralized fashion. Finally, the nodes use the hardware watchdog of the Alix board in order to provide reliable operation in case of a critical failure, such as a kernel panic or a crash of the `ssh` daemon: the software regularly checks the state of the network and filesystem, and it sends a notification signal to the hardware watchdog. If the watchdog is not notified after a timeout, it triggers a reboot. If there is a failure when booting using PXE, the nodes load an alternative standalone Linux system that is stored on their flash card. This system is configured to reboot after a few minutes. This ensures that, even if there is a problem with one of the servers, the nodes keep re-trying to reboot using PXE until the problem is solved.

[2] `http://openwrt.org`

A.2.2 Experiments

We use Click [KMC+00] for most of our experiments. We find this framework very convenient, as it enables us to implement complex network stacks, while focusing only on relevant packet-processing tasks. Click can run either in userspace or as a kernel module. We run it in userspace, mainly due to technical difficulties[3] and for convenience (as userspace lets us use more libraries and floating-point expressions). In order to capture the PHY rate in the packets' header (as needed in Chapter 2 to compute airtimes and in Chapter 3), we had to modify the Click element in charge of parsing the `radiotap` headers (which are the headers used by interfaces in monitor mode).

We also wrote a set of Python scripts that enable us to easily describe and launch experiments from one of the two servers. These scripts contain a set of Python objects that describe nodes and links, and which offer procedures to change configurations, launch or interrupt Click, launch or interrupt traffic, etc. Overall, these scripts help us make sure that all our experiments are performed in consistent ways.

A.3 Changing Spectrum Configurations

To implement the algorithms of Chapters 2 and 4, we need the ability to dynamically change the spectral configurations employed by the nodes. For 802.11g (used in Chapter 2), we add a `debugfs` entry in the ath9k driver, in order to read/write the channel and channel bandwidths via dedicated `debugfs` files, without restarting the network. With this approach, we can configure our cards to use bandwidths of 5 and 10 MHz (in addition to the default 20 MHz). We used a signal analyzer to measure the spectral footprints of our hardware when using channel bandwidth of 5, 10 and 20 MHz, shown in Figure A.2.

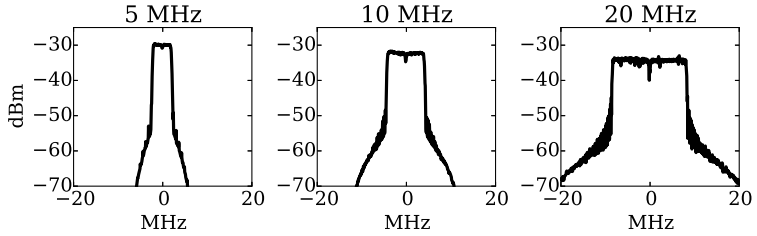

Figure A.2 – Spectrum footprint of the AR9220 cards using bandwidths of 5, 10 and 20 MHz.

802.11n readily supports bandwidths of 20 and 40 MHz. In this case, we use the `iw` tool for configuring the channel and bandwidth. However, by default `iw` refuses to change the configuration when the network is up. We thus modified the file `cfg.c` of `mac80211` in order to bypass the check and accept live re-configuration.

[3] Using Click as a kernel module together with ath9k caused repeated crashes in our case.

Bibliography

[AHK+03] KarenI Aardal, StanP Hoesel, ArieM Koster, Carlo Mannino, and Antonio Sassano. Models and solution techniques for frequency assignment problems. *4OR: A Quarterly Journal of Operations Research*, 1(4):261–317, 2003.

[AIKP08] Nabeel Ahmed, Usman Ismail, Srinivasan Keshav, and Konstantina Papagiannaki. Online estimation of rf interference. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 4:1–4:12, New York, NY, USA, 2008. ACM.

[AJSS05] Aditya Akella, Glenn Judd, Srinivasan Seshan, and Peter Steenkiste. Self-management in chaotic wireless deployments. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, MobiCom '05, pages 185–199, New York, NY, USA, 2005. ACM.

[BCM+09] Paramvir Bahl, Ranveer Chandra, Thomas Moscibroda, Rohan Murty, and Matt Welsh. White space networking with wi-fi like connectivity. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 27–38, New York, NY, USA, 2009. ACM.

[BEKF07] Ioannis Broustis, Jakob Eriksson, Srikanth V. Krishnamurthy, and Michalis Faloutsos. Implications of power control in wireless networks: A quantitative study. In *Proceedings of the 8th International Conference on Passive and Active Network Measurement*, PAM'07, pages 83–93, Berlin, Heidelberg, 2007. Springer-Verlag.

[Bia00] G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *Selected Areas in Communications, IEEE Journal on*, 18(3):535–547, March 2000.

[Bis06] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[BKSS13] Petra Berenbrink, Kamyar Khodamoradi, Thomas Sauerwald, and Alexandre Stauffer. Balls-into-bins with nearly optimal load distribution. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '13, pages 326–335, New York, NY, USA, 2013. ACM.

[BMS11] S. Borst, M. Markakis, and I. Saniee. Distributed power allocation and user assignment in ofdma cellular networks. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 1055–1063, Sept 2011.

[BPK+10] I. Broustis, K. Papagiannaki, S.V. Krishnamurthy, Michalis Faloutsos, and V.P. Mhatre. Measurement-driven guidelines for 802.11 wlan design. *Networking, IEEE/ACM Transactions on*, 18(3):722–735, June 2010.

[Bré79] Daniel Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, April 1979.

[Bre01] Pierre Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag New York Inc., corr. edition, February 2001.

[BSDG+04] G. Berger-Sabbatel, A. Duda, O. Gaudoin, M. Heusse, and F. Rousseau. Fairness and its impact on delay in 802.11 networks. In *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, volume 5, pages 2967–2973 Vol.5, Nov 2004.

[CHSO07] C. Clancy, J. Hecker, E. Stuntebeck, and T. O'Shea. Applications of machine learning to cognitive radio networks. *Wireless Communications, IEEE*, 14(4):47–52, August 2007.

[CLL+12] Eugene Chai, Jeongkeun Lee, Sung-Ju Lee, Raul Etkin, and Kang G. Shin. Building efficient spectrum-agile devices for dummies. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 149–160, New York, NY, USA, 2012. ACM.

[CMM+08] Ranveer Chandra, Ratul Mahajan, Thomas Moscibroda, Ramya Raghavendra, and Paramvir Bahl. A case for adapting channel width in wireless networks. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pages 135–146, New York, NY, USA, 2008. ACM.

[CRB+12] Krishna Chintalapudi, Bozidar Radunovic, Vlad Balan, Michael Buettener, Srinivas Yerramalli, Vishnu Navda, and Ramachandran Ramjee. Wifi-nc: Wifi over narrow channels. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 4–4, Berkeley, CA, USA, 2012. USENIX Association.

[DBL13] K.R. Duffy, C. Bordenave, and D.J. Leith. Decentralized constraint satisfaction. *Networking, IEEE/ACM Transactions on*, 21(4):1298–1308, Aug 2013.

[DDT07] Mathilde Durvy, Olivier Dousse, and Patrick Thiran. Modeling the 802.11 protocol under different capture and sensing capabilities. In *IEEE INFOCOM*, pages 2356–2360. IEEE, 2007.

[DDT09] Mathilde Durvy, Olivier Dousse, and Patrick Thiran. Self-organization properties of csma/ca systems and their consequences on fairness. *Information Theory, IEEE Transactions on*, 55(3):931–943, 2009.

[DGVB+11] Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. The impact of channel bonding on 802.11n network management. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 11:1–11:12, New York, NY, USA, 2011. ACM.

[DH09] R.C. Daniels and R.W. Heath. An online learning framework for link adaptation in wireless networks. In *Information Theory and Applications Workshop, 2009*, pages 138–140, Feb 2009.

[EPT07] R. Etkin, A. Parekh, and D. Tse. Spectrum sharing for unlicensed bands. *Selected Areas in Communications, IEEE Journal on*, 25(3):517–528, April 2007.

[FMDL13] Minyu Fang, David Malone, Ken Duffy, and Douglas Leith. Decentralised learning macs for collision-free access in wlans. *Wireless Networks*, 19(1):83–98, 2013.

[FZD15] Seyed K. Fayaz, Fatima Zarinni, and Samir Das. Ez-channel: A distributed {MAC} protocol for efficient channelization in wireless networks. *Ad Hoc Networks*, 31(0):34 – 44, 2015.

[FZZL12] Xiaojun Feng, Jin Zhang, Qian Zhang, and Bo Li. Use your frequency wisely: Explore frequency domain for channel contention and ack. In *INFOCOM, 2012 Proceedings IEEE*, pages 549–557, March 2012.

[Gas13] Matthew Gast. *802.11Ac: A Survival Guide*. O'Reilly Media, Inc., 1st edition, 2013.

[HBH06] Jianwei Huang, Randall A. Berry, and Michael L. Honig. Auction-based spectrum sharing. *Mob. Netw. Appl.*, 11(3):405–418, June 2006.

[HHSW10] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pages 159–170, New York, NY, USA, 2010. ACM.

[HP12] Nidhi Hegde and Alexandre Proutiere. Optimal decentralized spectrum sharing: A simulation approach. In *Proc. CISS*, March 2012.

[HRBSD03] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 836–843 vol.2, March 2003.

[HTF08] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2nd edition, 2008.

[IEE12] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, March 2012.

[KBC+07] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot. Measurement-based self organization of interfering 802.11 wireless access networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1451–1459, May 2007.

[KGD07] Anand Kashyap, Samrat Ganguly, and Samir R. Das. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, MobiCom '07, pages 242–253, New York, NY, USA, 2007. ACM.

[KMC+00] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, August 2000.

[KSKL09] Hojoong Kwon, Hanbyul Seo, Seonwook Kim, and Byeong Gi Lee. Generalized csma/ca for ofdma systems: protocol design, throughput analysis, and implementation issues. *Wireless Communications, IEEE Transactions on*, 8(8):4176–4187, August 2009.

[LCBM12] D. J. Leith, P. Clifford, V. Badarla, and D. Malone. WLAN channel selection without communication. *Computer Networks*, January 2012.

[LQZ+08] Yi Li, Lili Qiu, Yin Zhang, Ratul Mahajan, and Eric Rozner. Predictable performance optimization for wireless networks. In *ACM SIGCOMM*, SIGCOMM '08, pages 413–426, New York, NY, USA, 2008. ACM.

[MBB+06] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh. A client-driven approach for channel management in wireless lans. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.

[MCRR11] Eugenio Magistretti, Krishna Kant Chintalapudi, Bozidar Radunovic, and Ramachandran Ramjee. Wifi-nano: Reclaiming wifi efficiency through 800 ns slots. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 37–48, New York, NY, USA, 2011. ACM.

[MCW+08] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Yuan Yuan. Load-aware spectrum distribution in wireless lans. In *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, pages 137–146, Oct 2008.

[MDL07] David Malone, Ken Duffy, and Doug Leith. Modeling the 802.11 distributed coordination function in nonsaturated heterogeneous conditions. *IEEE/ACM Trans. Netw.*, 15(1):159–172, February 2007.

[MJS+09] Ritesh Maheshwari, CAO Jing, Anand Prabhu Subramanian, et al. Adaptive channelization for high data rate wireless networks. *Stony Brook University, New York, USA*, 2009.

[MPB07] V.P. Mhatre, K. Papagiannaki, and F. Baccelli. Interference mitigation through power control in high density 802.11 wlans. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 535–543, May 2007.

[MRWZ06] Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Analyzing the mac-level behavior of wireless networks in the wild. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 75–86, New York, NY, USA, 2006. ACM.

[MSA+06] Arunesh Mishra, Vivek Shrivastava, Dheeraj Agrawal, Suman Banerjee, and Samrat Ganguly. Distributed channel management in uncoordinated wireless environments. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, pages 170–181, New York, NY, USA, 2006. ACM.

[MSBA06] Arunesh Mishra, Vivek Shrivastava, Suman Banerjee, and William Arbaugh. Partially overlapped channels not considered harmful. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '06/Performance '06, pages 63–74, New York, NY, USA, 2006. ACM.

[MTH97] David L Mills, Ajit Thyagarjan, and Brian C Huffman. Internet timekeeping around the globe. Technical report, DTIC Document, 1997.

[MW00] Jeonghoon Mo and J. Walrand. Fair end-to-end window-based congestion control. *Networking, IEEE/ACM Transactions on*, 8(5):556–567, Oct 2000.

[NIK12] M. Nekovee, T. Irnich, and J. Karlsson. Worldwide trends in regulation of secondary access to white spaces using cognitive radio. *Wireless Communications, IEEE*, 19(4):32–40, August 2012.

[PBSA11] P. Patras, A. Banchs, P. Serrano, and A. Azcorra. A control-theoretic approach to distributed optimal configuration of 802.11 wlans. *Mobile Computing, IEEE Transactions on*, 10(6):897–910, June 2011.

[PVGa11] F. Pedregosa, G. Varoquaux, A. Gramfort, and al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[QZC10] Li Ping Qian, Y.J.A. Zhang, and Mung Chiang. Globally optimal distributed power control for nonconcave utility maximization. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6, Dec 2010.

[QZW+07] Lili Qiu, Yin Zhang, Feng Wang, Mi Kyung Han, and Ratul Mahajan. A general model of wireless interference. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, MobiCom '07, pages 171–182, New York, NY, USA, 2007. ACM.

[RMAQ07] E. Rozner, Y. Mehta, A. Akella, and Lili Qiu. Traffic-aware channel assignment in enterprise wireless lans. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pages 133–143, Oct 2007.

[RMR+06] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Measurement-based models of delivery and interference in static wireless networks. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 51–62, New York, NY, USA, 2006. ACM.

[RSBC11] Shravan Rayanchu, Vivek Shrivastava, Suman Banerjee, and Ranveer Chandra. Fluid: Improving throughputs in enterprise wireless lans through flexible channelization. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 1–12, New York, NY, USA, 2011. ACM.

[SdDF+11] Srikanth Sundaresan, Walter de Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Broadband internet performance: A view from the gateway. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 134–145, New York, NY, USA, 2011. ACM.

[Sha48] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal, The*, 27(3):379–423, July 1948.

[SRCN11] Souvik Sen, Romit Roy Choudhury, and Srihari Nelakuditi. No time to countdown: Migrating backoff to the frequency domain. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 241–252, New York, NY, USA, 2011. ACM.

[SRYB08] Vivek Shrivastava, Shravan Rayanchu, Jongwoon Yoonj, and Suman Banerjee. 802.11n under the microscope. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, IMC '08, pages 105–110, New York, NY, USA, 2008. ACM.

[SS04] AlexJ. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[TFZ+10] Kun Tan, Ji Fang, Yuanyang Zhang, Shouyuan Chen, Lixin Shi, Jiansong Zhang, and Yongguang Zhang. Fine-grained channel access in wireless lan. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pages 147–158, New York, NY, USA, 2010. ACM.

[VBHT14] Christina Vlachou, Albert Banchs, Julien Herzen, and Patrick Thiran. Analyzing and Boosting the Performance of Power-Line Communication Networks. In *ACM CoNEXT*, 2014.

[VFP09] Eduard Garcia Villegas, Rafael Vidal Ferré, and Josep Paradells. Frequency assignments in ieee 802.11 wlans with efficient spectrum sharing. *Wirel. Commun. Mob. Comput.*, 9(8):1125–1140, August 2009.

[wi-14] Wi-Fi Alliance 2014 Annual Report. `http://www.nxtbook.com/splash/WiFi_Alliance/wifi_annualreport2014.php`, 2014. [Online; accessed 22-May-2015].

[wia] `http://techcrunch.com/2012/04/05/study-61-of-u-s-households-now-have-wifi/`. [Online; accessed 22-May-2015].

[WL11] Beibei Wang and K.J.R. Liu. Advances in cognitive radio networks: A survey. *Selected Topics in Signal Processing, IEEE Journal of*, 5(1):5–23, Feb 2011.

[YBC+07] Yuan Yuan, Paramvir Bahl, Ranveer Chandra, Thomas Moscibroda, and Yunnan Wu. Allocating dynamic time-spectrum blocks in cognitive radio networks. In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '07, pages 130–139, New York, NY, USA, 2007. ACM.

[YHC+10] Lei Yang, Wei Hou, Lili Cao, Ben Y. Zhao, and Haitao Zheng. Supporting demanding wireless applications with frequency-agile radios. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 5–5, Berkeley, CA, USA, 2010. USENIX Association.

[YKQ13] Sangki Yun, Daehyeok Kim, and Lili Qiu. Fine-grained spectrum adaptation in wifi networks. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking*, MobiCom '13, pages 327–338, New York, NY, USA, 2013. ACM.

Julien Herzen

Av. de la Harpe 21
1007 Lausanne
Switzerland

j@hrzn.ch
20 Sep. 1984
Swiss citizenship

Education

Ph.D. (2015) in Communication Systems
Flexible Spectrum Assignment for Local Wireless Networks
École Polytechnique Fédérale de Lausanne (EPFL)

BSc. (2007) and MSc. (2009) in Communication Systems
With Specialization *Networking And Mobility*
École Polytechnique Fédérale de Lausanne (EPFL)

Professional Experience

PhD student, September 2010 - September 2015
EPFL, LCA lab, under the supervision of Prof. Patrick Thiran
• Design, implementation and analysis of various network algorithms
• Application and design of data analytics and machine learning techniques to various problems

Research Intern, September - December 2012
Technicolor Research Labs, Paris
• Work on adaptive resource allocation algorithms
Keywords: Machine learning, Testbed experiments, Simulation, Analysis

Research Intern, July - September 2011
Deutsche Telekom, T-Labs, Berlin
• Work on spectrum-assignment algorithms
Keywords: WLANs, Home networks, Self-organization, Testbed experiments, Simulation, Analysis

Research Intern, February - August 2009
DoCoMo USA Labs, Palo Alto CA
• Master thesis on scalable routing algorithms using virtual coordinates
Keywords: Random graphs, Graph embedding, Scalability, Simulation

Intern, summer 2007
EPFL, Distributed Information Systems Laboratory
• Design and deployment of Protopeer, a toolkit for rapid prototyping of peer-to-peer systems
Keywords: Distributed Systems, DHTs, Network Programming, PlanetLab

Selected Publications (complete list available: http://www.hrzn.ch)

- J. Herzen, A. Banchs, V. Shneer, P. Thiran. **CSMA/CA in Time and Frequency Domains**. to appear in *IEEE ICNP*, 2015
- J. Herzen, H. Lundgren, N. Hegde. **Learning Wi-Fi Performance**. in *IEEE SECON*, 2015
- V. Etter, J. Herzen (co-first author), M. Grossglauser, P. Thiran. **Mining Democracy**. in *ACM COSN*, 2014
  - ⋆ **Best paper award** – Featured in phys.org, epfl.ch, Scientific Computing, ACM Technews
- C. Vlachou, A. Banchs, J. Herzen, P. Thiran. **On the MAC for Power-Line Communications: Modeling Assumptions and Performance Tradeoffs**. in *IEEE ICNP*, 2014
  - ⋆ **Best paper runner-up award**
- C. Vlachou, A. Banchs, J. Herzen, P. Thiran. **Analyzing and Boosting the Performance of Power-Line Communication Networks**. in *ACM CoNEXT*, 2014

- J. Herzen, R. Merz, P. Thiran. **Distributed Spectrum Assignment for Home WLANs**. In *IEEE Infocom*, 2013
  - ⋆ Featured in Gizmodo, Engadget, `phys.org`, `epfl.ch`
- J. Herzen, C. Westphal, P. Thiran. **Scalable Routing Easy as PIE: a Practical Isometric Embedding Protocol**. In *IEEE ICNP*, 2011
- A. Aziz, J. Herzen, R. Merz, S. Shneer, P. Thiran. **Enhance & Explore: an Adaptive Algorithm to Maximize the Utility of Wireless Networks**. In *ACM MobiCom*, 2011
- J. Herzen, A. Aziz, R. Merz, S. Shneer, P. Thiran. **A Measurement-Based Algorithm to Maximize the Utility of Wireless Networks**. In *ACM S3*, 2011

Patents

- J. Herzen, R. Merz and P. Thiran. Method to optimize the communication parameters between an access point and at least one client device. US Patent 20140307571 (pending).
- H. Lundgren, J. Herzen and N. Hegde. Spectrum allocation in a wireless network. US and European (13305939.4) (pending).

Professional Services

- Reviewer for IEEE Transactions on Mobile Computing and Elsevier Computer Networks
- External reviewer for ACM Sigcomm, CoNEXT, Sigmetrics and IEEE Infocom, Secon
- TPC member of the ACM S3 2012 workshop

Teaching Assistant

TCP/IP Networking (MSc), Dynamical Systems Theory (MSc), C/C++ Programming (BSc), Mathematical Analysis (BSc)

Supervised Semester Projects and Master Theses

- Sébastien Epiney. "Indoor Localization using IEEE 802.11 WLANs", Master thesis
- Alexandre Becholey. "Improving Delay in Wireless Opportunistic Routing", Master thesis
- Zhu Jiahang. "Implementing and Evaluating the Gain of Opportunistic Routing on a Real Wireless Testbed", Master semester project
- Grégory Moix. "Implementation and Evaluation of a Geometric Routing Algorithm", Master semester project
- Gorica Tapandjieva. "Crawling and Mining Social Websites", Master semester project
- Pierre Pfister. "Design and Implementation of a Control Interface for a Wireless Testbed", Master semester project
- Hannah Muckenhirn. "Analysis of Political Data", Master semester project
- Jalloul Ben Soussia. "Evaluation of Wireless Networks Interference and Possible Counter-Measures", Bachelor semester project
- Bernard Gütermann. "Implementation of a Multipath Congestion Control Mechanism", Master semester project
- Victor Kristof. "Scraping and Mining Political Data from `www.admin.ch`", Master semester project

Data Science

Machine learning (supervised & unsupervised), Markov random fields, Bayesian models, Neural Networks, Support Vector Machines, Clustering, ...

Computer Skills

Python, Java, C/C++ , Linux, network programming, numerical computing (`numpy`, `scipy`, `sklearn`, `matplotlib`, Matlab), `git`, LaTeX, ...

Languages

French	mother tongue
English	strong knowledge
German	basic knowledge