PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Tesis de Maestría
en Informática

Self management of high density

wireless networks

Guillermo Apollonia

2015

Guillermo Apollonia
Self management of high density wireless networks
ISSN 0797-6410
Tesis de Maestría en Informática
Reporte Técnico RT 15-06
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República.
Montevideo, Uruguay, 2015

PEDECIBA INFORMÁTICA

Facultad de Ingeniería - Universidad de la República

Tesis presentada para la obtención del título de Magister en Informática

Self Management of High Density Wireless Networks

Author:
Guillermo Apollonia

Director de Tesis y Director Académico:
Dr. Javier Baliosian, Universidad de la República

Tribunal
Dr. Joan Serrat, Universidad Politécnica de Catalunia, España
Dr. Pablo Ezzatti, Universidad de la República
Dr. Eduardo Grampín, Universidad de la República

Montevideo, June 7, 2015

Abstract

IEEE 802.11 wireless networks are very popular in today's world. This popularity has been stimulated due to the use of mobile computing devices such as laptops, tablets, and Wi-Fi enabled phones. We can get 802.11 connectivity in schools, squares, parks and other public places. All of these places can have a high concentration of users. Moreover, there are other nonpublic places like lecture halls, hotel ballrooms, and convention centers that are common examples of spaces with high concentration of users in a high-density wireless communications environment.

Dense deployments of wireless networks suffer from increased interference and, as a result, bad user experience. The interference caused by the co-channel and adjacent channel interference driven by co-located devices is one of the main issues to address to improve network performance. The limited number of non-overlapping channels may lead to severe interference scenarios if no appropriated spectrum planning is employed.

In this work, we present an in-depth review of research work for the channel allocation strategies. Then, we formalize the channel allocation as a minimization problem of the interference level and we propose three different manners to optimize channel assignment between participating Access Points with the aim to improve network performance. The algorithms that we propose can be classified as local and uncoordinated, coordinated and distributed, and centralized. The local and uncoordinated solution behaves well in our testbed but present oscillatory issues that we tackle with a feedback control technique. Finally, this work presents an evaluation of the strategies, on a testbed and on a simulation environment. In the testbed we demonstrate the practical deployability of the solutions and lead to the conclusion that the local and uncoordinated implementation is worthy to be considered as a good strategy for the channels allocation problem where Access Points works in isolated manner. In the simulation, we test the scalability of both, the coordinated and centralized solution, and we show that they can be deployed in networks with more than thirty Access Points and as a results, we conclude that the centralized implementation is the best strategy to perform optimization decisions for channel allocation in connected networks .

Resumen

Las redes inalámbricas IEEE 802.11 son muy populares en el mundo actual. Esta popularidad ha sido estimulada debido al uso de dispositivos móviles tales como laptops, tablets y teléfonos Wi-Fi compatibles. Se puede tener conectividad 802.11 en escuelas, plazas, parques y otros lugares públicos. Todos estos lugares pueden tener una gran concentración de usuarios. Ms aún, hay otros lugares no públicos como las bibliotecas, centros de convenciones, salas de conferencias en hoteles, los cuales también son ejemplo de espacios comunes con una gran concentración de usuarios en entornos de comunicación inalámbrica de alta densidad.

Instalaciones de redes inalámbricas densas experimentan una interferencia creciente, y como resultado, una mala experiencia de usuario. Las interferencias co-canal y de canal adyacente producidas por dispositivos próximos entre sí, son uno de los principales problemas a abordar para mejorar la performance de la red. El número limitado de canales que no se superponen pueden conducir a escenarios de severa interferencia si no se emplea una planificación apropiada del espectro.

En este trabajo, se presenta una revisión profunda de los trabajos de investigación para estrategias de asignación de canales. Luego, se formaliza la asignación de canales como un problema de minimización del nivel de interferencia y se proponen tres diferentes maneras para optimizar la asignación de canales entre los Puntos de Acceso participantes con el objetivo de mejorar la performance de la red. Los algoritmos propuestos pueden clasificarse como local y no-coordinado, coordinado y distribuido, y centralizado. La solución local y no- coordinada se comparta de manera aceptable en el prototipo pero presenta problemas de oscilación que se aborda con una técnica de control por retro alimentación. Finalmente, este trabajo presenta una evaluación de las estrategias, en un prototipo y en un entorno de simulación. En el prototipo se demuestra el despliegue práctico de las soluciones y se llega a la conclusión que la implementación local y no-coordinada es digna de ser considerada como una buena estrategia para el problema de asignación de canales cuando los Puntos de Acceso trabajan en forma aislada. En la simulación, se prueban la escalabilidad de las soluciones coordinada y centralizada, y se muestra que pueden ser desplegadas en redes con más de treinta Puntos de Acceso y como resultado, se concluye que la implementación centralizada es la mejor estrategia para realizar decisiones de optimización para la asignación de canales en redes conectadas.

Contents

List of Figures

List of Tables

Glossary

ACI Adjacent Channels Inteference produced by transmissions on adjacent or partially overlapped channels. 7

ACL Access control list. 23

AP Access Point in 802.11. 9

BSS Basic service set. 33

CCI Co-channel Interference is caused by undesired transmissions carried out on the same frequency channel. 7

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance. 8

DCOP Distributed constraint optimization problem. 31

DFS Dynamic frequency selection. 15

DisJ DisJ (Simulator for Distributed Algorithms). 76

DPOP Distributed pseudotree optimization procedure. 31

DPR Distributed policy repository. 26

DSSS Direct-Sequence Spread Spectrum is a modulation technique. 10

FSM Finite state machine. 39

LCC Least congested channel strategy for channel assignment. 36

LDAP Lightweight directory access protocol. 26

LUPA The Lua Policy Agent (LuPA), in the RAN System, groups the policy-related services or functionalities. It includes a PDP and an EP. 39

MAC Media access control. A media access control address (MAC address) is a unique identifier assigned to network interfaces for communications on the physical network segment. 8

MANET Mobile ad-hoc network. 26

PBNM Policy-Based network management. 39

PDP Policy Decission Point in a PBNM. 39

PEP Enforcement Point in a PBNM. 39

RAN The RAN system aims to be a complete rule-based, distributed system specially designed and implemented to enable autonomic behavior on very constrained devices. The RAN system was developed to serve the objectives of Rural Ambient Networks project. 37

RF Radio frequency. 10

RMOON A service that monitors the state of a device, and triggers trap notifications when certain conditions occur. 39

RNR A distributed notification service for nodes communications in RAN System. 39

RSSI Received signal strength indicator. 22

SLA Service level agreement. 16

WLAN Wireless Local Area Network. 8

Chapter 1

Introduction

This thesis studies the use of High-Density Wireless Networks, particularly focusing on the assignment of frequencies seeking to minimize the level of interference. From a practical perspective, these kinds of networks are formed when there is a high concentration of users, as in a classroom or a convention center. In these environments each user, carrying their personal wireless device such as laptops, PDAs, and mobile phones, searches for wireless connectivity with his/her own device. As the number of users increases, the network performance in these contexts enters into a process of significant degradation. To address the challenge of demanding connectivity by the users, usually, network administrators just add more Access Points with the aim to keeping the quality of service perceived by the users, leading to what Akella et al [16] called *Chaotic Wireless Deployments*.

In educational institutions, cafés, airports, convention centers and homes, wireless LANs are now one of the most important network access technologies in the Internet today. Although many technologies and standards for wireless LANs were developed, one particular class of standards has clearly emerged as the winner: the IEEE 802.11 wireless LAN, also known as WIFI. The low cost and the ease of deployment of WIFI devices, as well as the need to support high bandwidth applications over 802.11 Wireless Networks has led to the emergence of high density 802.11 networks in urban areas and enterprises.

For this reason and because today's user devices, such as mobile phones and tablets, are able to connect to WIFI networks, this thesis will focus mainly on the IEEE 802.11b/g protocol. But the 802.11b/g networks have the following limitations: "*Radio resource management is often limited to a small number of non-overlapping channels, which leaves only three possible channels in the 2.4GHz band used in IEEE 802.11b/g*". This means that the network capacity may be reduced due to the interference between simultaneous neighboring transmissions. This problem can deteriorate if we augment the number of Access Points without an appropriate spectrum planning.

In consequence, the small number of non-overlapping channels and the deployment of an increasing number of APs (to serve a high concentration of users) lead to serious interference problems that need to be addressed. Usually, two types of interference are distinguished [42]:

- Co-Channel Interference (CCI), which is caused by undesired transmis-

sions carried out on the same frequency channel.

- Adjacent Channel Interference (ACI), produced by transmissions on adjacent or partially overlapped channels.

As a result of higher number of Access Points and the growth in the amount of users connected to them, transmitting at the same time, the undesired effect of radio interference becomes more problematic, affecting the network performance. This contrasts with the original goal of augmenting the Access Point deployments to address users requirements. Hence, the assignment of channels to this potentially large set of APs needs to be carefully coordinated, otherwise the broadcast nature of WLANs can lead to a serious performance degradation of the user's connections.

In this chapter, we will present the definition of a high-density wireless network. Then, we will extend on the concepts of co-channel interference and adjacent channel interference. Next, we will review the mechanisms to address the interference problem and we will focus on the channel assignment due it is the main factor to handle interference. Finally, we will present the thesis objectives and contributions.

1.1 What are High Density Wireless Networks?

There is not a precise definition of what a "High Density Wireless Network"is in formal terms. Indeed, each network administrator has forged their own definition drawing upon their own experience. For example lets review how the industry refers to the term:

- Aruba Networks[39] defines High-density (HD) WLANs as radio frequency RF coverage zones with a large number of wireless clients and Access Points in a single room.

- Meru Networks[8] defines a High-density environment as the one that supports over 500 wireless devices such as a convention center, a campus auditorium or a big hotel lobby.

- For Cisco[21], High-density WLAN design refers to any environment where client devices will be positioned in densities greater than coverage expectations of a normal enterprise deployment.

Beyond differences, all these definitions share a common concept: the high concentration of users in high-density environments such as large meeting rooms, lecture halls and auditoriums, convention center, meeting halls, hotel ballrooms, press areas at public events, concert halls and amphitheaters, airport concourses, financial trading floors, casinos, stadiums, arenas, and ballparks.

The proliferation of wireless-enabled personal and enterprise mobile devices presents challenges for designing and deploying a wireless network. Lets imagine the following scenario: we gather 1000 people in an auditorium, each person using his/her own wireless device searching for good quality connectivity. In such scenario one Access Point clearly will be overloaded and the throughput will be very poor. The AP has the physical capacity to handle up to 2048 MAC addresses [7], but, because the AP uses a shared medium and acts as a wireless

hub, the performance of each user decreases as the number of users increases on an individual AP. Similarly to the Ethernet segment, in this technology the nodes of a 802.11b/g share the medium using, in both cases, a CSMA/CA (carrier sense with collision avoidance) medium access control scheme.

In cases of high connectivity demand the deployment of various Access Points is a recurring pattern, although this solution is not enough to gain performance. Even supposing that users connect to different Access Points, the level of interference between transmissions can lead to poor performance if no appropriated spectrum planning is employed. Therefore we need to find new strategies to minimize the level of interference in this type of networks. To address this problem and examine the different types of interference we devote the following section.

1.2 Interference Types

Usually, two types of interference are distinguished: co-channel interference, which is caused by undesired transmissions carried out on the same frequency channel; and adjacent channel interference, produced by transmissions on adjacent or partially overlapped channels [37]. Lets look at each type in more detail.

1.2.1 Co-Channel Interference

Co-channel interference occurs between two access points (APs) that are on the same frequency channel (see figure 1.1). This type of interference can severely affect the performance of your wireless network. The spectrum that is available for the deployment of WiFi is limited. For example, in the 2.4GHz band, there is just 79MHz of spectrum in the United States. Given that your IEEE 802.11g devices use a 20MHz channel, you have room for three non-overlapping channels. You will need to reuse the frequency channels when you deploy your APs.

Co-channel interference is more problematic when you deploy your wireless network in scenarios that require a denser deployment of APs. Denser deployments mean that your APs are closer together. This creates a greater potential of two devices that transmit on the same frequency, which will be close enough to cause significant interference to each other's signals. The two solutions for co-channel interference are, first, the use of a different, non overlapping channel for each of the APs, and second, moving the wireless network far enough apart that the access points cells do not overlap.

1.2.2 Adjacent Channel Interference

Adjacent channels are those channels within the RF band being used that are, in essence, side-by-side. For example, channel one is adjacent to channel two, which is adjacent to channel three, and so on. These adjacent channels overlap each other because each channel is 22 MHz wide and their center frequencies are only 5 MHz apart. Adjacent channel interference happens when two or more access points using overlapping channels are located near enough to each other that their coverage cells physically overlap. Adjacent channel interference can severely degrade throughput in a wireless network.

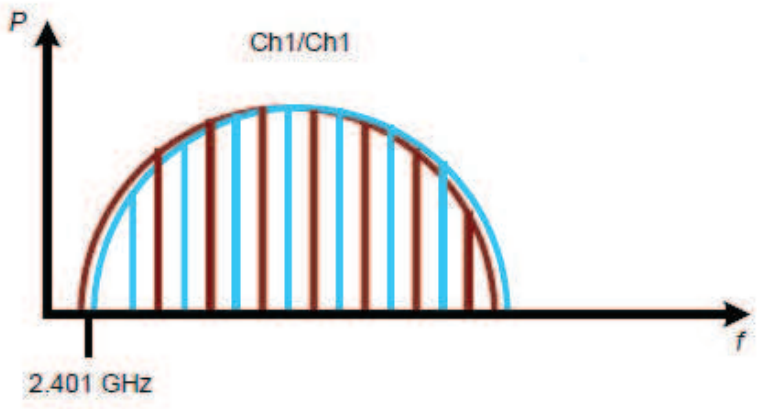

Figure 1.1: Co Channel Interference. Source [37].

It is especially important to pay attention to adjacent channel interference when allocating access points in an attempt to achieve higher throughput in a given area. Access points on non-overlapping channels can experience adjacent channel interference if there is not enough separation between the channels being used, as illustrated in Figure 1.2.

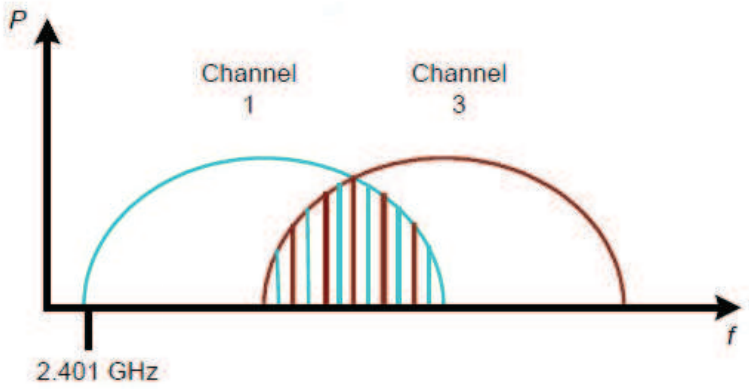

Figure 1.2: Adjacent Channel Interference. Source [37].

Commonly two solutions are used for the problem with adjacent channel interference. The first solution is to separate those access points that are transmitting on adjacent channels far enough to reduce the overlapping between their respective cells, or turn the transmit power down on each access point low enough that the cells do not overlap. The second solution is to use only channels that have no overlap whatsoever. For example, in systems with direct-sequence spread spectrum (DSSS) modulation technique the use of channels one and eleven would accomplish this task.

1.3 Techniques to Minimize Interference

There are three main approaches to address the challenge of minimize the level of interference between Access Point [20] that presents one of the interference described in previous section:

1. Intelligent frequency allocation across APs
2. Load-balancing of user affiliations across APs
3. Adaptive power-control for each AP.

One approach is that power control can be used to mitigate interference in high density environments [31]. The benefits of power control for interference mitigation in the context of cellular networks have been widely studied. However, unlike cellular networks, today's commercial 802.11 APs support power adaptation on a per-cell basis instead of a per-client basis. So there is no straightforward way to apply those concepts to 802.11 networks. One of the main challenges when dealing with power control is that they can lead to throughput starvation due to the introduction of asymmetric links. The symmetry is preserved only when the product of transmission power and the Clear Channel Assessment (CCA) threshold for each AP is constant throughout the network. The proposed algorithm in this topic enable the exchange of appropriate information (by means of Beacon frames) between APs to allow them to optimally tune the transmission power and the CCA thresholds. The algorithm developers do this by changing the MAC drivers implementation.

Another approach to improve dense networks performance is the association of users to APs [31] User devices are programmed to associate with the AP with the strongest received signal strength. This leads to scenarios where some APs have very few users, while other APs are overloaded with many users. In general, the problem with User Association strategy is that the algorithms requires modifications in the AP driver and firmware, as well as in the client driver.

Finally, the last approach is the channel assignment strategy. The performance of a dense wireless local area network depends on the channel assignments among neighboring access points (APs). The limited number of non-overlapping channels may lead to severe interference scenarios if no appropriated spectrum planning is employed. For example, 802.11b/g operates in the frequency range of 2.4 GHz to 2.485 GHz, within this 85 MHz band, 802.11b/g defines eleven partially overlapping channels. Any two channels are non-overlapping if and only if they are separated by four or more channels. In particular, the set of channels 1,6,11 is the only set of three non-overlapping channels.

Considering the three approaches above mentioned this thesis studies the channel assignment strategy as a way of minimizing the interference level in dense wireless networks. The next section is dedicated to examine channel assignment problem as it is found in 802.11 wireless networks.

1.4 Channel Assignment

One way to reduce the interference problems of high density wireless LAN deployments is to automate as much as possible the channel assignment strategy.

[25, Physical Layer Selection and Design section in chapter 25]

Nowadays several products offer the ability to automatically lay out channels along the frequency spectrum. Furthermore, the available technology allows some devices to continuously monitor the radio space in order to adjust the channel settings dynamically. In this respect one of the most common approaches is to layout channels based on physical measurements that assess the quality of frequencies usage.

Embracing a different approach network administrators, place access points based on the expected number of users, mounting convenience and environmental constraints. In this cases, when the network is powered on, then APs communicate with each other through a wired network searching for the optimal channel assignment.

In this section we will focus on 802.11 wireless networks, examining the frequency ranges for both 802.11b/g and 802.11a standards. There is a clear distinction between a 2.4 GHz (802.11b/g) and a 5 GHz (802.11a) channel's layout. In Figure 1.3 you can see a graphical representation of WiFi channels in the 2.4 GHz band.

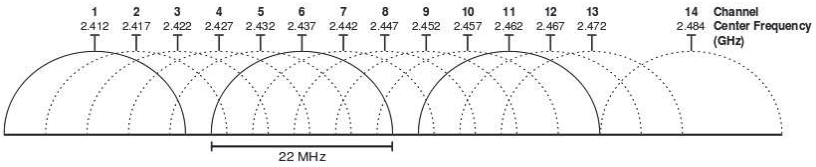

Figure 1.3: 802.11b/g Channels in 2.4GHz Band. Source [44].

The 2.4 - 2.485 GHz band is divided into 13 channels spaced 5 MHz apart, with channel 1 centered on 2.412 GHz and 13 on 2.472 GHz (to which Japan added a 14th channel 12 MHz above channel 13 which was only allowed for 802.11b). 802.11b was based on DSSS with a total channel width of 22 MHz and did not have steep skirts. Consequently only three channels do not overlap. Moreover, many devices are shipped with channels 1, 6 and 11 as preset options. Even though 802.11g channels are 20 MHz wide, and have space for four non-overlapping channels (1, 5, 9 and 13), due to the use of *orthogonal frequency-division multiplexing* signaling, this is not the case as per 17.4.6.3 Channel Numbering of operating channels of the IEEE Std 802.11 (2012) which states "In a multiple cell network topology, overlapping and/or adjacent cells using different channels can operate simultaneously without interference if the distance between the center frequencies is at least 25 MHz". For this reason, both standards, 802.11b and 802.11g, have three non overlapping channels. Then, in a practical deploy, part of the site survey is to lay out coverage areas in a way that minimizes channel overlap.

The inability to perform a channel layout with three channels is not surprising. In mathematics, the general result is known as the *Four-Color Map Theorem*. Map makers discovered in the mid-19th century that an arbitrary two-dimensional map can be filled in with four colors. Unfortunately, 802.11 b/g networks have only three non-overlapping channels. Maps where adjacent regions share a color are unsightly. It is often possible to minimize channel

overlap by careful AP placement or the use of external antennas.

In contrast, 802.11a has one major advantage when laying out your network. The major advantage that 802.11a has over 802.11b/g is that it contains 12 or 13 non-overlapping channels, while 802.11b/g has only 3 channels making it more likely to have interference from adjacent APs. With this being said we can conclude that for high capacity, high-density networks, the use of 802.11a will get better results.

But 802.11a has two major drawbacks compared with 802.11b/g. First, the 5 GHz band has a shorter wavelength. Higher-frequency signals will have more trouble in propagating through physical obstructions encountered in an office (walls, floors, and furniture) than those at 2.4 GHz. Second, there are more country controls in this band, for example in 2007 the FCC (United States) began requiring that devices operating on 5.250 - 5.350 GHz and 5.47 - 5.725 GHz must employ dynamic frequency selection (DFS) and transmit power control (TPC) capabilities. This restriction was implemented to avoid interference with weather-radar and military applications.

Finally, 802.11b/g networks have become more popular than 802.11a, because most personal devices support 802.11b/g. For the above mentioned reasons this thesis will focus on 802.11b/g networks and the constraints of having only three non overlapping channels.

A strategy to distribute channels between APs is needed to improve wireless networks performance. A channel assignment strategy can be classified as static or dynamic. In the static approach, a channel is assigned to the AP permanently or for a long period. The dynamic strategy assumes that the network interface can switch the communication channel using a coordinated or uncoordinated approach.

This thesis seeks to provide a good solution for the channel allocation problem using a self-managed network while leaving the client devices configuration untouched. To achieve this goal we formalized the channel allocation as a minimization problem of the interference level. Then, we propose three different strategies to assign channels between APs with the aim to improve network performance. The algorithms that we propose can be classified as local and uncoordinated, coordinated and distributed, and centralized. Finally, to test the proposed strategies we conducted a testbed and simulated a dense network environment, where the network minimize the interference by assigning the best channel for each AP and we did this without changing the client devices.

1.5 Objectives

The high concentration of users in any high-density environment presents challenges for designing and deploying a wireless network. Lecture halls, hotel ballrooms, and convention centers are common examples of spaces with this requirement. As the usage of corporate 802.11 wireless networks (WLANs) grows, network performance is becoming a significant concern. We will look for a system for improving the performance of enterprise wireless networks using a dense deployment of access points (APs).

Moreover, the performance of a wireless local area network depends, in part, on the channel assignments among neighboring access points (APs). The limited number of non-overlapping channels may lead to severe interference scenarios if

no spectrum planning is performed. In consequence, our goals of this work are:

- design and implement different strategies to improve the usage of wireless spectrum in the context of wireless local area networks (WLANs) using new channel assignment strategies among interfering Access Points (APs).
- test the strategies in a testbed and in a simulation environment to evaluate the behaviors and compare the results.

1.6 Contributions

The main contributions of this thesis will be:

- An in-depth review of the different channel assignment strategies. We will be reviewing: industry and academic studies, static and dynamic approaches, coordinated or uncoordinated strategies.
- Optimize channel assignment in three different manners: in an uncoordinated and distributed scenario, in a coordinated but still distributed scenario and in a centralized scenario where one node has a view of the whole network. The proposed strategies were designed for a deployment environment such as lecture hall or hotel ballrooms, this means that a typical deployment will consist of less than 50 access points.
- Apply a feedback control technique to address some limitations of the uncoordinated strategy. The uncoordinated solution present oscillatory issues that must be handle to be useful. To bound the number of oscillations that can occur in a given time period we apply a feedback control scheme where we dynamically adjust a *threshold* value with the aim to control the amount of channel switches for the aforementioned time period.
- Deployment of the strategies on a testbed to study the feasibility of the work and to get some real metrics.
- Evaluate the systems in simulations with the aim of measuring scalability and to determine the best strategy as the number of Access Point increase.

1.7 Thesis Structure

This work is organized as follows. In the next chapter we will present a detailed description of the state of the art in this field. Then in chapter 3 we present three different approaches for channel assignment among APs. Chapter 4 presents a method to handle oscillatory issues encountered in the uncoordinated strategy. Chapter 5 covers the evaluation of the proposed strategies. And finally, chapter 6 presents conclusions and future works.

Chapter 2

State of the Art

The need to improve dense Wireless Networks performance has been recognized by both the research community as well as the industry. In the first part of this chapter we will examine the industry's approach to this problem based on the guidelines developed by major network providers. Secondly we will conduct an in depth review of the academic research in this field.

2.1 Planning-based Approach

In this work we will describe some of the industry's approach to deal with High Density wireless networks.

Aruba Networks has a reference guide (VRD - Validated Reference Design) for high density wireless network [39]. The purpose of the guide is to describe the best practices for implementing coverage zones with high numbers of wireless clients and APs in a single room such as lecture halls and auditoriums. The guide presents a HD WLAN Capacity Planning Methodology that follow these steps:

1. **Choose a capacity goal:** The first step is to pick an application-layer throughput target linked to the seating capacity of the auditorium.
2. **Determine the usable number of channels:** For each band, decide how many non overlapping channels are usable for the HD WLAN. Use a database of regulatory information (included in the guide), augmented by site-specific decisions such as whether or not Dynamic Frequency Selection (DFS) channels are available.
3. **Choose a concurrent user target:** Determine the maximum number of simultaneously transmitting clients that each AP will handle. Use a lookup table based on test data supplied by Aruba. You must do this for each radio on the AP.
4. **Predict total capacity:** Use the channel and concurrent user count limits to estimate the maximum capacity of the auditorium using lookup tables supplied by Aruba.
5. **Validate against capacity goal:** Compare the capacity prediction with the capacity goal from step 1. If the prediction falls short, you must start

over and adjust the goal, concurrent user limit, or channel count until you have a plan that you can live with. For large auditoriums over 500 seats, you should be prepared to accept a per-client throughput of 500 Kbps or less, assuming a 50/50 mix of .11n and .11a stations and nine usable channels.

At the heart of this guide, there is Aruba's Adaptive Radio Management (ARM) technology. Enterprise WLANs use automatic channel selection algorithms because static channel assignment is cumbersome to design and not responsive to real-world dynamic RF conditions. Then, Aruba's ARM technology uses a dynamic channel planning algorithm in which each access point makes decisions independently by sensing its environment and optimizing its local situation. The algorithm is designed so that this iterative process converges quickly on the optimum channel plan, for the entire network, without requiring a central coordinating function.

Each access point periodically scans all channels for other access points, clients, rogue access points, background noise, and interference. During the scan, the access point is not servicing its own associated clients, so scanning can be suspended for situations such as clients in power-save mode or active voice calls. Following scanning, two figures are derived: the *coverage index*, and *interference index*. These are used to calculate the optimum channel and transmit power for the access point. The interference index is a single figure representing both Wi-Fi channel activity and non-Wi-Fi channel noise and interference. When the interference index on the current channel is high compared to other channels, the access point will look for a better channel, generally choosing the channel with the lowest interference index. This not only avoids non-Wi-Fi interference, but minimizes co-channel interference, as other access points on the same channel contribute to the interference index.

The coverage index comprises the number of access points transmitting on a particular channel, weighted by their signal strengths as measured by the access point. The ARM algorithm maximizes and equalizes coverage indices for all channels across all access points, and this is the primary parameter used to control an access point's transmit power, within configured limits. An ARM dynamic channel planning algorithm optimizes the RF plan by making best use of the available spectrum, avoiding interference while also meeting the desired coverage parameters.

Aruba's guide the algorithm is embraced by many others techniques. They recommend that once you planned the capacity you must follow others design topics, like *coverage strategy* (a specific method or approach for locating APs inside a wireless service area), *choosing access points and antennas*, *aesthetic considerations*, among others.

Similarly, Cisco[21] has a design guide that provides engineering guidelines and practical techniques for designing, planning, and implementing a Wireless LAN within a high density environment. As Aruba, Cisco mention a series of step for a successful high user density WLAN. The steps are:

1. Plan: Determine application and device requirements such as bandwidth, protocols, frequencies, service level agreement (SLA), etc.

2. Design: Determine density, cell sizing, antennas, coverage, site survey, etc.

3. Implement: Install, test, tune, establish baseline, etc.

4. Optimize: Monitor, report, adjust, review baseline for SLA

5. Operate: Cisco Wireless Control System (WCS) monitoring, troubleshooting tools, capacity monitoring and reporting tools, etc.

During design phase the guide have many sections devoted to channel assignments issue. After discussing why Co-Channel Interference (CCI) is important in high-density WLANs they present some design steps to mitigate the effects of CCI.

They focus on two technologies to help deal with channel issues.

The Cisco WCS and controllers make monitoring co-channel interference and identifying the responsible AP or APs a fairly straightforward exercise. Cisco Radio Resource Management (RRM) algorithms are centralized and are a network-wide resource that continuously evaluates every single AP in the RF network to determine its relationship to every other AP in the system. It does this through the use of over the air (OTA) measurements and observations. Knowing how well other APs can hear a selected AP is a very useful feature when considering or planning a high-density WLAN deployment. Using Cisco WCS, it is possible to evaluate how well APs can hear one another-independent of a channel. This information is shown in a graphic display that shows not only how APs are effecting each other on a particular map, but also how other APs that are not on the map can impact a WLAN as well.

The wireless LAN controller maintains two lists of APs, both transmit and receive (TX and RX) neighbors that indicate how other APs hear a selected AP and how a selected AP hears other APs. This can be viewed using the Wireless LAN Controller (WLC) Configuration Analyzer tool and used to tune the resulting network and identify sources of RF as the APs themselves see it. Since this observation is based on OTA metrics and not based on predictive modeling, these values are independent of the antenna and AP combination.

Both guides are *a serie of step* for network administrator or network installer. They not only based on proprietary technologies but also they are based on a careful planning. In those work they plan (cisco) or set a capacity goal (Aruba). How many users, how big is the room, how many seats, and so on. In both guides there are recommendations on how to choose APs, (if needed) which kind of antennas and where to locate APs taking into account aesthetic considerations.

All these management solutions are customized for proprietary hardware and use proprietary algorithms to achieve their ends, making them both hard to validate and hard to compare with other algorithms. I think the most important contributions of these guides are they put together a number of issues to deal with high density wireless network. I choose them because it touched most of the topics about how to design a High Density WLANs from channel assignment between APs to transmission power. And that both works discussed as the main topic the channel assignment strategy.

So this sentence can summarize the importance they put on channels: *The number of allowed non overlapping channels is the primary capacity constraint on an HD WLAN.*

2.2 Academic Related Work

Academic approach to the channel assignment strategy can be divided in different ways: genetic algorithm, distributed algorithms, algorithms with the help of client, and centralized algorithms.

Techniques that solve management problems for wireless LANs fall into two broad categories: static optimization and dynamic optimization. The academic research can be classified into those categories, for example genetic algorithm are intended for a static optimization whereas the other proposed techniques are mainly dynamic optimization. Lets review each approach in the following subsections.

2.2.1 Genetic Algorithms

FAP web [3] is a web-site devoted to Frequency Assignment Problems (FAPs) in wireless communication networks. In this website they classified the frequency assignment problem for fixed channel assignment schemes in four different flavors.

In the minimum order frequency assignment problem (MO-FAP), we have to assign frequencies in such a way that no unacceptable interference occurs, and the number of different used frequencies is minimized. In the minimum span frequency assignment problem (MS-FAP), the problem is to assign frequencies in such a way that no unacceptable interference occurs, and the difference between the maximum and minimum used frequency, the span, is minimized. In case all assignments contain some unacceptable interference, we can decide to find a partial assignment that minimizes the overall blocking probability. In the minimum blocking frequency assignment problem (MB-FAP), the problem is to assign frequencies in such a way that no unacceptable interference occurs and the overall blocking probability of the network is minimized. Besides the approaches in which the maximum interference level is minimized, another approach is given by the minimization of the total sum of interference levels. In the minimum interference frequency assignment problem (MI-FAP), we have to assign frequencies from a limited number of available frequencies in such a way that the total sum of weighted interference is minimized.

All of these flavors are formally modeled using graphs notations. As one of the FAP website major contribution is to provide a formal definition of the frequency assignment problem we will now present the mathematical models they have developed.

The standard representation of a FAP is by means of a graph G=(V,E), the interference graph or constraint graph. Each connection is represented by a vertex $v \in V$. The available channels or frequencies for a vertex are denoted by the set $D_v \subseteq D$. Let c_v denote the required number of frequencies for connection $v \in V$. Two vertices v and w for which the corresponding connections may interfere for at least one pair of frequencies, are connected by an edge $\{v, w\} \in E$. For each pair of frequencies $f \in D_v$ and $g \in D_w$ we penalize the combined choice by a measure depending on the interference level. This penalty is denoted by p_{vwfg}. The interference between two frequencies $f, g \in D_v$ assigned to the same vertex v can be modeled in the same way: an edge $\{v, v\} \in E$ and penalty p_{vvfg}. Another way to model this, is by replacing v by c_v vertices and additional edges between all of them. Some instances deal with a frequency plan in which

changes are considered, to reduce interference. This reduction should take place under minimal changes of the total frequency plan, thus changes in the plan are penalized per change as well. This is modeled with additional penalties on the frequencies to be chosen for each vertex: the choice of frequency $f \in D_v$ costs q_{vf}.

In the FAP web there is a digest of frequency assignment literature where for each of the flavors above there are a list of papers from more than a decade that propose solutions using heuristic search techniques. The techniques are Greedy, Integer Linear Programming, Tree Search, Simulated Annealing and Genetic Algorithms among others.

There are various paper to look at like [29, 22, 43] but we will look just to one example that use Genetic Algorithms. In [30] a heuristic algorithm is proposed and analyzed after that the authors prove that the problem for the channel assignment problem for the 802.11 network is NP-complete. The authors define an objective function like:

$$U_i \equiv \rho_i + \sum_{k=1}^{N} X_{ik}[\sum_{j \in C_i(1)} \rho_j X_{jk} + \sum_{(m,n) \in C_i(2)} \rho_m \rho_n X_{mk} X_{nk}]$$

This definition can be interpreted as follows. The first term on the right hand side is the offered load associated with AP i. The first summation term inside the brackets represents the total traffic load of all class-1 interfering APs that are assigned with the same channel as AP i. This is so because according to the CSMA protocol and the detection threshold α in use, AP i senses channel busy when any one of its class-1 interferers transmits on the same channel. Similarly, the last summation term represents the effective channel utilization due to class-2 interferers if they use the same channel.

The objective function for the channel assignment is to minimize the utilization at the most stressed bottleneck AP:

$MinimizeMax\{U_1, U_2, ..., U_M\}$

The algorithm is outlined as follow:

1. Generate a random, initial channel assignment for the network, which is treated as the best assignment obtained so far. Let the maximum effective channel utilization for the assignment be denoted by V (i.e., $V = max\{U_i\}$).

2. Based on the best assignment, identify the AP (say i) with the highest effective channel utilization. In case of tie, one such AP i is chosen randomly as the "bottleneck".

3. For the bottleneck AP i, identify its current assigned channel, say k. For each available channel n from 1 to N with n≠k and each co-channel AP (say j) in $C_i(1)$ (i.e., those APs in the set that have been assigned with channel m), temporarily modify the channel assignment by re- assigning only AP j with channel n. Based on (3), re- compute the maximum effective channel utilization, denoted by W_{jn}, for the new assignment. After completing such testing for all such n and j, let W be the minimum among all the W_{jn}'s.

4. Compare W with V and perform the following:

   (a) If $W < V$, then replace V by W and record the associated new assignment as the new best solution (i.e., to finalize the channel change

for one AP that minimizes the objective function the most − a greedy step). Continue with Step 2.

(b) If $W = V$, then with a pre-specified probability δ, replace V by W and record the new assignment as the best solution. Continue with Step 2

(c) If $W > V$, a local optimum has been reached (i.e., the best assignment obtained so far is the local suboptimal solution). Continue with Step 5.

5. Repeat Steps 1 to 4 with a number of random, initial assignments. The final solution is chosen to be the best, according to (4), among the local suboptimal assignments.

6. Test if constraints (5) for all APs are satisfied for the final assignment. If so, the final assignment is feasible. Otherwise, it is considered that no feasible solution exists for the network under consideration.

The drawback of this methodology is that it is for fixed channel assignment schemes. You must run the algorithm beforehand the network is operational and then assign each AP the corresponding channel. Also, the genetic algorithm does not get the optimum just a good value that approximate to the optimum. Can be costly on the computer resource to evaluate it.

2.2.2 Client-Assisted Channel Assignment Algorithms

Yue et al [46] proposed a completely distributed channel assignment scheme for uncoordinated WLANs. They study are based under the following premises:

- *Network nonspecialists.* Unlike WLANs managed by certified system administrators, uncoordinated WLANs are usually set up by inexperienced and independent users who are not knowledgeable in network configuration. These users very likely expect the devices to be plug-and-play (i.e., self- configurable). They cannot be expected to know how to configure the appropriate channel to minimize interference in their neighborhoods.

- *Unplanned topology.* In managed WLANs, such as campus or enterprise networks, the system administrators can calculate where the APs should be placed in order to minimize co-channel interference and achieve high throughput. In uncoordinated WLANs, on the other hand, APs are placed without any concerted planning; some areas may have high AP density and hence may experience high interference and poor performance.

- *AP independence.* Due to the absence of central management and the prevalence of inexperienced users, configuration for uncoordinated WLANs should be as simple and automatic as possible. The APs of different WLANs should operate independently without any direct communication.

They present the problem formulation and show that it is NP-hard. They then propose CACAO, a novel distributed channel assignment scheme for uncoordinated WLANs. The APs auto-configure their channels depending on their local traffic information. The approach, termed Client-Assisted Channel Assignment Optimization (CACAO), makes use of client feedback to perform channel

assignment. Such feedback may be obtained using the proposed IEEE 802.11k standard for radio resource management, which defines a series of measurement requests and statistical reports between an AP and its clients.

To formulate the problem the authors model the network as graph $G = (V, E)$, where $V = \{ap_1; ap_2; ...; ap_n\}$ is the set of n APs. In this model, clients associated with an AP are grouped with the AP and collapse into a single (super) node for the purpose of interference accounting. Let $W(ap_i, ap_j)$ be the potential interference level between AP_i and AP_j. The larger the weight $W(ap_i, ap_j)$ is, the higher the interference between AP_i and AP_j will be if they use the same channel.

Next they define a Boolean function interference map $I(ap_i, ap_j)$ for each edge, where

$$I(ap_i, ap_j) = \begin{cases} 1 & \text{if } ap_i \text{ and } ap_j \text{ are on the same channel;} \\ 0 & \text{otherwise} \end{cases}$$

One may consider that the product of $W(ap_i, ap_j)$ and $I(ap_i, ap_j)$ indicates the total interference level between ap_i's BSS and ap_j's BSS. Given $G(V, E)$ and $W(ap_i, ap_j)$, channel assignment C is a mapping $C : V \to \{1...k\}$, where k is the total number of non overlapping channels (e.g., k = 3 for the IEEE 802.11b/g standard). The channel assignment problem is to find a mapping C such that the total interference level is minimized, i.e.,

$min_C L(G, C) = \sum_{\forall e=(ap_i, ap_j) \in E} W(ap_i, ap_j) * I(ap_i, ap_j)$

Finally, the authors formalize this proving a theorem

`Theorem.` Given a weighted undirected graph $G = (V, E)$, with n vertices, $V = \{v_1; v_2; ...; v_n\}$, and a weight function $W(v_i, v_j)$. Let $C : V \to \{1...k\}$ be a k-coloring of G. The problem of finding C that minimizes $L_G = \sum_{\forall e=(ap_i, ap_j) \in E, C(v_i)=C(v_j)} W(v_i, v_j)$ is NP-hard.

The proposed distributed algorithm is as follow: Let $ap_i.c$ denotes the operating channel of ap_i . Let $G(ap_i) = (V(ap_i), E(ap_i))$ be the subgraph of G containing only vertex ap_i and all its directly connected neighbors.

The CACAO algorithm for each ap_i has two phases:

1. Initialization. Initial assignment
   (a) $ap_i.c \leftarrow rand(k)$
2. Optimization. Repeated for each AP
   (a) $Gatherstatistic()$
   (b) $c_t = ComputeInterference()$
   (c) $SwitchTo(c_t)$

The algorithm is self-explanatory. From the algorithm you can see that they proposed a completely distributed channel assignment scheme for uncoordinated WLANs. Their model is based on a single (super) node, representing channel condition information in an AP, gathered by its associated clients. We will see later in the Graph Coloring based section a strategy named Hsum and the CACAO algorithm employs the Hsum strategy and the key contribution is the

use of a modified channel metric based on client traffic. This algorithm does not employ coordination among APs in order to improve the solution quality.

Before the CACAO strategy there was another client driven approach for channel management in Wireless LANS.

Mishra et al. [33] proposed an efficient client-based approach for channel management (channel assignment and load balancing) in 802.11-based WLANs that lead to better usage of the wireless spectrum. This approach is based on a conflict set coloring formulation that jointly performs load balancing along with channel assignment.

The strategy tries to minimize interference at wireless clients by implicitly modeling its location and distribution with respect to the APs. The algorithm is centralized and is well suited for centrally managed wireless networks.

2.2.3 Centralized Algorithms

Another approach to solve the problem of dense deployment of APs is to use centralized algorithms where one entity makes the decisions and communicate those decisions to the rest of the entities. In this regards there are two interesting works.

The first one is named DenseAP [36] by Murty et al at Harvard University and Microsoft Research.

The design is very simple as you can see in Figure 2.1.

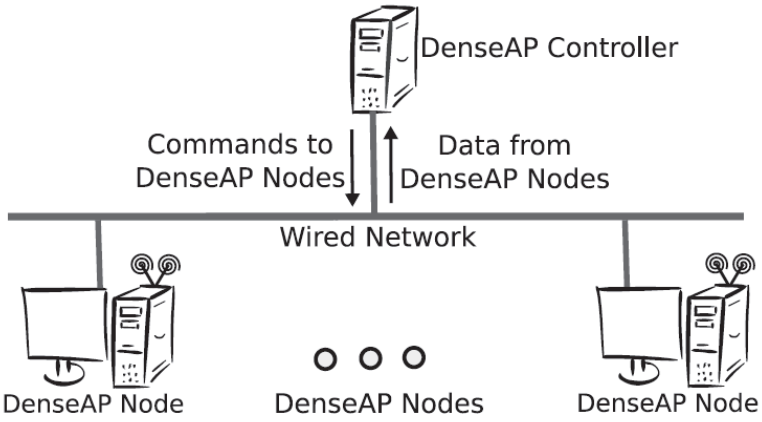

Figure 2.1: Overall Architecture of the DenseAP System. Source [36].

The system consists of several DenseAP nodes (DAPs) which provide wireless service and a DenseAP controller (DC) which manages the DAPs. A DAP is a programmable Wi-Fi AP connected to the wired network. Each DAP periodically sends summaries to the DenseAP controller comprising of a list of associated clients, their traffic pattern summaries, RSSI values of a few packet samples from their transmissions, current channel conditions, and reports of new clients requesting service from the network. They classify DAPs into two categories: they refer to DAPs that do not have any clients associated with them as passive; those that have at least one associated client are called active.

The DC manages the DAPs. The periodic reports sent by the DAPs provide the DC with a global view of the network activity. Using this global view, the DC selects the right DAP for a client, allocates channels to DAPs, performs load balancing when needed, handles client mobility, and deals with DAP failures.

One of the interesting point in the proposed algorithm is in the association a new client process. When a new client first appears in the network, it scans on all channels and sends out probe requests. Because this client has not yet been added to the ACL (list of clients associated with an AP) of any DAP, all DAPs that hear the probe requests simply report them to the DC. To calculate reasonable signal strength estimates, the DC waits for a short while (10-30 seconds in the current implementation) after the first report of a new client is received. During this interval it continues to collect reports of probe request packets from DAPs. At the end of this interval the DC calculates the average signal strength of all the probe request frames seen by each DAP.

If two or more APs see the client, then the DC use a metric called *Available Capacity* (the metric is based on the transmission rate the client and the DAP can use to communicate with each other; and how busy the wireless medium is in the vicinity of the client and the AP) to decide which AP must respond the client. Assume two DAPs, A and B hear probe requests from a client M. Assume that A is active i.e. it already has other clients associated with it, whereas B does not (passive). After calculating $AC(A)$ and $AC(B)$ the DC then compares those values and picks the higher of the two. If they are equal, it decides in favor of B, since B has no clients associated with it. If the DC picks A, it adds M's mac address to A's ACL. If it picks B instead, it first instructs B to stop scanning and to stay on channel Y (assuming that DAP B has recently seen the highest available free air time on channel Y) It then adds M's mac address to B's ACL.

Note two key aspects of this algorithm. First, they never move existing clients to another DAP as a result of a new client association. Second, DAPs are only assigned channels on an on-demand basis, as part of the association process. A DAP is assigned a channel only when a client in its vicinity requests service from the network. When a DAP becomes passive, it no longer has an assigned channel.

An interesting aspect of this work is that the authors design DenseAP system keeping a key emphasis on practical deployability. The authors argue that because of the incredibly wide diversity of existing Wi-Fi devices, DenseAP must provide significant performance benefits without requiring any modifications to existing Wi-Fi clients. Furthermore, they do not consider any changes that require hardware modifications or changes to the 802.11 standard.

The work although very interesting is based on centralized architecture, so it has associated the disadvantage of this kind of architecture like one point of failure. Another aspect is that the solution is based on wired infrastructure between APs, so this mean that the APs belongs to the same administrative domain.

SMARTA [15, 13] is similar to DenseAP in that it uses a centralized server to increase the capacity of a dense AP deployment without requiring client modifications.

The SMARTA architecture is illustrated in Figure 2.2. The central controller coordinates the channels and power levels of the thin access points. The choices of channels and power levels are decided based on optimizing a utility

function, whose value is computed using measurements performed by the access points. The controller periodically cycles through five phases: startup, channel assignment, annotation, power-level assignment, and refinement.

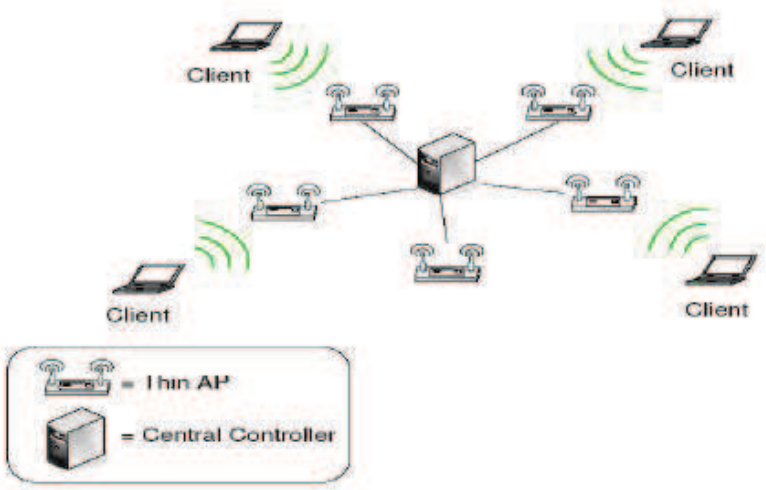

Figure 2.2: SMARTA System Architecture. Source [13].

The controller computes and periodically updates a Conflict Graph (or CG) which is a graph where nodes are APs and there is an edge between two APs, if they interfere when assigned the same channel. In the second or channel assignment phase, the optimizer makes use of the CG to generate optimal channel assignments for the access points. At the end of this step, every AP is assigned a good channel. In the third or annotation phase, the CG is augmented further to generate an annotated conflict graph, or ACG. The annotated conflict graph adds clients to the conflict graph, which previously only contained access points. During ACG construction, access point channels may be re-assigned to reflect client information in the channel assignment process. In the fourth or power-level assignment phase, SMARTA computes optimal power levels for access points. After this procedure completes, SMARTA moves to the fifth or refinement phase. In this phase, the power levels of access points are altered to account for small dynamic changes in the environment. This allows the system to evolve the configuration in response to changes in the environment.

Lets talk briefly about the channel assignment algorithm used in SMARTA. Channel assignment attempts to allocate orthogonal channels to nodes in the conflict graph that have an edge between them. The algorithm first assigns a random channel to each access point and computes the current total number of conflicts. Then, considering each access point (a_i) in turn it computes the gain in utility (in terms of reducing the total number of access-point conflicts) by switching that access point to a different channel. It computes the gain in utility for the access point on all channels and selects the channel C that yields the greatest gain for a_i . It then checks whether changing a_i to C yields an improvement in utility that is larger than the best utility gain seen in the

iteration so far. If so, (a_i, C) is labeled as the best improvement seen so far. Because the algorithm performs this operation across all access points, it selects the access point and channel change that yields the largest gain in overall utility. This process repeats until it reach a configuration where any further one-point alterations do not yield a gain in utility. Because the solution of the algorithm may depend on the initial assignment of channels to access points, they perform multiple runs of the algorithm and choose the best solution (in terms of utility) among them.

In the second phase of channel-assignment, they refine channel allocations as an optimization of the assignment they computed previously. For channel refinement, whenever they add a client to an AP (to construct the ACG), they try all other channels for that AP to see if they can reduce the total number of client conflicts, keeping the number of inter-AP conflicts constant. If such a channel is available (e.g. in the case of 802.11a), the access point is switched to that channel. If not, the access point remains on the same channel. This procedure is local to an access point as it does not require AP coordination to perform the channel search.

SMARTA uses a different approach to one of DenseAP.

The central controller builds a conflict graph among the APs, and uses this graph to tune the APs channel and transmit power. It does not manage client associations.

There are two main differences between SMARTA and DenseAP. First, DenseAP relies on correctly managing client associations. Second, DenseAP also shown that unilateral power control (without client cooperation) can hurt the performance of the system.

SMARTA is evaluated entirely in simulations impeding to do a fair comparison of SMARTA with DenseAP.

2.2.4 Distributed Algorithms

A natural question regarding distributed algorithms is if they are able to achieve the optimal solution for the channel assignment problem. Optimal solution means the minimal sum of all interference factors among interfering APs in the network [34].

Proposal of distributed algorithms to solve the problem of a wireless LAN selecting a channel to minimize interference with other WLANs can be classified in the following flavors:

- Single Self managed
- Policy Based
- Game Theory Based
- Constrain Based.
- Graph Coloring

An important implementation detail to be considered in each of these techniques if they are coordinated or uncoordinated. If the proposed algorithm require collaborations among entities. Commonly if the solution requires coordination among the participating APs usually means that the technique is

applicable to wireless network under the same administrative domain. Lets review each type of distributed algorithms in the following sections

2.2.5 Single Self-Managed Algorithms

The algorithm is a fairly simple one. The station periodically measures the channel quality (the interval between measurements need not be constant and can be selected to respect the cost of switching channel and the time needed to measure channel quality). When the channel quality of acceptable, keep using the same channel. Otherwise, randomly choose a channel with weighted probability based on past experience.

2.2.6 Policy-Based Distributed Algorithms

Hadjiantonis and Pavlou[27] present a policy based approach for dynamic channel configuration. They propose a policy-based solution based on the ad-hoc networking paradigm. A Distributed Policy Repository provides the support for policy-based device management and devices can autonomously adapt by examining local conditions and taking corrective actions in real-time.

The authors adopt the notion of "hybrid mobile ad hoc networks" by relaxing the main constraints of pure general-purpose MANET (a mobile ad-hoc network (MANET) is a self-configuring infrastructureless network of mobile devices connected by wireless), i.e. they consider the deployment of a network that consists of user devices with limited infrastructure support and connectivity.

Policy-based management (PBM) simplifies the complex management tasks of large scale systems, since high-level policies monitor the network and automatically enforce appropriate actions in the system. Policies are defined as Event-Condition-Action (ECA) clauses, where on event(s) E, if condition(s) C is true, then action(s) A is executed.

The components of a Policy-based management (PBM) system can be described as follow: The **Policy Repository (PR)** is an integral part of every policy-based system because it encapsulates the management logic to be enforced on all networked entities. It is the central point where policies are stored by managers using a **Policy Management Tool (PMT)** and can be subsequently retrieved either by **Policy Decision Points (PDP)** Once relevant policies have been retrieved by a PDP, they are interpreted and the PDP in turn provisions any decisions or actions to the controlled **Policy Enforcement Points (PEP)**.

PR is a centralized concept, various techniques exist to physically distribute its contents for resilience and load balancing. Typical implementations of a PR are based on Lightweight Directory Access Protocol (LDAP) Servers, also known as Directory Servers (DS).

The self-management framework proposed by the authors is as follows:

Management logic is encapsulated in policies that are transparently enforced to devices. Network Operators and Service Providers use the policy-based system to introduce the appropriate policies, aiming to set guidelines for the management of numerous user devices. Contrary to traditional management systems, the designed system does not require the mandatory enforcement of policies and tight control of managed devices. The system physically and logically distributes the policies among devices, making them available to vast

numbers of users that voluntarily choose to enforce the relevant policies that would eventually relieve them from manual configuration.

The DPR is designed, as an extension of the traditional PR. It is responsible for the distribution of policies in the network and for logically connecting the devices that collaboratively participate in the managed domain.

Distributed Policy Repository

They propose the distribution of management tasks among PDPs hosted on user devices and based on policy guidelines stored in DPR. The DPR is a set of distributed and/or replicated LDAP directories (replicas), configured to store policies. The design is based on the advanced replication and distribution features of modern LDAP servers.

Depending on user density and population, devices are organized in clusters and each Cluster Head hosts a PDP, facilitating the policy provisioning for its cluster's PEPs. The selected Cluster Heads collaboratively share management tasks as well as the hosting of the DPR. In order to decide where to place the DPR replicas, all Cluster Heads (PDPs) execute a special set of policies that combines a-priori knowledge of localized events (e.g. scheduled sport event) with dynamic real-time context information (e.g. processing load or free memory of each PDP).

For the authors the coordination of distributed PDPs in a wireless environment is quite hard. In the proposed solution, they transform this problem to the maintenance and deployment of the DPR by exploiting standardized LDAP operations and replication features. In this way, the DPR glues together the distributed PDPs and offers a logically uniform view of network management objectives through policies. Each DPR replica controls a configurable number of PDPs and each PDP is responsible to discover a replica for retrieving of policies and updates. The adopted pull-based approach relieves the DPR replicas from tracking PDPs and their operation is not affected from the intermittence of connections or the fluctuating number of PDPs.

Hybrid Ad Hoc Networks Self-Management

The authors select the case study of hybrid ad hoc networks for experimentation.

Inter-layer communication is used between MAC and Application layers, aiming to make the PBM system aware of the wireless channel conditions and provide a feedback mechanism for policies. Based on specified application events (e.g. reduced throughput), the triggered policies can initiate relevant procedures that with the inspection of MAC layer headers provide feedback to the system and possibly trigger further policies to correct the problem. A "closed control loop" management system is thus formed, adding a degree of autonomy. The authors observes two important advantages with the adoption of this approach. First, by using a policy-based design, the system is highly extensible and easily configurable. Policies can change dynamically and independently of the underlying technology. And second, by implementing decision logic at the Application layer, modularity is preserved without modifying the MAC protocol. Policies and extracted inter-layer parameters relieve the MAC layer from additional computations since inter-layer communication is only used when needed.

The authors then focus on two potential obstacles that need to be overcome in order to make the deployment of ad hoc networks easy, efficient and safe:

1. interference between newly created ad hoc networks and existing WLANs, and

2. regulatory conformance of ad hoc networks deployment.

and they attempt to propose solutions to the above problems based on the designed policies for a PBM system.

We will talk only on point number one, because number two is out of the scope of this work.

From the first point above: Interference between deployed ad hoc networks and existing infrastructure-based WLANs, as well as interference with already deployed ad hoc networks in the same area is the main reason for the disappointing performance of ad hoc networks and it can lead to severe problems in the throughput and coverage of collocated infrastructure-based WLANs. Choosing a random channel is likely to have a detrimental effect on the ad hoc network performance. To tackle this problem, they design policies that exploit MAC layer information for the initial configuration as well as the dynamic adaptation of the occupied wireless channel.

For the implementation of the Distributed Policy Repository they have used OpenLDAP Server [11]. They made the decision based in that it is an open source implementation of a very fast and reliable LDAP v3 Directory Server for Linux. In addition, the minimum specifications required for running this server allow an extensive range of devices, including low-spec laptops to efficiently host a directory replica. The DPR consists of one or more Master read-write directories and several read-only directory replicas (shadow copies). Master directories are hosted and controlled by the managing network entities, i.e. Network Operator and/or Service Providers.

After a global view of the work lets concentrate our attention to the policies defined to tackle the interference problem. There two important actions that are triggered named: *optimizeChannel* and *channel-switch*. Triggered actions *optimizeChannel* using as parameters the monitored measurements of a channel set. In the initialization event of the ad-hoc network a scanChannels() action is triggered. After finishing scanning (on scan complete event) three different actions can be triggered:

if the are PC (Preferred Channels) the optimizeChannel action is triggered with PC as parameters and algorithm

if the are FC (Free Channels) but the are no PC then the optimizeChannel action is triggered with FC as parameters and algorithm.

else the optimizeChannel is triggered with all as parameters and algorithm.

Note that in the three cases there is an algorithm parameter. They have implemented an algorithm based on the weighted average (WA) of a channel metric. For each candidate channel, the algorithm use for the calculation of the WA the channel metric and weights.

Similarly, when there is a new WLAN detected or when the Link Quality is reduced (because of an increased interference) and the LinkQuality fall below a

given threshold an action named generateStartAdapt is triggered to initiate the adaptation process for channel optimization. This action monitor an specified application metric. If it detects the reduction of goodput below some value, it acts by scanning all channels it then trigger the channel-switch method using the weighted average algorithm with specified weights.

Summary

The value of this work is in the proposal of use policy-based solution for the self management of wireless network for dynamic channel configuration. During last years a strong impulse have been done for the self management architecture on the network devices. Moreover, the RAN System that we will be using for our solutions design is a complete rule-based, distributed system specially designed and implemented to enable autonomic behavior on very constrained devices, such as domestic wireless routers with resources as low as 16 MB of RAM and 4 MB of storage memory.

The framework presented by the authors is based on autonomic elements concept, this mean the strategy is uncoordinated. The study is focused in hybrid ad hoc networks and it is based on LDAP technologies.

One of the drawback of this work is the empirical results. We believe that in the lack of simulations results more testbed must be considered. Although the one presented here shows promises results more scenarios must be tested. The testbed implemented for the demonstrations has only one scenario. It consists of 10 nodes: 2 laptops, 4 PDAs, and 4 Internet Tablets. The devices were organized in two independent clusters of five nodes each.

2.2.7 Game Theory Based Distributed Algorithms

Consider a dense urban residential area where each house/unit has its own wireless access point (AP), deployed without any coordination with other such units. It would be much better for individual APs that are in physical proximity to each other to form groups, where one member of the group would serve the terminals of all group members in addition to its own terminals, so that the other access points of the group can be silent or even turned off, thereby reducing interference and increasing overall Quality of Experience (QoE). These groups would include only members whose signal strength is sufficient to serve all group members.

Since there is no centralized entity that can control the APs and force them to form cooperative groups, the creation of such groups must be able to arise from a distributed process where each AP makes its own decisions independently and rationally for the benefit of itself and its terminals. Antoniou et al [17] propose a Game Theory based approach to model such decentralized schemes.

The authors model the idea of cooperative neighborhoods as a game and show that a group cooperative strategy in equilibrium, i.e. a strategy for units to voluntarily participate in a group where members serve terminals on a rotating basis, has the property that a unit participating in the group strategy is more likely to gain more in terms of QoE, than a unit defecting from such cooperation. They propose a protocol (the cooperative-neighbourhood game) with point of operation the game theoretic equilibria of a game. They further propose a

protocol for the participating units to operate in the associated equilibrium point of the game to achieve the reduced interference.

Game Theory provides appropriate models and tools to handle multiple, interacting entities attempting to make a decision, and seeking a solution state that maximizes each entitys utility, i.e. each entity's *quantified satisfaction*. The work is inspired by the *Prisoner's Dilemma/Iterated Prisoners Dilemma* game model.

For a good although succinct description of the game you can refer to the prisioner dilemma page on wikipedia [45].

The interactions in a cooperative neighbourhood can be modeled as a game between the participating units, where each member of the group has two choices at any given time: (a) to cooperate with its group members or (b) to defect from cooperation. Which of the two behaviors to select in each round depends on the strategy of behavior that a player has decided to follow during the repeated game. The strategy of each player, i.e. each unit, is selected such that it results in the highest possible payoff for the particular player. We refer to such interaction between any two neighbours as a cooperative-neighbourhood game.

In a one shot strategic game with two players if a player believes that his opponent will cooperate, then the best option is certainly to defect (according to the rules of the game). If a player believes that his opponent will defect, then by cooperating he takes the risk of receiving the least payoff, thus the best option is again to defect. Therefore, based on this reasoning, each player will defect because it is the best option no matter what the opponent chooses.

Therefore, based on this reasoning, each player will defect because it is the best option no matter what the opponent chooses. However, this is not the best possible outcome of the game for both. The best solution for both players would be to cooperate and receive the second best payoff.

What we have described is a one-shot Prisoners Dilemma, i.e. the players have to decide only once, no previous or future interaction of the two players affects this decision. Cooperation may evolve, however, from playing the game repeatedly, against the same opponent. This is referred to as Iterated Prisoners Dilemma, which is based on a repeated game model with an unknown or infinite number of repetitions. The Iterated Prisoners Dilemma is a quite popular repeated game model which demonstrates how cooperation can be motivated by repetition (in the case the number of periods is unknown), whereas in the one-shot Prisoners Dilemma as well as in the finite version of the Iterated Prisoners Dilemma, the two players are motivated to defect from cooperation.

Then the equilibrium point of the cooperative neighbourhood game, is that all members of a neighbourhood cooperate by assuming either the role of the leader and serving all terminals of the neighbourhood, or remaining silent to reduce interference while its terminals are being served by the leader.

Once the neighborhood is setup and session is stablished between members, each access point behaves in either of two ways, serves all terminals in a neighbourhood or remains silent. The rotation phase of the protocol occurs in a timely manner according to the member number of each participant, by a session modification initiated by the leader.

Summary

Game theory has been used in networking research as a theoretical decision-making framework, for example for routing and congestion control[24]. So it is valuable that this proposal tries to apply game theory to reducing interference in dense deployments of wireless network.

The use of game theory in wireless networks unfortunately comes with a set of challenges, the most important of which are the following ones: assumption of rationality, assumption to willingness to cooperate, choice of utility functions/payoff calculation and not guaranteed existence of the equilibrium. Another issue is that to succeed must be cooperative incentives. The basic idea for node punishment is that nodes should be rewarded or penalized based on their behavior. Nodes that offer resources should be aided. On the other hand, selfish nodes should be gradually isolated from the network.

2.2.8 Constraint Based

In artificial intelligence research area, a distributed constraint optimization problem (DCOP), consists of a set of variables that are distributed to a group of collaborative agents as valued constraints. The goal is to optimize a global objective function, maximizing the weight of satisfied constraints. Consider a set of N agents, $A = \{a_1, a_2, ..., a_N\}$, and a set of N values, $D = \{d_1, d_2, ..., d_N\}$, where each value d_j is assigned to an agent a_j and belongs to a finite discrete domain D_j . For each pair of agents (a_i, a_j) a cost function $f_{ij}(x, y) : D_i \times D_j \rightarrow R$ is defined. Agents have to coordinate themselves in order to find a set of values that optimize a global function, establishing the costs for constraints. Therefore, the goal of a DCOP algorithm is to find the optimal set, denoted D^* , whose values minimize a global cost function $g^* = g(D^*)$.

Monteiro et al [35] formalize the channel allocation as a distributed constraint optimization problem and propose a new cooperative channel allocation strategy. A solution based on the distributed pseudotree-optimization procedure (DPOP) is employed. The adjacent channel interference is analytically modeled for DPOP. In this work the APs can coordinate themselves using a dedicated wireless control interface.

They propose in this paper a new cooperative channel allocation strategy using the distributed pseudotree-optimization procedure (DPOP) described by Pectus and Faltings in [4]. The goal is to reduce the number of control messages exchanged between APs. The new algorithm is denoted **DOCA (Distributed Optimal Channel Assignment)**.

The DPOP algorithm employs an utility propagation method, based on dynamic programming inspired by the sum- product algorithm. It considers a pseudotree arrangement of the constraint graph G, which is composed of tree edges, edges that are part of a spanning tree, and back edges that are not part of the spanning tree. It is important to point out that the pseudotree can be obtained by executing the depth-first search (DFS) traversal algorithm in the constraint graph G, starting from one of the nodes $a_j \in A$. A path in the graph that is entirely made of tree edges is denoted as a tree-path. A tree-path associated with a back-edge is the tree path that connects two nodes involved in a back-edge.

With the following definitions: $A = \{a_1, a_2, ..., a_N\}$: set of agents (APs).

$D = \{d_1, d_2, ..., d_N\}$: finite domain for agents variables (available channel set). P_j : parent of node a_j , the single node above in the hierarchy of the pseudotree that is directly connected to node a_j through a tree edge. C_j : the children of node a_j , the set of nodes below in the pseudotree that are directly connected to the node a_j through tree edges. PP_j : the pseudo-parents of node a_j , the set of nodes above in the pseudotree that are directly connected to node a_j through back-edges. PC_j : the pseudo-children of node a_j , the set of nodes below in the hierarchy of the pseudotree that are directly connected to node a_j through back-edges. The algorithm is divided in three phases. First the agents organize in a pseudotree structure to be used in the next two phases, denoted UTIL and VALUE propagations, respectively.

In the UTIL propagation phase, the UTIL messages are propagated up the tree. The leaf agents send UTIL messages to their parents. A child a_k of a node a_j will send a vector of the optimal utilities, containing the cost of its best solution for every possible combination of variable assignments that can be achieved by the subtree rooted at node ak plus the relation between ak and its neighbor a_j .

After a node a_j receives all its children messages, it can compute the optimal values that can be achieved by the entire subtree rooted at a_j . Since all of a_j s subtrees are disjoint, by summing then up, it is possible to compute how much each of its children C_j values gives for the whole subtree rooted at itself. Thus a_j can send to P_j its UTIL message. Then node a_j can store its optimal values corresponding to the values received from its children C_j . Each node a_i relays its messages according to the UTIL phase of the algorithm. In the VALUE propagation phase, messages are propagated down the tree. This phase starts after the root node receives all its children messages. It can compute the optimal overall solution, based on all the UTIL messages received from its children. The root node then chooses the values that will leads to the optimal solution and informs its children sending a VALUE message to all of them. After receiving the VALUE message from its parent P_j , each node is able to recognize the optimal value d_j for itself and pass its d_j to its children. At this moment, the algorithm is finished for node a_j .

Summary

The paper proposes a distributed algorithm to minimize the global interference of a network of wireless access points. The issue is of high interest for the management of high density wireless networks and the authors address it timely. Another strong point of the paper is that presents a good review of the state of the art. Overall, the authors demonstrate a good understanding of the problem under study.

One of the main weakness of the paper is the feeling that good part of the proposal complexity is being hidden or understated. For example, the authors are not clear about the size of the UTIL messages and they are indirectly revealed by the last graph.

More important is that the authors are comparing their solution with LO-A algorithm in terms of the number message exchanged and how both algorithm approximate to the optimal cost but they do not mention the actual communication overhead induced by LO-A algorithm (in bits) this way they can compare it against their proposed algorithm.

Building the underlying tree needed by the algorithm is not a trivial process in terms of communication overhead and complexity. This complexity, or even the fact that electing a leader is not always a resoluble problem, is understated or ignored by the authors in this text.

2.2.9 Graph Coloring Based

A channel assignment problem is typically modeled as a graph-coloring problem to improve the above basic strategy: there is a vertex on the graph corresponding to each AP, an edge on this graph represents potential interference, and the colors represent the number of non-overlapping channels. A goal of the channel assignment problem is to cover all APs (vertices) with the minimum number of channels (colors) such that no two adjacent APs (vertices) use the same channel (color). This is the minimum graph coloring problem.

Mishra et al[34] propose a variation on the above problem and define a weighted variant of the graph-coloring problem, In this weighted variant, each vertex corresponds to a distinct AP as in the case of traditional modeling. However, each edge on this graph now has a weight associated with it. The weight of an edge indicates the importance of using different colors (channels) for the corresponding vertices (APs) that are connected by that edge. Note that in this variant it is permissible to allocate overlapping channels to neighboring APs.

They present two techniques. The first technique does not require any collaboration among the APs and can be applied to a wireless network formed by APs belonging to different WLANs. The technique assumes that the APs are greedy in nature they try to minimize the interference within their area of coverage. The second technique tries to reduces the total number of clients suffering interference in the network as a whole. However, this technique requires collaboration among the APs. This is particularly suited for efficient channel assignment for a single wireless network, or for multiple wireless networks if the administrators are willing to cooperate.

The authors give a formal definition of the problem as follows:

They modeled the network with a *overlap* graph: $G = (V, E)$ where $V = \{ap_1, ap_2, ..., ap_N\}$ is the set of APs. There is an edge between APs ap_i and ap_j ($ap_i \neq ap_j$) if there is an overlap in the interference region of the BSS created by APs according to the interference model.

Let W be the *weight function* on G. $W(ap_i, ap_j)$ indicates the number of clients associated with the two corresponding APs that are affected if these APs are assigned the same channel.

A channel assignment $C(ap_i)$, $ap_i \in V$ is a mapping $C : V \rightarrow \{1...k\}$ from the set of vertices to the set of colors. The edge (ap_i, ap_j) can be classified as *conflict free* edge if the interference between these two channels is zero (e.g. channels 1 and 6 in 802.11b) or as *conflict edge* if the choice of colors have some positive interference (e.g. channels 1 and 2 in 802.11b).

They define a term *Interference-factor*, denoted by $I(ap_i, ap_j)$, for each edge, which is the interference between the colors assigned to the two APs.

Thus, given G and W as defined above, the channel assignment problem is to find a mapping C such that an objective function is optimized. Then in this work the authors define three objective functions L_{max}, L_{sum} and L_{num}

Minimize: $L_{max}(G,C) = \max_{\forall e=(ap_i,ap_j)\in E} I(ap_i,ap_j)W(ap_i,ap_j)$

This mean, minimize the maximum impact of interference among all overlap regions between APs.

Minimize: $L_{sum}(G,C) = \sum_{\forall e=(ap_i,ap_j)\in E} I(ap_i,ap_j)W(ap_i,ap_j)$

This mean, minimize the total effect of all interference experienced by clients as a consequence of channel assignments.

Minimize: $L_{num}(G,C) = \sum_{\forall e=(ap_i,ap_j)\in E} I(ap_i,ap_j)$

this mean, minimize the total impact of conflict edges.

The authors focus on the L_{max} objective function and they present two different strategies which they call $Hminmax$ and $Hsum$ to minimize this objective function. As mentioned before $Hminmax$, does not require communication between the APs, and applies to multiple coexisting wireless networks sharing a limited RF spectrum. It attempts to reduce the L_{max} objective function. The second algorithm, Hsum, tries to reduce the L_{max} objective function (like the $Hminmax$) algorithm, but additionally is also able to reduce the L_{sum} objective function to a significantly lower value than what $Hminmax$ is able to achieve, without compromising the L_{max} objective function. To do this, the Hsum algorithm requires some cooperation between APs.

As we do not present algorithms details here, the reader can consult the referenced paper for further information.

Summary

Two channel assignment algorithms for WLANs were proposed, modeling the problem as a weighted vertex coloring problem. The authors did a good formalism of the problem presenting three objectives function, but they focus in one of them only. As we will see in the next chapter our focus will be in another function, more precisely in the L_{num} objective function which we will be extending with some other restriction.

In $Hminmax$ algorithm, they tries to minimize L_{max} objective function using a locally and uncoordinated approach. It does not require any collaboration among access points and can be applied to wireless networks where APs belong to different administrative domains.

In $Hsum$, aims to minimize the overall interference experienced by clients, but requires coordination among the participating APs. This technique is applicable to wireless networks under the same administrative domain, where APs coordinate themselves using a wired network.

2.3 Conclusion

As we can see from previous sections there are several studies for channel assignment problem in dense wireless networks. But, far from solved the subject under study continues to be a challenge. For example a newspaper article [6] in july 2, 2012 commented Hollywood's plan to be on the front lines of technology was a failure. The article continue reporting that "The aim was to install transmitters

throughout the city that would enable digital water-meter readings, credit-card parking-meter payments and a secure police and fire network. The cherry-on-top fringe benefit: a free citywide wireless network. The Wi-Fi tanked for a lack of places to set up transmitters without signal interference."Those interferences produced by Access Points installed in big buildings, towers and houses. And to the point of view of residents the article reports that "What frustrates residents are the hundreds of access points or little antennas affixed to the tops of poles and light posts across the city that continue to broadcast *Wireless Hollywood* yet are unable to connect to the Internet."

This thesis reviews the state of the art in channel assignment for wireless networks, describing many algorithms individually and conforming a very complete description of available strategies, being a good starting point for the channel assignment for wireless networks study.

We can summarize related work as:

- Planning-based approaches (industry's approach).
- Genetic Algorithms. To run once and assign channels as results of algorithm run.
- Client Assisted. Needs to install some piece of software in client devices.
- Centralized. One node with a global network view making decisions.
- Distributed. Distributed algorithms can be subclassified as:
  - Policy-Based
  - Game Theory Based
  - Constraint Based
  - Graph Coloring Based

In the following chapters we will be using ideas from works described in previous sections. We will present three different solutions to the problem of channel assignment that use RAN[32] as their control communication bus. RAN is a rule based system similar to the one on section 2.2.6. In the formalization of the problem we will be extending L_{num} objective function presented in section 2.2.9 with more constraints like that there are neighboring access points that are not running our software but that produce interference.

We will implement a totally uncoordinated approach like the one presented in 2.2.6, a coordinated approach like the one presented in 2.2.7 and finally we will implement a centralized approach like the one presented in 2.2.3.

The centralized algorithm must produce the best result based on their knowledge of the whole network. We decided to implement it mainly to compare the other two approaches. The only similarity with the strategy in 2.2.3 is that both rely on a central node. DenseAP algorithm only assign channel to an AP, when a new client will be assigned to them and the AP does not has clients yet. In our case, the central node calculates channel assignments at the beggining for each participating node.

We decided to implement the uncoordinated approach for those cases where connection between nodes can be a mess. We implemented this strategy using RAN, that is a rule based system, therefore we designed a set of policies to

control de autonomic behavior. In this regard this work is similar to the one presented in 2.2.6, with the exception they are constrained to hybrid ad hoc networks.

Finally, the coordinated approach is a natural way of doing distributed computation; that is, on how to solve problems and perform tasks efficiently in a distributed computing environment. This environment consists of a finite collection of computational entities communicating by means of messages in order to achieve a common goal; for example, to perform a given task, or to compute the solution to a problem. Although each entity is capable of performing computations, it is the collection of all these entities that together will solve the problem or ensure that the task is performed. In this kind of computation, to solve a problem, we must discover and design a distributed algorithm or protocol for those entities: A set of rules that specify what each entity has to do. The collective but autonomous execution of those rules, possibly without any supervision or synchronization, must enable the entities to perform the desired task to solve the problem. In 2.2.7, the authors present a coordinated approach although it is based on game theory. We present a greedy implementation of a coordinated approach to apply the concepts of a distributed computation.

Chapter 3

Proposed Strategies

In this chapter we will formalize the channel allocation problem which is defined as a minimization problem where we have an objective function named L_{ran} that we want to minimize. L_{ran} is similar to L_{num} presented by Mishra2.2.9 L_{ran} is defined with the goal of minimize the total interference level seen by the APs. To calculate L_{ran} we defined a term *Interference-level*, denoted by $I_l(ap_i, ap_j)$, which is the interference level (or measured interference) that node ap_i senses with respect to ap_j.

Once the problem is formalized, we will present the strategies that have been implemented to solve it. In this respect, we implement three different proposals, with the intent to evaluate the practical deployability and scalability of the strategies. In a later chapter we will present a testbed for the three approaches to evaluate the real behavior of the protocols, and also present some simulations mainly for scalability evaluation.

The proposals can be classified as:

- Local and uncoordinated.
- Distributed and coordinated.
- Centralized.

These implementations match many of the basic concepts that we reviewed in the previous chapter.

For the local and uncoordinated approach we will follow a strategy that bibliography refers as *Least Congested Channel* (LCC) [26]. The conceptual steps described in bibliography are:

1. Each AP periodically checks other data transmissions in the channel that it is using.
2. If the volume of traffic in that channel (generated by other APs or clients of other APs) is greater than a threshold, then the first AP tries to move over to a less congested channel.

In our implementation the strategy is implemented as follows:

1. One node will observe the channels used by neighboring nodes and measure the signal level quality for each of them.

2. If the level of interference in the node is greater than a threshold, then the node tries to move over to a less congested channel.

For the distributed and coordinated approach we will determine a leader that will coordinate the execution of the distributed protocol. Although leader election can be implemented using a distributed algorithm (we will review this topic in more detail later, in section 3.4.1), in our implementation, the leader is determined statically during system set up. Once the leader is determined, then this particular node will start the process. It will coordinate a two step distributed algorithm:

1. First, a ranking process is fired. When the ranking process completes each node is ranked according to the level of interference (the one that has more interference is first).

2. Second, in the order determined in the previous step, each node is requested to assign itself a channel knowing it's neighbors assignment.

This is a greedy implementation of the algorithm.

Finally, in the centralized implementation, one node acquires information of all the others in the system and their level of interference. Then, that node will use a backtracking algorithm to obtain the optimal solution in a reasonable time. After the algorithm has been completed, then this central node informs other nodes in which channel they must operate. This approach is similar to those presented in section 2.2.3.

To implement all these strategies we use RAN [19] as the core infrastructure. We will review RAN in section 3.2.1. Our goal is to have the same implementation of our strategies to run both, in the testbed and in the simulator, without changes. To be able to accomplish this goal we stablished the necessity of a set of utilities or operations. These utilities are needed to acquire information about the level of interference at each node and the channels being used. We will provide a high level description of these utilities in section 3.2.2.

Lastly, we will present the design of the three strategies used to solve the channel allocation problem showing in detail the algorithms implementation.

3.1 Formal Definition of the Problem

In this section we will formalize the problem and then, in later sections we will present different strategies to solve it. We will base this theoretic formulation on what Mishra et al[34] present as L_{num} objective function, but we extend that notion with some other constraints.

Let k denote the total number of non-overlapping channels available. The **first restriction** of the problem is that we will be working with the 802.11 b/g protocol, so in this case $k = 3$.

Next, we will define a graph $G = (V, E)$ where $V = \{ap_1, ap_2, ..., ap_N\}$ is the set of APs that form the network.

From those N APs there is a subset that are under our control and there is another subset that are in the environment interfering with our nodes. Notice that, because we cannot manage these neighbouring APs, they are not running our algorithms.

The **second restriction** can be stated as follows: We have control over a subset V' of V, so we can manage only V', in other words we change channel assignment of nodes belonging to V'. The set of nodes in $V - V'$ are in the neighborhood, but interferes with the set V' and they already have a channel assigned that cannot be changed.

If ap_j can interfere ap_i then an edge between APs ap_i and ap_j ($ap_i \neq ap_j$) is placed. For this to happen both APs must be in channels that overlap (see Figure 1.2), for example if ap_i is in channel 1 and ap_j is in channel 3, and the signal of node ap_j is listened by node ap_i.

In most cases, if ap_j interferes with ap_i the opposite is also true, ie ap_i interferes with ap_j because both APs are operating in overlapping channels, and is highly probable that if ap_i can listen ap_j then ap_j can listen ap_i. But this last statement is not true in all cases, for example if the transmission power is not the same in both nodes and they are far enough so that one node can listen to the other but the latter cannot listen the former.

The **third restriction** is that the graph is directed.

As Mishra et al stated for their weighted graph coloring problem, we will say that the problem of *Channel Assignment* $C(ap_i)$ where $ap_i \in V'$ is a mapping $C : V' \rightarrow \{1...k\}$ from the subset of vertices we can manage to the set of available channels.

We define the *Interference-level*, denoted by $I_l(ap_i, ap_j)$, for each edge, which is the interference level (or the measured signal-to-noise ratio that is presented as the link quality) that node ap_i senses with respect to ap_j. In chapter 5 when we present our testbed and the simulation we will explain how $I_l(ap_i, ap_j)$ is measured.

Finally, we present our objective function:

$$Minimize$$
$$L_{ran}(G, C) = \sum_{\forall e=(ap_i,ap_j)\in E \wedge ap_i \in V'} I_l(ap_i, ap_j)$$

Note that this objective function L_{ran} (that we named in honor of our RAN system) is very similar to L_{num} presented by Mishra. The differences are that we use *Interference-level* instead of the *Interference-factor* and the restriction that one of the nodes belongs to a V' and the other node can be an unmanaged neighbor (may be part of another network).

L_{ran} as defined above minimizes the total interference level seen by the APs.

There is a **fourth restriction** in our problem not mentioned above but that has practical implications. The amount of nodes that we are considering in this work are in the order of tens. We will not work in an environment with hundreds or thousands of APs. This is because the focus of this work are places such as schools, lecture halls, hotel ballrooms, work offices and convention centers that are reduced in size.

3.2 Solution Design

Two of the proposed strategies need to exchange messages between participating nodes. So the first constraint that we need to overcome is the design of a communication infrastructure. For this purpose we use RAN[19][32]. RAN,

as we will see in the next section, has a communication bus named rnr, that connect all participating nodes. Because we developed the RAN System we have a good knowledge of what kind of message are interpreted by the RAN System and how those message must be formatted.

We want to minimize our objective function L_{ran}. For this to happen we need a way to measure interference between nodes and we need to detect the channels being used by each node, independently of the operating host. In this regard we abstracted a set of utilities that must be available in both, in the testbed and in the simulator. While in the testbed we make use of existing Linux operating system commands, in the simulation we build four programs for the Windows platform that mimic the behavior of the Linux operating system commands.

As was mentioned in the introductory section of this chapter we are proposing three solutions for the channel allocation problem. We will first present a local and uncoordinated approach. This approach will work locally, where each node will take decisions based on his knowledge of the environment without any coordination with the rest of the nodes in the network.

Also, we will present a distributed and coordinated approach that is based on a greedy strategy. A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In this case we propose to sort nodes according to a rank value. To do this we need a leader and a coordination between nodes. Then each node, starting in the best ranked node, makes the locally optimal choice of the channel that it must operate in.

Finally, we will present a centralized solution where one node will have the view of the entire network and then it will be responsible to calculate the best channel assignment for each node.

The rest of the chapter is as follows. In the next section we will present a brief description of the RAN System and we will present the set of commands mentioned above. Then we will present the implementation design of the local and uncoordinated solution in detail. Later, we will present the coordinated and distributed solution where we will see that we need to have a leader, and we will show how the rest of nodes coordinates with the leader for a channel assignment. Finally, we will present the centralized solution and we will see how a backtracking algorithm can determine the channel assignment for each node in the network.

3.2.1 Ran System Architecture

We here, present a very succinct description of the RAN system. For more information please refer to [19] or [32].

The system comprises a set of services and a communication bus (RNR). The services are deployed in the participating nodes, and use the rnr bus to communicate between them, be it on the same node or on the ones deployed elsewhere (but belonging to our set of nodes V'). The services are a policy decision and enforcement points as defined in PBNM (collectively known as LUPA), and a monitoring service (RMOON). The architecture can be seen in Figure 3.1

The monitoring service (rmoon) will use the *GetXxx* utilities presented in the next section, to inform the policy decision point (PDP) about the environment.

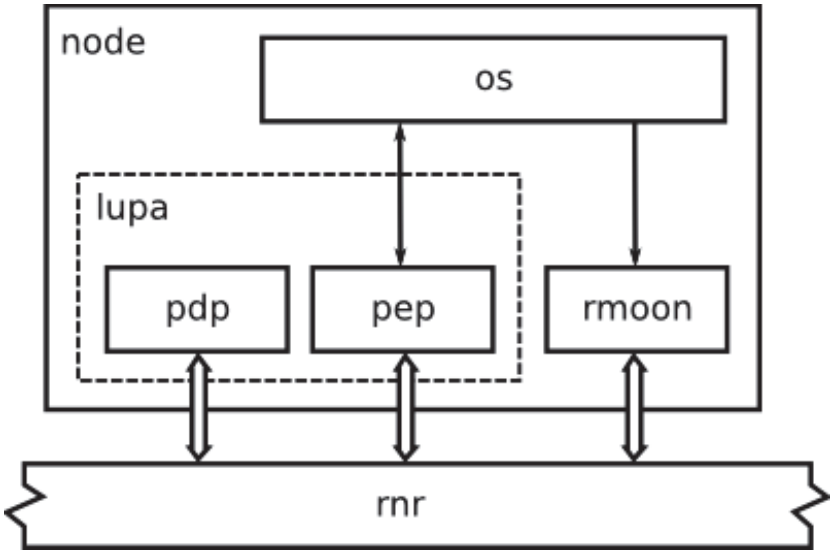

Figure 3.1: Ran System Architecture. Source [32].

For example rmoon will use *GetChannel* to let pdp know in which channel the AP is operating. The policy enforcement point (PEP) is in charge to do actions instructed by the pdp component. To do this, pep will use the *SwitchChannel* command.

At the heart of the Ran System is the policy decision point. Based on data passed from the rmoon via the rnr communication bus pdp can instruct pep to switch channel via the rnr communication or pdp can determine to stay in the same channel.

pdp is feeded with a finite state machine (fsm) to implement the behavior of each deployment. For each of our proposed strategies we need to provide pdp with a FSM. In the following sections we will be explaining what is inside these fsms.

To compile the rules into the finite state machine understood by our pdp implementation we use a special kind of finite state machine, presented in [18], called a Finite State Transducer extended with Tautness Functions and Identities (TFFST).

Then we have one fsm for each node of our local and un-coordinated approach, we will have two kinds of fsm for the distributed and coordinated approach, one for the leader and one for the rest of nodes, and for the centralized approach in the nodes we will have only rmoon and pep components without pdp and the central node will have a program to calculate the best channel assignment for each node.

To finish with this brief presentation of the Ran System we must say that all the RnR communications are trough TCP sockets, using a simple protocol with four control messages. There is an instance of the RnR router running on every node that is part of the system, this makes the system homogeneous from the communications point of view.

The *RnR* protocol has four messages:

Subscription when an entity of the system, such as a PDP or an *RnR* router, wants to subscribe to some kind of notifications it has to send a subscription message like the one depicted bellow.

```
SUBSCRIBE
subscriptor_id=srid
subscription_id=snid123
FILTER
anatribute = this text
anotheratribute < 20
END
```

The message has a header with basic information about the message itself. That is: the identifier of the subscriptor needed later to forward the notifications, and the identifier of the subscription, needed for processes such as unsubscriptions.

Notification a notification is a message that either (i) informs about the occurrence of an event at some entity of the system, or (ii) carries a request for the enforcement of an action. Any client (publisher) can issue them or subscribe to receive those notifications of its interest. A sample notification could be:

```
NOTIFICATION
source=srid
notification_id = notid234
timestamp= 1234
anatribute = this text
anotheratribute = 5.0
END
```

Control Messages for completeness we will mention that there are two messages with a supporting role: `HELLO` and `HELLO_REPLY`. They are used during the connection phase to interchange identities between clients and routers and between routers.

3.2.2 Utilities

As we mention before to implement our testbed and our simulation we identified a set of operations to be available at each node. These operations will be invoked from the RAN system to obtain information or to do an action. The following is the list of operations:

1. *GetChannel*. Each node must know in which channel he is operating. For example, in 802.11bg this can be an integer between 1 and 11.

2. *GetCellsInRange*. Each node must be able to find who are the interfering nodes. This mean it must be able to detect neighbors node and on which channel those nodes are operating, and the level or quality of the signal perceived for each of those nodes.

3. *GetAddress*. This is intended for node identification. In testbed this can be the mac address and in the simulation can be a node name.

4. *SwitchChannel*. Each node can be requested to switch the channel he is actually using.

In appendix A there are the implementation details of each operation for the testbed environment and for the simulation environment.

3.3 Local Un-Coordinated Solution for Channel Assignment

Commercial access points have an option to automatically configure the channel where the access point will operate, named Auto. This auto channel scan option lets the access point automatically find the channel with the least interference and uses that channel for communication. If you disable this option, the access point uses the channel that you specify manually. In this section we present an un-coordinated solution where each node in isolated mode keeps monitoring the environment for a less congested channel and swithching to that channel if some conditions, that we will be explaining below, are met.

In this solution each RAN system deployed in a node work locally, this mean that the Rnr communication bus is not connected to the Rnr of other nodes. The Rnr in this implementation is used only to communicate pdp, pep and rmoon components of the same node.

The implementation is as follow:

Step 1: initialize At startup pdp instruct rmoon to continuously inform two values: the channel least used and the interference level in the channel the node is using. rmoon accomplish this using the commands presented before. rmoon invokes *GetCellsInRange* to get the list of all nodes that are in the area of interference, the channels they are using and the perceived quality of the signal. With this information it can determine the channel least used. Also with this information plus the command *GetChannel* that return the channel currently used by the node, rmoon can determine the level of interference.

The messages send from the pdp to the rmoon are the following:

```
NOTIFICATION
notification_id=wt1123637438
service=/lupa/pdp
host=127.0.0.1
target_host=127.0.0.1
target_service=/lupa/rmoon
watcher_id=clu
mib=ChannelLeastUsed
message_type=action
command=watch_mib
op=>
value=0
END
```

```
NOTIFICATION
notification_id=wt2617761189
service=/lupa/pdp
host=127.0.0.1
target_host=127.0.0.1
target_service=/lupa/rmoon
watcher_id=noaumc
mib=NumberOfAPsUsingMyChannel
message_type=action
command=watch_mib
op=>
value=0
END
```

The meaning of the fields:

- *notification_id* is the identifier of the message
- *service* and *host* indicate which component is sending the message, in this case pdp on the local host.
- *target_host* and *target_service* indicates the destination of the message, in this case rmoon on the local host.
- *message_type* and command tells the destination (rmoon) to do a *watch_mib* action
- *watcher_id* is an identifier of the watcher.
- mib is the value that must be monitored by the *watch_mib* action. More on this below.
- op and value indicates that rmoon must notify when the values monitored satisfy the condition, in this case always that the value is greater than zero.

The two values monitored by the rmoon service in this solution are: *ChannelLeastUsed* and *NumberOfAPsUsingMyChannel*. *ChannelLeastUsed* is the channel number with less interference, ie is a value belonging to the set: $\{1, 6, 11\}$. *NumberOfAPsUsingMyChannel* is the level of interference in the channel the node is using. Is a real value in the range 0..1. This value is calculated from all nodes in the same channel as this node and the perceived quality of the signal.

Step 2: rmoon notifying pdp After the initialization step rmoon as it was instructed continuously monitor for the conditions and notify pdp with the following notification messages:

```
NOTIFICATION
notification_id=trap_997422823
service=/lupa/rmoon
host=127.0.0.1
target_host=127.0.0.1
target_service=/lupa/pdp
watcher_id=noaumc
mib=NumberOfAPsUsingMyChannel
message_type=trap
value=0.78571428571429
END
```

```
NOTIFICATION
notification_id=trap_670617586
service=/lupa/rmoon
host=127.0.0.1
target_host=127.0.0.1
target_service=/lupa/pdp
watcher_id=clu
mib=ChannelLeastUsed
message_type=trap
value=6
END
```

In both cases you can observe that the origin is rmoon and the target is pdp, and that the message type is a trap message. In the *mib* attribute is specified which value is being reported: *NumberOfAPsUsingMyChannel* or *ChannelLeastUsed*. In the first case as we mention is a real value between 0 and 1 and in the second case is the least congested channel.

Step 3: pdp evaluate conditions After receiving trap notification messages from rmoon, pdp evaluates if the AP needs to perform a channel switch or not. Lets suppose the AP is operating in channel C then two conditions must be met for the pdp take the decision to switch channel. The conditions are:

- C is different from channel (value attribute) informed in the trap message for mib *ChannelLeastUsed*. This mean that exists a least congested channel other than C.
- the value informed in the trap message for mib *NumberOfAPsUsingMyChannel* is greater than a $Thresohold$ value.

At first $Thresohold$ is a configuration value but then we realize that the system present oscillatory issues, so for this reason we applied a feedback control technique to set this value at runtime based on desired value of a another measured value. This will be presented in next chapter.

Step 4: if conditions are met then pdp instruct pep to switch channel If conditions presented in the last paragraph are both true then pdp request pep to switch channel with the following notification message.

```
NOTIFICATION
notification_id=randomrandom4
service=/lupa/pdp
host=127.0.0.1
target_host=127.0.0.1
target_service=/lupa/pep
message_type=action
command=switchChannel
message=6
END
```

In this message the source is pdp and target is pep. The message type is action. When the message type is of type action two other attributes must be specified: *command* and *message*. In this solution when this kind of notification

is emitted the *command* attribute value always is *switchChannel*. In other ways when pdp instruct pep to do an action, this action always be a *switchChannel*. The message attribute value indicates to which channel pep must change. In the example above the AP will start operating in channel 6.

To actually implement the action pep will use the *SwitchChannel* command presented in the RAN System.

This complete the implementation details of the local and un-coordinated RAN system for channel assignment in dense wireless networks.

3.4 Distributed Coordinated Solution for Channel Assignment

The distributed coordinated solution considered here, as well as the centralized solution presented in the next section, require, for their solution, the *Connectivity* (CN) restriction (i.e., every node must be reachable from every other node). We will also assume *Total Reliability* (TR) and *Bidirectional Links* (BL).

The connectivity restriction is implemented connecting all participating rnr services. Every notification message is transmitted to all participating rnr, and this way it reach every component of the system. Remember when we presented the RAN System that when an entity of the system, such as a PDP, wants to subscribe to some kind of notifications it has to send a subscription message like the one depicted bellow.

```
SUBSCRIBE
service=rnr
subscription_id=pdp_sub_447657310
FILTER
target_host=host8
target_service=/lupa/pdp
END
```

Observe the Filter criteria, *target_host* is the node identification (usually the ip address) that is running the component and *target_service* is the component identifier in that node (usually it will be */lupa/pdp*, */lupa/rmoon* or */lupa/pep*). So, when a notification message match the filter criteria the message will be deliver to the service that registered the subscription.

In a distributed environment, most applications often require a single entity to act temporarily as a central controller to coordinate the execution of a particular task by the entities. In some cases, the need for a single coordinator arises from the desire to simplify the design of the solution protocol for a rather complex problem; in other cases, the presence of a single coordinator is required by the nature of the problem itself. In this particular strategy we will have a node named leader that will act as a coordinator to control the algorithm behavior.

3.4.1 Leader Election

From chapter three of the book [41] the problem of choosing such a coordinator from a population of autonomous symmetric entities is known as Leader Election (Elect). Formally, the task consists in moving the system from an initial configuration where all entities are in the same state (usually called available)

into a final configuration where all entities are in the same state (traditionally called follower), except one, which is in a different state (traditionally called leader). There is no restriction on the number of entities that can start the computation, nor on which entity should become leader.

As is demonstrated in the book, under our set of restrictions: Bidirectional Links, Connectivity, and Total Reliability, there is unfortunately a very strong *impossibility* result about election.

We already said that pdp is feeded with a finite state machine (fsm) to implement the behavior of each deployment. In this strategy we will have a fsm for the leader node and another fsm for all other nodes. We will install manually the leader fsm in one particular node, so the leader will be set by hand. Changing the RAN System to select a leader with some of the mechanism described in the book will be scheduled for future works.

3.4.2 Ranking Mechanism

The algorithm for the distributed and coordinated solution is based on an algorithm for the *ranking problem*. The *ranking problem* can be stated as follows: Considering a graph where each entity x has an initial value $v(x)$; these values are not necessarily distinct. The rank of an entity x will be the rank of its value; that is, $rank(x) = 1 + |\{y \in V : v(y) < v(x)\}|$. So, whoever has the smallest value, it has rank 1. See figure 3.2

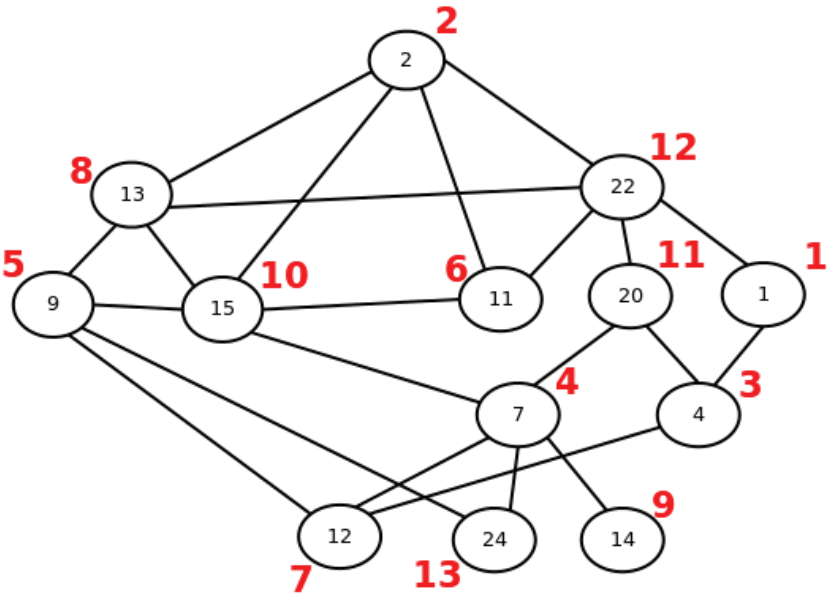

Figure 3.2: Ranking in a Network. Source [1].

In our solution the value $v(x)$ that each entity have is the interference level at that node (including the leader). The node with less interference will have value 1 and the node with greater interference will have value $|Nodes|$.

To solve the ranking problem there are two approaches: centralized or decentralized. We will use a centralized approach where each node will send the value that it has to the leader. Then, the leader compute the ranking by simply comparing the values.

Once the ranking has been stablished the leader instructs in turn each node to assign a channel. First, the node that is ranked greater (the one that has more interference). The node based on their local information assign a channel itself. Once the node has assigned a channel the node inform the leader that is has completed the work. The leader then instruct the second node (according to the ranking) to assign a channel. The process continues until all nodes, including the leader have a channel assigned.

3.4.3 Leader Implementation

Step 0: leader store its own interference value At startup the pdp component of the leader node instruct its own rmoon to measure its interference level (this is done in all nodes). Then it store this value because the leader node will be ranked too.

Step 1: leader request all other nodes to send it values The first step of the protocol itself begin with the pdp component of the leader node broadcasting a message to all participating pdp to send its value, so the leader can rank the entire network.

The message send from the pdp to all others pdp is the following:

```
NOTIFICATION
notification_id=randomrandom4
service=/lupa/pdp
host=host_lider
target_service=/lupa/pdp
mib=rank
message_type=trap
value=1
END
```

Two things must be observed. First, the identification of the sender is important because all pdp will respond to the *service* and *host*, in the example above the */lupa/pdp* service in the *host_lider* node. Second, the destination is the *target_service*. Note that there are no *target_host* attribute. This is because RAN System deliver messages to the susbcriptor if the filter criteria evaluates true. pdp services subscribe with two conditions: *target_service* and *target_host*. The notification only comes with *target_service*. The condition to deliver a message is that if the attribute is present it must be true. This is the way we can send a broadcast message to all pdp services.

Step 2: leader ranks node and notify the first one to assign a channel As we will see below each */lupa/pdp* in the network will respond to the broadcast message with a *ranking* message in which one of its attribute is the value that its the level of interference measured by the node in which */lupa/pdp* is running.

Leader node wait until all nodes respond. To do this there is a configuration value in the leader node named $totalNodos$ with the number of participating

nodes. To count the responding nodes the */lupa/pdp* service in the *host_lider* node take the value of the *host* attribute in the notification messages received. It store in a table like structure the node identification and the corresponding node value reported. This will fire an *all ranked* event.

When all nodes have responded then */lupa/pdp* simply sort the table structure according to the value field. After sorted the table then it picks the first entry in the table an send a notification message to the corresponding node requesting it to assign a channel. The notification message send is as follow:

```
NOTIFICATION
notification_id=ntf1123637438
service=/lupa/pdp
host=host_lider
target_service=/lupa/pdp
target_host=host1
mib=assignChannel
message_type=trap
value=0.5464
END
```

Note that the *target_service* is a peer */lupa/pdp*.

One thing to consider is that if the first node ranked is the *host_lider* itself it first instruct its own */lupa/pep* service to assign channel and then it picks the second node in the table structure and send the above notification message.

The notification message to its own *pep* is as follow

```
NOTIFICATION
notification_id=ntf1123637438
service=/lupa/pdp
host=host_lider
target_host=host_lider
target_service=/lupa/pep
message_type=action
command=switchChannel
message=0.847
END
```

Step 3: leader iterates until all nodes has been instructed to assign channel Every participating node respond to the leader with a *okReceived* message, this produce a *okReceived* event that tell to */lupa/pdp* service to move to the next entry in the ranked table.

Then it send to this node the same message of step 2. The protocol continue until there are no more entries in the table and the last *okReceived* is received.

3.4.4 Node Implementation

Step 0: each node store its own interference value At startup the pdp component of each node instruct its own rmoon to measure its interference level. When we explain our testbed and our simulation implementations in later chapter we will explain how this measure is done. Then each pdp store this value to send it back to the *host_lider* when is instructed to do so.

Step 1: response to the broadcast message The broadcast message from the leader node, the message labeled with $mib = rank$, produce a *rank* event in pdp nodes. Then the pdp service will respond with the value stored in step 0.

The message send from the pdp in the nodes to the pdp of the leader is the following:

```
NOTIFICATION
notification_id=notif_670617586
service=/lupa/pdp
host=host1
target_host=host_lider
target_service=/lupa/pdp
mib=ranking
message_type=trap
value=0.785468
END
```

This is the notification message that *host_lider* is waiting from all participating nodes as we described in step 2 of leader implementation. Then the nodes start waiting for the message request to assign a channel.

Step 2: pdp assign channel and send OK back to the pdp of the leader Upon arrival of the notification that instruct the node that its your turn to assign channel then an assign channel event is fired. This cause the pdp send a message to the pep service as follow:

```
NOTIFICATION
notification_id=ran4123637438
service=/lupa/pdp
host=host1
target_host=host1
target_service=/lupa/pep
message_type=action
command=switchChannel
END
```

To assign a channel to the AP, pep first scan which channel has the less interference level and then using the *switchChannel* command change the frequency of the AP.

Lastly, pdp respond to the pdp in the leader that it completes the process sending a response message as follow:

```
NOTIFICATION
notification_id=ran1123637438
service=/lupa/pdp
host=host1
target_service=/lupa/pdp
target_host=host_lider
command=okReceived
message_type=trap
message=1
END
```

3.4.5 Protocol Sequence Representation

In Figure 3.3 you can see a visual representation of the protocol that describes how messages are exchanged between leader and the rest of the network nodes.

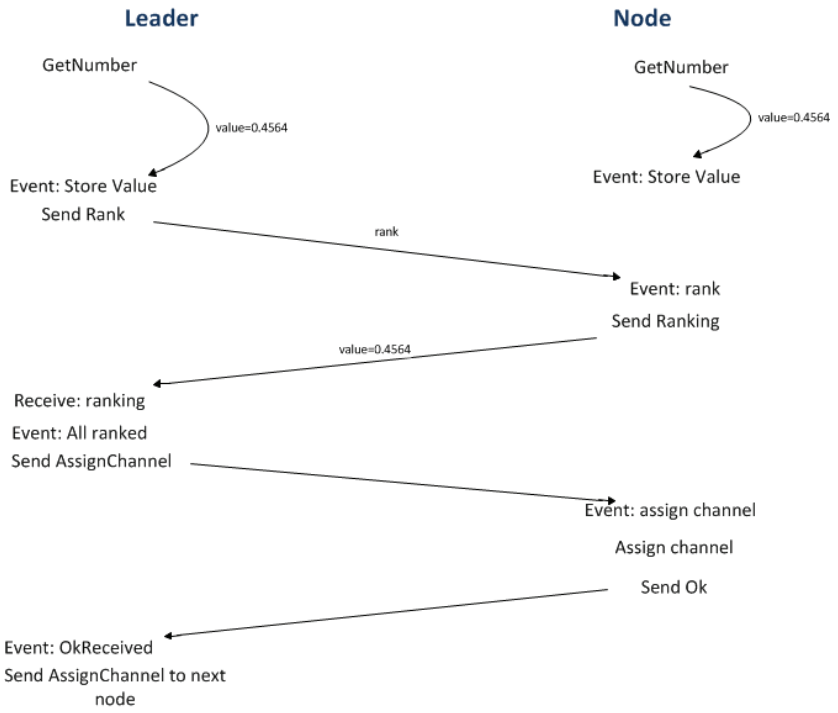

Figure 3.3: Distributed Coordinated Protocol.

3.5 Centralized Solution for Channel Assignment

Last, is the centralized implementation where one node acquires information from all others node in the system and their level of interference. That node will use a backtracking algorithm to obtain the optimal solution in a reasonable time. Then this central node inform each other node in which channel they must operate. This approach is similar to the ones presented in section 2.2.3.

The central node must have enough power to run the centralized algorithm. It can be a notebook, a PC or a router with higher capacity than domestic ones. The central node is implemented in java in contrast to the network entities that run the Ran System implemented in lua. Java node communicate with a local Rnr that connects to the rest of Rnrs in the network. In this implementation java algorithm acts like a big pdp. All other nodes does not have a pdp service.

The protocol works as follows:

Step 1: central node request all other nodes to send it values At startup central node broadcast a message to all Rmoon services in the network.

The message send from the java node to all others rmoon is the following:

```
NOTIFICATION
notification_id=central_not45687
host=monitor_server
target_service=/lupa/rmoon
mib=GetCells
command=watch_mib
message_type=action
op=
value=0
END
```

The things to be noted here are that there is only *target_service* attribute and there is not *target_host* attribute, as we explained in the coordinated solution, this is the way to broadcast a message, and that there is only *host* attribute and there is no *service* attribute, this mean all rmoon in nodes will respond to that host.

Step 2: all rmoon services in network nodes respond to central node Upon received above message each Rmoon in network nodes send back the response message to central node. To respond to the $mib = GetCells$ message, each Rmoon must use *GetAddress* command to get it own identification and *GetCellsInRange* to get it neighbor nodes and the quality of the signal of each of them.

With this information Rmoon send a message like this:

```
NOTIFICATION
notification_id=trap_997422823
service=/lupa/rmoon
host=host1
target_host=monitor_server
watcher_id=noaumc
mib=GetCells
message_type=trap
value=host1#hostx?6?0.8979?-45#hosty?1?0.8546?-54#strangez?1?0.458?-61#
END
```

Note the string assigned to the *value* attribute. This is a string composed of a set of substring separated by the ♯ character. The first substring is the node identification that is obtained from *GetAddress* command. The following substrings are one for each neighbors of the node, with information obtained from *GetCellsInRange* command. Each of this substring is also composed of a set of substrings separated by the ? character. The first part is the neighbor identification, second is the channel that it is using, third is the signal quality and fourth is the signal level.

One important thing to note in the last substring is that the neighbor identification is *strangez* (usually this value will be a mac address). This mean that this is a node that not belong to our network but interfere with our nodes. Central node cannot assign a channel to this node but must take it into account for the computation.

Step 3: central node acquires all network nodes information and compute the best channel assignment for each node Central node knowing the total nodes in the network wait until all nodes respond to the broadcast message. When all nodes have responded then the central node builds a direct graph of the entire network using java structures where each node is a vertex and there is link from v to w if v reported w as it neighbor. There is also a structure named *neighborsConstraint* for each node that maintains the set of nodes that not belong to our network but interferes with nodes in the graph.

When all the structures has been set up then the centralized algorithm is computed to assign each node the best channel assignment. Instead of using brute force enumeration of all possible solution we use a technique known as backtracking to improve the performance of the algorithm. Remember our fourth restriction of the problem that we have only tens nodes. If we are in an environment with hundreds or thousands nodes then we must use another alternatives like genetic algorithms.

The algorithm is implemented as follow: We use fixed length tuples of the form: $< X_0, X_1, ..., X_{n-1} >$, where $n = |V|$ Each component X_i represents the channel assigned to vertex i, for all i, $0 \leq i \leq n-1$

We have the following explicit constraints:

- $X_i \in 1, 6, 11$ for all i, i, $0 \leq i \leq n-1$.
- for each node i, we know "foreign" nodes and the channels C_i used by them.

The objective function is $f = min(interference(G))$, where $interference(G)$ is the sum of interferences that are produced inside the network for a candidate solution tuple having into account the "foreign" nodes also.

The algorithm tries to construct all candidate solutions but as soon as the solution is greater than one previously found it prunes the solution tree.

After the algorithm finishes, the channel that each node must use is stored in java structures, then the algorithm iterates over all nodes information and send a notification message to each pep service requesting it to switch channel. In this case there is no need to wait for the ACK message from pep because the central already calculated the channel for each node, so it instructs each pep as soon as possible.

The notification message to each pep is as follow:

```
NOTIFICATION
notification_id=central_not324657
host=monitor_server
target_service=/lupa/pep
target_host=host1
message=6
command=switchChannel
message_type=action
END
```

Note that in this case there are *target_host* and *target_service* attributes. The command for pep is $command = switchChannel$ and in the *message* attribute is the channel to switch (6 in the example above).

3.6 Summary

In this chapter we formalized the problem of study. Then we presented three strategies to solve the channel allocation problem. The strategies match the kind of approaches presented in the state of the art. We described in detail how each strategy accomplish their work.

We presented a distributed and coordinated approach. In our design a leader must be identified and then this leader is in charge to coordinate the algorithm execution. The leader instruct each node to send it a rank value (that is the interference level measured by each node), then the leader apply a greedy strategy, sorting the nodes with respect to the rank values and asking each one in turn to assign a channel.

Also we presented a centralized solution. In this solution there is a particular node that acquire information of the entire network by mean of receiving information from each participating node. Then this node construct a graph in memory representing the network and use a traditional technique (backtracking) to assign each node a channel in a way that minimize the interference level.

Finally, we presented a local and uncoordinated approach. This approach work locally, ie, with information sensed by the node it takes action in response to that information without any coordination with the rest of nodes in the network. This approach, of uncoordinated operating, can cause to an unstable network re-configuration process. Then, in the next chapter we study the problem from a control-theoretical point of view and provide a feedback control-based solution to bound the oscillations of the solution.

Chapter 4

Feedback Control Applied to Un-Coordinated Solution

Most commercial APs have the option to automatically configure the RF channel where the access point will operate. This auto channel scan option lets the APs to automatically find the less used channel and set it for communication.

But, how does channel auto-tuning work? According to Juniper networks company [38], there are four phases in the auto-tuning mode. These are the four phases of auto-tuning:

- *Measurement Phase:* Most of the time spent in the auto-tune channel algorithm is during the measurement phase. RF data from access points is continually collected during times of typical use.

- *Calculation Phase 1 Enrollment:* At the end of the measurement phase, each access point is polled to determine the most recent state of the access point. Discovery is done in the background with no impact on other system functions. During this enrollment portion of the calculation phase, the domain is bootstrapped from a seed radiothis continues until all domain members are discovered.

- *Calculation Phase 2, Optimization:* When all access point discovery has taken place, the access points begin a process of proposing and considering hypothetical channel moves to and from other access points. The results of these moves is calculated to determine whether the new topology is better than the existing one. Any proposed changes are stored by the mobility domain and are ready for deployment, and one optimal solution is selected.

- *Deployment Phase:* During the deployment phase, all proposed channel changes are applied to all access points in a domain simultaneously. You can schedule the time of the deployment phase or the solution can be deployed immediately. All client-affecting channel changes in an interference domain should be completed within the time a client detects signal failure and tries to re-associate. Few clients, if any, should to have to roam twice. For this reason they recommend that deployment occurs during times when there would be minimum client impact.

By default, the controller evaluates the scan results for possible channel changes every 3600 seconds (1 hour). A distributed algorithm on controllers and access points can cause the system to tune to another channel. If the radio has active sessions, the controller does not change the channel unless it detects radar or severe interference on the channel. RF auto-tuning also can change a radio channel when the channel tuning interval expires, if a channel with less interference or radar is detected. Disturbance is based on the number of neighbors the radio has and the RSSI of each neighbor. A radio also can change channels before the channel-tuning interval expires to respond to interference or radar.

In contrast, our implementation of un-coordinated approach keeps monitoring the channels condition and if it detects that there are a new least congested channel then the system instructs the access point to change channel. However, in an environment such as a conference room or a dense urban area in which each AP uncoordinatedly and continuously performs this process, each AP may fall in a cascade of channel re-configurations that will make unstable the whole system.

So, in this work we search for a mechanism to dynamically change channels in a controlled manner, when and if better conditions are found and without cause the instability of the system. In this way, the system can adapt to changing environments with the aim at minimizing the total interference level seen by the APs. This motivates us to apply a feedback control technique with the goal of bounding the inherently oscillatory behavior of the solutions.

The remain of the chapter is as follow. The next section presents oscillatory issues we found in our implementation of the uncoordinated solution. In section 4.2 we review the feedback control framework when applied to computing systems. Section 4.3 describes how our target system, the uncoordinated solution, is embedded into a closed-loop to achieve the objective of minimize the number of channel switches per minute. Section 4.4 details our approach to system identification and section 4.5 discusses controller design and uses empirical studies to assess the accuracy of insights obtained from control theory. Finally, we provide some conclusions of our work.

4.1 Oscillatory issues

As expected, with the un-coordinated solution deployed in a testbed (see chapter 5), the AP's behavior strongly depends on the *threshold* value (refer to 3.3). When the *threshold* value is set equal to 0.0 the APs keep changing channels continuously. When the *threshold* value is set to 1.0, the access points change its channel only once and then remains in that channel for the rest of the experiment. Any intermediate value causes a behavior as shown in Figure 4.1. In this figure we show that if we fix the *threshold* value in 0.6 and we start the system three times and let it run during five minutes, we get different number of channel re-configurations. During the first run there are fourteen channel changes in five minutes, in the second run ten, and during the third run eleven changes.

While in the Figure 4.1 we showed the cumulative number of changes as time pass, in Figure 4.2 we show another view of the same executions that help to see the variability of the number of channel changes per unit of time.

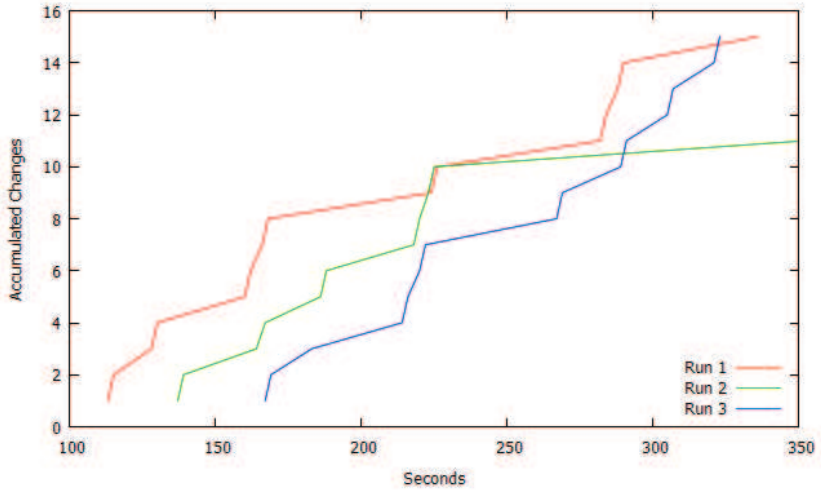

Figure 4.1: Cumulative number of channel re-configurations in a five-minute period (threshold = 0.6).

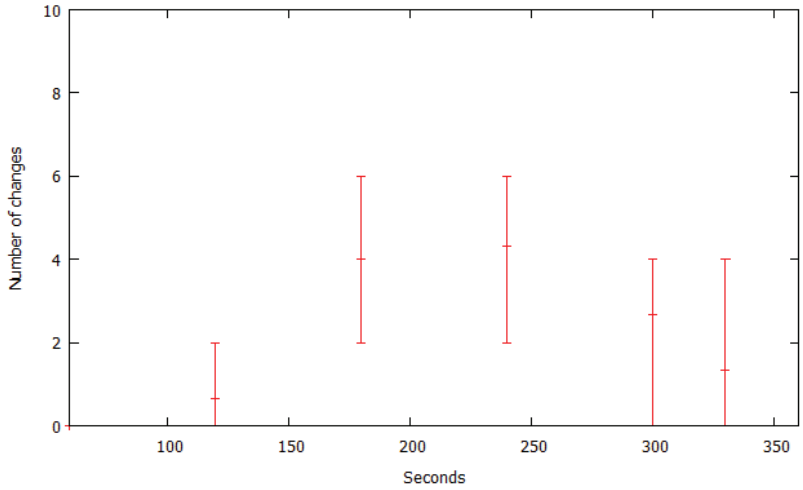

Figure 4.2: Average of channel changes with minumum and maximum for various executions (threshold = 0.6).

More precisely, in the Figure 4.2 we show the average of channel changes per minute in three different executions, i.e. in a time t the figure show the number of channel re configurations between t and $t-1$. For example, after two minutes, the average of channel changes between minute one and minute two is 0.66 with two channel changes as the maximum and zero channel changes as the minimum. In the third minute, the average is four channel changes between minute three and minute two, with a maximum of six and a minumum of two. The important to note here is in the variability of the results. There are no relation between to consecutive average values and if you observe carefully both Figure 4.1 and Figure 4.2, the difference between maximum and minimum with respect to the average is also variable. The minimum at time three minute is measured in the execution labeled *Run 3*, but the maximum at time 330 seconds is also measured in the execution labeled *Run 3*.

In this way, we can observe the unpredictability on the number of channel changes for a given deployment. Our conclusion is that, in our implementation, the number of channel re-configurations is unstable.

The first observation is that, in some way, the uncoordinated solution is trying to solve a combinatorial optimization problem by trial-and-error. Since the information available at each node is limited, and the system highly dynamic, it does not seems to be a very bad idea. Actually, the intrinsic unstability of the solution allows to try a high number of solutions in a short period of time. The obvious drawback is that after trying all possible combinations of channel distribution among APs, the system keeps changing making it unusable. Thus, we want a solution that is fast and flexible at start up and relatively stable in the long run. It needs also to allow changes from time to time to adapt to environmental changes.

Our goal is to develop a mechanism to bound the oscillations of the solution while having a quick adaptation to the environment during boot-time and maintaining the adaptability to change in the long run. To do this we will apply feedback control techniques. In the next section we will describe feedback control as applied in computing systems.

4.2 Feedback Control Concepts

We pretend to apply a well known strategy for traditional engineering discipline named feedback control with the goal of make the system more stable. We based our study on a reference book in this area "Feedback Control of Computing Systems"[28] Also we studied a couple of paper that the same authors wrote where they applied the feedback control theory to the apache web server [23] and to the IBM Lotus Notes Server [40]. In the first case the authors control a desired CPU and memory utilization indirectly by manipulating tunning parameters such as MaxClients and KeepAlive, whereas in the second case the authors control the queue length of the queue of in-progress RPC requests (it is a client server system) by tunning parameter $SERVER\_MAXUSERS$. In the same way we will control the number of channel switch in a access point by adjusting the *threshold* value. Remember that the condition to switch channel are: (1) exits a channel with less interference than the one currently in use, (2) and the level of interference in channel used is upper than a certain *threshold* value. Note that if we set the *threshold* to a value of 1.0 they never change channel

because is impossible to find a channel with interference level greater than 1.0 and by the other hand if we set the *threshold* to a value of 0.0 it will always find a better channel and it will keep changing channel constantly (the way we implement the interference level is a value between 0 and 1). So, the intent is to adjust the threshold value dynamically so that the access point can find a channel with a good level of interference.

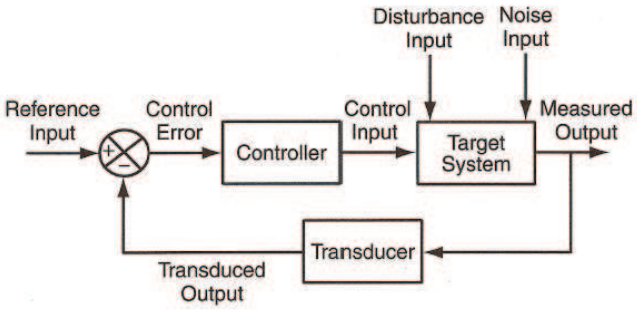

Figure 4.3: Feedback Control in Computing System. Source [28].

The essential elements of feedback control system are depicted in Figure 4.3. These elements are:

- *Control error*, which is the difference between the reference input and the measured output
- *Control input*, which is a parameter that affects the behavior of the target system and can be adjusted dynamically
- *Controller*, which determines the setting of the control input needed to achieve the reference input. The controller computes values of the control input based on current and past values of control error
- *Disturbance input*, which is any change that affects the way in which the control input influences the measured output
- *Measured output*, which is a measurable characteristic of the target system, such as CPU utilization and response time
- *Noise input*, which is any effect that changes the measured output produced by the target system
- *Reference input*, which is the desired value of the measured outputs, such as CPU utilization, should be 66
- *Target system*, which is the computing system to be controlled.

The classical controller design methodology consist of two steps:

1. *System identification:*, Construct a transfer function which relates past and present input values to past and present output values. These transfer functions constitute a model of the system.

2. *Controller design:*, Based on properties of the transfer function and the desired objectives, a particular control law is chosen. Techniques from control theory are used to predict how the system will behave once the chosen controller is added to it.

Much work have been done that used first-principles approach to system identification. Although these efforts suggest that first-principles approaches can yield success, they also indicate that considerable sophistication is required to do so. Further, for complex computing systems, constructing first-principles models may be extremely difficult, if not impossible. In that case, we have to rely on empirical models, that is, models based on data collected from an actual system.

4.3 Un-Coordinated Solution and Its Closed-Loop Control

As we saw in previous chapters, we implement the un-coordinated solution for channel allocation strategy using RAN as their control communication bus. Architecturally the nodes in RAN System, as in the well-known policy-based architecture, are composed by a Policy Decision Point (PDP), an Enforcement Point (EP) and an agent that monitors the state of the device and the behavior of the network. The PDP follows the event-condition-action (ECA) paradigm and the notifications sent by the monitoring agent of a node of the network is the source of events for the PDP of the same node. In the uncoordinated approach RMoon send two kinds of events to PDP forever: (1) if exist or not a least congested channel than current channel, and (2) the level of interference in the current channel. PDP module then evaluate two conditions to determine if it is worth to switch channel, that really exist a channel less congested than the current channel and that the level of interference is greater than a threshold. We consider this threshold value as tunning parameter. If we set the parameter to a value of 0, then the second condition always met whereas if we set it to a value of 1 it will never succeed. As such, this parameter has a somewhat complex effect on stability of the system.

The measured output we will be considering here is the amount of switch channel per minute. Clearly, the intention here is to reduce the frequency of changes. If this happens the meaning is that the node found a good channel to operate. Intuitively in the first minute the node can switch channel more often but as time pass the node will adjust threshold and it will remain in a good channel with a reasonable interference level.

In line with the objective of a system that starts very dynamic to try as many solutions as possible, and then converges to a stable state, the reference value in this closed loop control system will be the number of channel changes in a time period (for our testbed we will say x changes in 1 minute). This value specifies a management policy that the closed loop system tries to achieve. Based on the current and past values of the control error (the difference between the reference value and measured amount of channel switch), the controller adjusts the value of the threshold. The algorithm for computing this adjustment is called the control law.

The main challenge of this work is about how to treat the interference we

receive from the other nodes. The problem here is that neighbors node are using the same uncoordinated solution and so they are changing channel in the same pace.

Studying the reference papers mentioned above where the authors treat the RPCs from Users as inputs for the close-loop control of Lotus Notes in [40], we will consider the interference input as input to our system. Later we will discuss how it is related to the *threshold* value and which will be our operating region.

4.4 System Modeling

The difficulty of constructing firs-principles models of computing systems motivates an approach that requires a less detailed knowledge of the relationships between inputs and outputs. Statistical techniques have considerable appeal since we reduce the knowledge required for model construction. That is, instead of employing detailed knowledge of the target system, we infer relationships between inputs and outputs by applying statistical techniques to data collected from the target system. The term *black-box model* is used since only the inputs and outputs of the target system need be known.

In system identification, we identify the main components of the system, the data values that flow in and out of them, and the mathematical relationships between these values. This section describes our approach to system identification and its application to Uncoordinated RAN System.

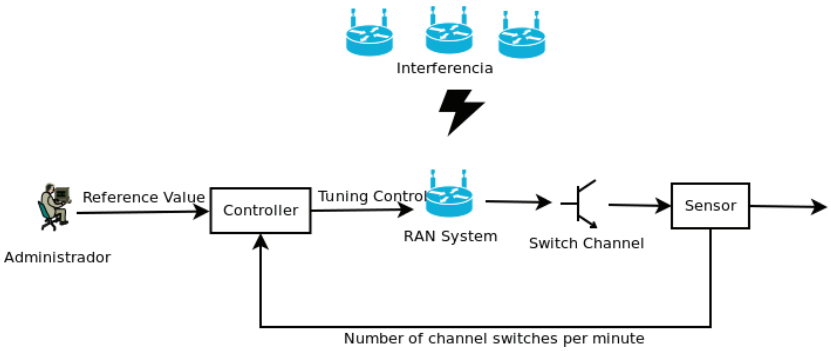

Figure 4.4: Feedback Control in RAN System.

We construct a block diagram as shown in Figure 4.4 for component identification. We see that interference level and *threshold* are inputs to the Uncoordinated RAN System, and the channel switch is the output. Since the interference level has an indirect effect on *threshold* value, the two inputs are not independent and hence they do not combine in a linear fashion. So is important to consider the operating region of the system. The *operating region* of a system is the range of control inputs (and their associated outputs) that are observed in operation. Then we divide the operating region in two regions. When the $threshold > interferencelevel$, the tunning parameter has no effect so we can ignore this case. When $threshold \leq interferencelevel$, exactly threshold value is the level interference allowed, then the interference level value is not relevant (as long as we stay in this region) and hence it can be ignored. This

mean, we will not consider interference level as an input. This results in a SISO (single-input, single output) model.

To model the system we assume that time is discrete with uniform interval sizes. Then the relationship between control inputs and measured outputs can be quantified by linear difference equations:

$$y(k+1) = a.y(k) + b.u(k) \tag{4.1}$$

where u is the input signal and y is the output signal.

4.4.1 Experimental Design and Parameter Estimation

Constructing black-box models requires data to estimate model parameters such as a and b in Equation 4.1. To collect data we set up a laboratory infrastructure with six access points running the uncoordinated approach. We will be varying *threshold* value in only one of them and measure the number of channels switch in five minutes.

The results are shown in Table 4.1

Table 4.1: Data Used to estimate parameters

k	$\tilde{u}$(k)	$\tilde{y}$(k)	u(k)	y(k)
1	1	0	0.4	-20.33
2	0.9	4	0.3	-16.33
3	0.8	6	0.2	-14.33
4	0.7	13	0.1	-7.33
5	0.6	14	0	-6.33
6	0.5	24	-0.1	3.67
7	0.4	29	-0.2	8.67
8	0.3	28	-0.3	7.67
9	0.2	28	-0.4	7.67
10	0.1	37	-0.5	16.67

To estimate model parameters once data have been collected we use a commonly-used method called *least squares* regression. We proceed as follows:

1. The mean *threshold* value and amount of channels switch are computed. The former is computed over $[1, 9]$ (since the last value is not used in the estimates produced) and the latter over $[2, 10]$. The average of these values yields the following results $\bar{u} = 0.6$ and $\bar{y} = 20.33$

2. y and u are computed as shown in table , ie as the difference between $\hat{u}$ and $\bar{u}$ and the same for y.

3. As described in section 2.4.3 of the reference book of this work we calculate the following values:

   $S_1 = \sum_{k=1}^{N} y^2(k)$

   $S_2 = \sum_{k=1}^{N} u(k)y(k)$

$$S_3 = \sum_{k=1}^{N} u^2(k)$$

$$S_4 = \sum_{k=1}^{N} y(k)y(k+1)$$

$$S_5 = \sum_{k=1}^{N} u(k)y(k+1)$$

Note that the S_i are computed using $k \in [1, 9]$.

Table 4.2: Values of S_i

i	S_i
1	1185.42
2	-24.1
3	0.6
4	979.21
5	-24.3

4. From the values of S_i we compute the parameters as:

   $a = \frac{S_3S_4 - S_2S_5}{S_1S_3 - S_2^2}$

   $b = \frac{S_1S_5 - S_2S_4}{S_1S_3 - S_2^2}$

   We have $a = 0.0145$ and $b = -39.92$

   (b is negative since a larger *threshold* decreases channel switches).

5. With this, we can calculate $\hat{y}(k+1) = ay(k) + bu(k)$

If we passing that with different data, we get different estimates. In general, having more observations decreases the variability of parameter estimates.

4.4.2 Model Evaluation

There are various metrics for assessing accuracy of the model: the root-mean-square error (RMSE) and the *correlation coefficient* (CC). Another way to quantify accuracy is by computing the variability explained by the model. This is denoted R^2 and

$R^2 = 1 - \frac{var(y - \hat{y})}{var(y)}$

where $var(y)$ is the variance of the $y(k)$. R^2 ranges from 0 to 1. A value of 0 means that the model does no better than using the mean value of y to estimate $y(k)$. A value of 1 suggest but does not guarantee a perfect fit.

The value of R^2 that we get is 0.94, that is quite good.

4.4.3 Analysis using Z-Transform Theory

Z-transforms can be used to describe systems. Such a description is called a transfer function. A *transfer function* of a system describes how an input $U(z)$ is transformed into the output $Y(z)$.

We define the transfer function $G(z)$ as

$G(z) = \frac{Y(z)}{U(z)}$

That is, $G(z)$ is a ratio of polynomials in z. The denominator polynomial $D(z)$ is referred as the *characteristic polynomial*. The *characteristic equation* is obtained by setting the characteristic polynomial to 0. The roots of this equation are called the *poles*.

For a linear difference equation of the form $y(k+1) = a.y(k) + b.u(k)$ the transfer function is:

$G(z) = \frac{Y(z)}{U(z)} = \frac{b}{z-a}$

One of the most important properties of a system is stability. Intuitively, we might think of a system with an unbounded output as unstable. So, we are focusing on bounded-input, bounded output (BIBO) stability. Consider the following theorem:

Theorem 1 ***BIBO stability*** *A system represented by a transfer function $G(z)$ is BIBO stable if and only if all the poles of $G(z)$ are inside the unit circle.*

Another dynamic property that can be determined by the pole is speed of response. As $|a|$ approaches to zero, which is our case, the speed of response to changes in the input becomes faster. If $a = 0$, then $y_{k+1} = b.u_k$. Hence, it takes one time step for the output y, to converge after changes in the input. As $|a|$ approaches one, it takes more time steps for the output to converge to a steady-state after changes in the input.

The final property that can be determined by the pole is whether the system will respond in an oscillatory manner to changes in the input. If $a < 0$, then the behavior of the system will be oscillatory.

The pole of our transfer function is $a = 0.0145$. So we can say that our system is stable, have a faster response time and it is not oscillatory.

4.5 Control Analysis and Design

4.5.1 Control Laws and Controller Operation

We need to specify a *control law* that quantifies how to set the control input to the target system.

In this work we applied the *proportional integral control law* that has the form

$$u(k) = u(k-1) + (K_p + K_l)e(k) + K_p e(k-1) \tag{4.2}$$

where $u(k)$ is the output of the integral controller and $e(k)$ is the control error. In other words $e(k)$ is the difference between the desired and actual values of the output, that is $e(k) = r(k) - y(k)$. $r(k)$ denote the *reference signal* and it is the desired output value that we must explicitly specified.

Proportional integral (PI) control combines the integral control with the proportional control.

The *proportional control law* is

$$u(k) = K_p e(k) \tag{4.3}$$

where K_p is a constant that is chosen when designing the proportional controller. K_p is often referred to as the controller gain of the proportional con-

troller. The transfer function of a proportional controller is

$$\frac{U(z)}{E(z)} = K_p \tag{4.4}$$

The integral control law has the form

$$u(k) = u(k-1) + K_l e(k) \tag{4.5}$$

The controller parameter K_l defines the ratio of control change (the difference between the current and past inputs) to the control error. The transfer function of the integral controller is:

$$\frac{U(z)}{E(z)} = \frac{K_l z}{z-1} \tag{4.6}$$

PI control combines the advantages of integral control (zero steady-state error) with those of proportional control (increasing the speed of the transient response).

The transfer function of the integral controller is:

$$\frac{U(z)}{E(z)} == K_p + \frac{K_l z}{z-1} \tag{4.7}$$

The PI controller has a pole at $z = 1$ (this corresponds to the integral action). The control transfer function has a finite zero at $z = \frac{K_p}{K_p+K_l}$. If K_p and K_l have the same sign (as is usually the case), the zero is always on the real line between 0 and 1. When the zero is exactly at 0, PI control reduces the pure integral control case. When the zero is exactly at 1, it cancel the pole at $z = 1$, negating the effect of the integral control, and reduces the pure proportional control case.

4.5.2 Desirable Properties of Controllers

Designing feedback systems requires having clear criteria for what makes one controller preferable to another. In computing systems, we focus on four properties: stability, accuracy, settling times and overshoot. We refer to these as the SASO properties.

The most basic property of a controller is stability; that is, it results in a stable closed-loop system. Stability is assessed by determining if the poles of the closed-loop transfer function have a magnitude less than 1.

A second property that we desire in a controller is that it be accurate. We quantify the accuracy of a closed loop in terms of steady-state error. We can assess the accuracy of a closed-loop system by computing the steady-state gain of its transfer function from the reference input to the measured output. There is a zero steady-state error if and only if this steady-state gain is 1.

A third property of interest is the settling time of the system, the time for the output to reach a new steady-state value after a change in one of the inputs. We use K_s to denote the settling time. Settling time is a function of the closed-loop poles and is estimated using $K_s \approx \frac{-4}{log|a|}$ by employing the dominant pole approximation.

The final property we consider is *maximum overshoot*. Maximum overshoot, which we denote by M_p, is the largest amount by which the transient response exceeds the steady-state value as a result of a change in an input, scaled by the

steady-state value. Overshoot is a property of the response to a step change in the reference input. Overshoot occurs if one or more poles with a nonzero angle in the complex plane.

The purpose of control analysis is to ascertain the SASO properties of the closed-loop system. The purpose of control design is to construct a closed-loop system with the desired SASO properties.

4.5.3 PI Control Design

We have four design goals for the PI controller: (1) the closed-loop is stable; (2) steady-state error is minimized; (3) settling time does not exceed K_s^*; and (4) maximum overshoot does not exceed M_p^*

The first design goal is achieved by ensuring that all poles lie within the unit circle. The second goal is achieved by using a PI controller, at least for a step change in the reference and/or disturbance inputs. Thus, the design problem is reduced to goals 3 and 4. These control goals can be achieved by properly selecting the parameters K_p and K_l of the PI controller.

A commonly used approach to controller design is pole placement, in which the closed-loop system poles are chosen to meet some desired criteria. The steps in pole placement control design are outlined below. The transfer function of our system:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b}{z-a} \tag{4.8}$$

Recall that $y(k)$ is the offset of amount channel switches per minute from the operating point, and $u(k)$ is the offset of $THRESHOLD$ from the operating point. The design goals are that settling time does not exceed $K_s^* = 10$; and the maximum overshoot does not exceed $M_p^* = 10\%$

1. Compute the dominant poles. The first step is to compute the desired poles of the closed-loop system based on K_s^* and M_p^*. There is an assumption that the the poles are complex conjugates $re^{\pm j\theta}$. We have an upper bound for r that is $r = e^{\frac{-4}{K_s^*}}$. So we have $r = e^{\frac{-4}{10}} = 0.67$. Having calculated r then we can calculate θ as $\theta = \pi \frac{logr}{logM_p^*}$. Using that, we determine that $\theta = \pi \frac{logr}{log0.1} = 0.7$

2. Construct and expand the desired characteristic polynomial. The desired characteristic polynomial is $z^2 - 2rcos\theta z + r^2 = z^2 - 1.2z + 0.36$

3. Construct and expand the modeled characteristic polynomial. The modeled characteristic polynomial is the denominator of

$$\frac{k(z)G(z)}{1 + k(z)G(z)} \tag{4.9}$$

where

$$k(z) = \frac{(K_p + K_l)z - K_p}{z - 1} \tag{4.10}$$

Eliminating all fractions in the denominator (by multiplication) and expanding the polynomial we get: $z^2 + [b(K_p + K_l) - 1 - a]z + a - bK_p$ replacing a and b for the values obtained in section 4.4.1 we get: $z^2 + [-39.92(K_p + K_l) - 1.0145]z + 0.0145 + 39.92K_p$

4. Solve for K_p and K_l.

   We want the desired characteristic polynomial to equal the modeled characteristic polynomial. That is $z^2 - 1.2z + 0.36 = z^2 + [-39.92(K_p + K_l) - 1.0145]z + 0.0145 + 39.92K_p$

   So we have:

$$-1.2 = -39.92(K_p + K_l) - 1.0145$$

$$0.36 = 0.0145 + 39.92K_p$$

   Solving, the system of equations, we have:

$$K_p = 0.009$$

$$K_l = -0.004$$

5. Verify the result. The closed-loop transfer function from the reference input to the measured output is

$$F_R(z) = \frac{Y(z)}{R(z)} = \frac{([(K_p + K_l)z - K_p]/(z-1))G(z)}{1 + ([(K_p + K_l)z - K_p]/(z-1))G(z)} \tag{4.11}$$

$$F_R(z) = \frac{-0.2z + 0.36}{z^2 - 1.2z + 0.36} = \frac{-0.2z + 0.36}{(z - 0.6)(z - 0.6)} \tag{4.12}$$

   As expected, the poles of $F_R(z)$ are 0.6, so the system is stable. Also, $F_R(1) = 1$ and hence there is no steady-state error to a step change in the reference or disturbance inputs.

4.6 Evaluation

The goal of the evaluation is to show that using the method developed in this chpater it is feasible to control the system stability. Using the feedback control technique we can bound the number of channel changes for a given time period. The other alternative is to have an algorithm that scan for possible channel changes every t time period, for example every 3600 seconds (1 hour). But we do not want to wait until the next run if channel conditions become poor. We want to continuously monitor channel conditions and let the AP change to a better channel if necessary to keep the quality of the service. But we need to do this in a controlled manner to avoid a disruptive service. One way to control and bound the channel changes is by using a feedback control technique and we devote this section to evaluate the solution by presenting empirical assessments.

4.6.1 Empirical Assessments

Here, we present empirical results for the feedback control technique applied to our un-coordinated solution for the channel allocation problem in Wireless Network. The empirical results were obtained from a subset of the testbed presented in section 5.3. In this case we use only six of the ten access points available. The metric we use to evaluate the system are the number of channel changes per minute. We measure in a time t, the number of channel changes in the period t and $t-1$. In this work the time unit is one minute. Our hope is that as time pass, the number of channel changes is bound to a configured value.

We showed in section 4.1 how the system behaves before we applied the technique presented in this work. We led to the conclusion that the oscillatory issues are not able to be predicted. For this reason, we developed a feedback control technique with the aim of controlling the number of channel changes.

To evaluate the system, we install the uncoordinated solution enhanced with the feedback control technique (we do this in the policy decision point component) in each access point of the testbed. We set the *threshold* to a value equal to 0.1 and a $referenceinput$ value equal to 5. The meaning of this value is that we want to allow until five channel reconfigurations per minute. We let the system run for ten minute period and we obtain the results shown below. For the experiment we did ten executions of the system.

In Figure 4.5 we show the average of the control error in all access points per minute. Remember, control error is the difference between the reference input and measured output. The reference input was set to a value equal to 5. Observe that at time pass the control error is bound between a value greater than -5 and a value less than 5. A value of 0 means that the measured output is five channel changes in the last minute. By setting the reference value equal to 5 we are allowing to have five channel re-configurations per minute. If you observe the Figure the average value keeps close to zero and this happen in all executions. Contrast this with the behavior shown in Figure 4.2 where the number of channel changes are unpredictable.

Moreover, in Figure 4.6 we show how the *threshold* value varies with the time for one particular execution. As was mention before we set the initial value of the *threshold* equal to 0.1. Remember that the *threshold* is a value in the range of $[0, 1]$. Then, as the system became more stable the *threshold* value tend to be equal to 0.05.

The testbed allow us evaluate how the predictions made by control theory compare with the behavior of the real system. We updated our implementation by embedding the *proportional integral control law*. This control law has the form depicted in equation 4.2.

In our case, $u(k)$ is the *threshold* value and $e(k)$ is the control error which is the difference between the reference input (set to 5) and measured output. The measured output is the number of channel changes and is calculated every minute. Having successfully updated our system we could evaluate how the system self-manage the *threshold* value, by adjusting the value so that the difference between the measured number of channel changes is in close proximity to the value set as the desired number of channel hopping. In Figure 4.6 we showed how the *threshold* value converge to a value close to 0.05 to allow the system to have a number of channel changes per minute around a value of 5

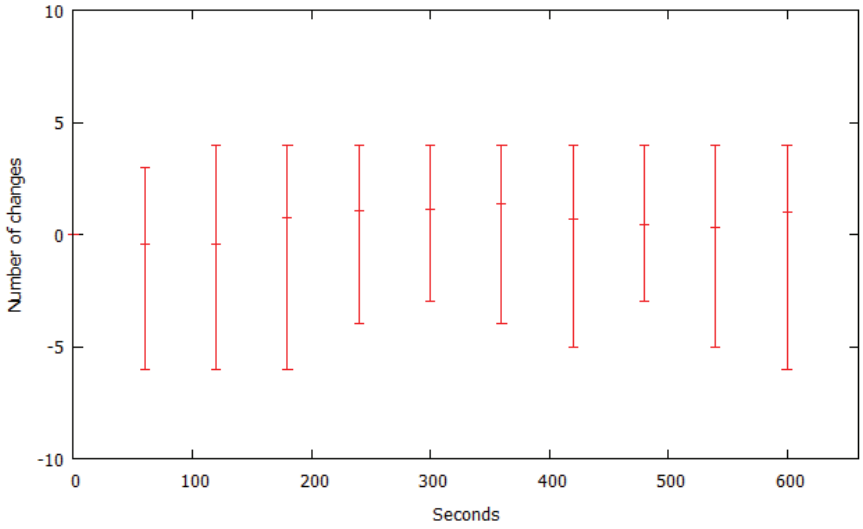

Figure 4.5: Control error per minute in six different APs (initial threshold=0.1).

changes as showed in Figure 4.5.

This is our first approximation to study the validity of the feedback control technique applied to an un-coordinated solution for channel allocation in wireless networks with the aim to help in the stability issues. We will schedule for future works far more scenarios that those presented here to get more statistics and thus get more robust conclusions.

4.7 Conclusions

In this chapter, we have introduced a methodology for constructing and analyzing closed-loop system using a statistical approach to system identification. This approach is more generally applicable than conventional first-principles approach because of the difficulty of constructing first-principles models. In first-principles models we need to know the internal relationships between inputs and outputs to find the mathematical model. In our model by applying statistical techniques to data collected from the target system we could infer relationships between input and outputs.

In section 4.4.2 we presented R^2. R^2 ranges from 0 to 1. A value of 0 means that the model does no better than using the mean value of y to estimate $y(k)$. A value of 1 suggest but does not guarantee a perfect fit. The fit for our models of the UnCoordinated RAN System is: $R^2 = 0.94$ that is quite good.

With this work we can control the number of channel changes in a given fraction of time. We do not eliminate the channel changes but we can control and bound the velocity of changes. This is a reasonable behavior for our un-coordinated implementation. Moreover, this allows to have a system that is capable of changing at the beginning to try multiple possible configurations in a distributed and un-coordinated manner and, then, to stabilize to make the system usable.

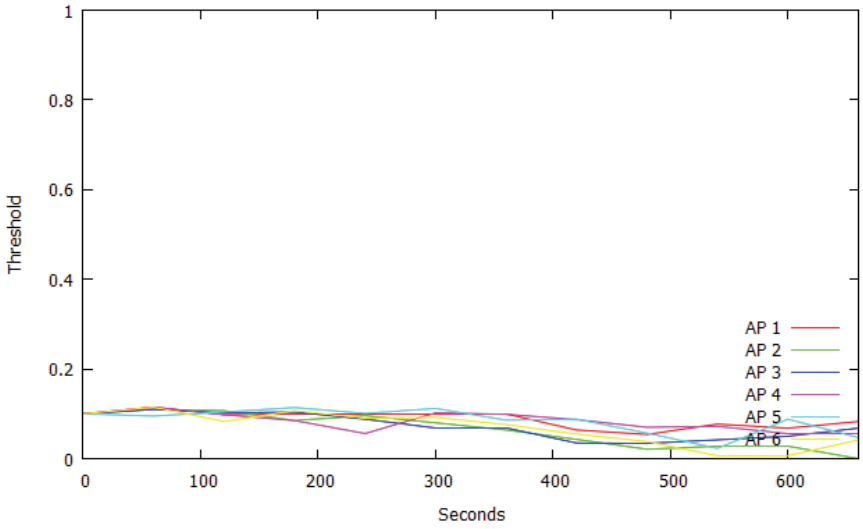

Figure 4.6: Variability of threshold value per minute (initial threshold=0.1).

We can go further by adding some kind of intelligence to the system. When we scan the frequencies use we get the MAC address of the emitting APs. We can use this knowledge to better manage the way we handle the changes of channels. But it is out of scope of this work.

Chapter 5

Model Evaluation

5.1 Introduction

This chapter performs the validation of the proposed strategies. Conceptually, this chapter can be divided into two main topics: a proof of concepts and a scalability evaluation. The proof of concepts is made by deploying the algorithms in a testbed scenario. For the scalability evaluation we perform simulations where can assess the behavior of the strategies when the number of nodes increase.

The rest of the chapter is as follow. In the next section we present the motivation of the proof of concepts. In section 5.3 we present the testbed scenario and in section 5.4 we present the empirical result of the testbed. Then, in section 5.5 we present the motivation for a simulation-based evaluation. Next, in sections 5.6 and 5.7 we present the network simulator used and the hosting environment under which simulation run. Finally in the last section we show results by comparing the L_{ran} values obtained from different simulation scenarios.

5.2 Proof of Concepts

We started by implementing a testbed scenario to prove that the implementations are technically feasible. To do this we need to setup a set of access points and deploy our algorithms in each of them. The scalability of the solutions will be measured by simulations, so we only need a small set of access points for the testbed. Ideally we need more than three access points for the test to be meaningful. So as to be sure that there will be interference between participating access points, due we will only make use of the three non-overlapping channels. Moreover to avoid having the same number of access points in each channel we considered that the amount of access points must not be a multiple of three. For this consideration we finally decided to have ten access points for our testbed implementation.

Also, during the testbed we want to take measures to get some initial insights about the behavior of each strategy. So, after each strategy run we measure the interference level at each access point and then we can calculate the L_{ran} value for the entire system. This will give us some knowledge that we will study deeply in the simulation.

The testbed part of this chapter is as follow. In the next section we present the hardware specification and details of the testbed and then in section 5.4 we present the results of each implementation.

5.3 TestBed Description

In the context of this work a fully working deployment was implemented in a testbed. This laboratory setup is a network with ten wireless access points. Seven access points are LinkSys WRT160NL and three access points are LinkSys WRT54GL. These access points are designed as consumer-level devices that the vendor has trimmed down as much as possible.

The main characteristic of the LinkSys WRT160NL are shown in Table 5.1.

Table 5.1: LinkSys WRT160NL

Feature	Value
Platform	Atheros AR9130
CPU Speed (MHz)	400
Flash (MB)	8
RAM (MB)	32
Wireless NIC	Atheros AR9100 (integrated)

The main characteristic of the LinkSys WRT54GL are shown in Table 5.2.

Table 5.2: LinkSys WRT54GL

Feature	Value
Platform	Broadcom 5352
CPU Speed (MHz)	200
Flash (MB)	4
RAM (MB)	16
Wireless NIC	Broadcom (integrated)

For the coordinated protocol and the centralized protocol a communication network must be provided. To achieve this we also added a TP-LINK desktop switch model TL-SF1008D with 8 ports and 10/100 Mbps. Each access point has 4 ports so we use ethernet to make the link between all nodes.

For the centralized protocol we also added a notebook capable of running a java program. The notebook is an Acer Aspire with AMD Dual-Core Processor E-350 with 2 GB DDR3 Memory and 250 GB HDD. The operating system is Ubuntu 12.04 LTS of 32 bit with kernel version 3.2.0. Java version is 6 update 27. This notebook is connected via cable to one of the switch ports.

For the testbed, the original Linksys operating system has been substituted by OpenWRT Backfire 10.03.1 [12], a linux distribution for embedded devices.

The procedure we follow was to install each strategy in turn and run the testbed. First of all, we deployed the solution for the local and uncoordinated approach and we observed the oscillatory issues that we addressed in chapter 4. But also we found some very interesting results in the interference level reached by the uncoordinated solution.

Then we proceed to install the solutions that exchange messages between nodes: the coordinated and centralized solutions. For both of these strategies we need to configure network connectivity in each access point. We wired all access points and configured a Local Area Network with fixed IP addresses for the RAN System be able to exchange messages between nodes. Both strategies run to the end and yield results that at first glance are no so promising but are the basis to stimulate analysis in simulation.

5.4 TestBed Results

We set up all the access points in channel 1. It is important to notice that in houses close to the building where we deploy the laboratory there were other access points which interfere with our deployment. In Table 5.3 are the list of our access points in the initial state and in Table 5.4 are the list of neighbors AP that we do not manage.

Table 5.3: Initial set up

Ip	Mac Address	Channel
13.13.13.1	68:7F:74:26:3C:36	1
13.13.13.110	68:7F:74:26:3F:DE	1
13.13.13.111	68:7F:74:26:3E:E2	1
13.13.13.112	68:7F:74:26:39:DB	1
13.13.13.113	68:7F:74:26:3F:BD	1
13.13.13.114	68:7F:74:26:39:D5	1
13.13.13.115	68:7F:74:26:3B:04	1
13.13.13.116	C0:C1:C0:01:25:B3	1
13.13.13.117	C0:C1:C0:01:27:C9	1
13.13.13.118	C0:C1:C0:01:26:D6	1

Table 5.4: Neighbor APs

ESSID	Mac Address	Channel
TuleWiFi	00:25:12:6D:17:3D	4
Gaby	98:FC:11:BD:66:B4	9
wifigonza	00:23:54:DB:0A:10	1
tatocine	C8:6C:87:A0:4A:41	7
gatito	94:44:52:7E:E0:0E	2

After running each of our implementations we can see the new channel assignment: in Table 5.5 we can see the result of the uncoordinated implementa-

tion, in Table 5.6 we can see the result of the coordinated implementation and in Table 5.7 we can see the result of the centralized implementation.

Table 5.5: UnCoordinated Result

Ip	Mac Address	Channel
13.13.13.1	68:7F:74:26:3C:36	6
13.13.13.110	68:7F:74:26:3F:DE	6
13.13.13.111	68:7F:74:26:3E:E2	11
13.13.13.112	68:7F:74:26:39:DB	11
13.13.13.113	68:7F:74:26:3F:BD	11
13.13.13.114	68:7F:74:26:39:D5	1
13.13.13.115	68:7F:74:26:3B:04	1
13.13.13.116	C0:C1:C0:01:25:B3	1
13.13.13.117	C0:C1:C0:01:27:C9	1
13.13.13.118	C0:C1:C0:01:26:D6	1

Table 5.6: Coordinated Result

Ip	Mac Address	Channel
13.13.13.1	68:7F:74:26:3C:36	1
13.13.13.110	68:7F:74:26:3F:DE	6
13.13.13.111	68:7F:74:26:3E:E2	1
13.13.13.112	68:7F:74:26:39:DB	6
13.13.13.113	68:7F:74:26:3F:BD	6
13.13.13.114	68:7F:74:26:39:D5	11
13.13.13.115	68:7F:74:26:3B:04	11
13.13.13.116	C0:C1:C0:01:25:B3	11
13.13.13.117	C0:C1:C0:01:27:C9	6
13.13.13.118	C0:C1:C0:01:26:D6	11

In the uncoordinated implementation five access points are in channel 1, three are in channel 11 and two are in channel 6, while in the coordinated implementation four access points are in channel 6, another four access points are in channel 11 and two are in channel 1, finally in the centralized implementation five access points are in channel 1, three are in channel 6 and two are in channel 11.

Now, lets review the interference measured in each of the testing scenarios. To compute the interference at a node ap_i we first define $I_l(ap_i, ap_j)$ as follow: if ap_i and ap_j are in the same channel then the quality reported by *iwlist* is used as the value. Note that in the $I_l(ap_i, ap_j)$ the node ap_i is one of the access point in Table 5.3 and the node ap_j can be one of the access point in Table 5.3 or one of the access point in Table 5.4. As you can see there are nodes in Table 5.4 that are not in channels $\{1, 6, 11\}$. In this case we multiply the quality by a factor: 0.8 if channels are adjacent (1 and 2 for example), 0.6 if channels are two channels distance (1 and 3 for example) and so on. If ap_i and ap_j are in non overlapping channels then the value is 0.

Table 5.7: Centralized Result

Ip	Mac Address	Channel
13.13.13.1	68:7F:74:26:3C:36	11
13.13.13.110	68:7F:74:26:3F:DE	11
13.13.13.111	68:7F:74:26:3E:E2	6
13.13.13.112	68:7F:74:26:39:DB	1
13.13.13.113	68:7F:74:26:3F:BD	1
13.13.13.114	68:7F:74:26:39:D5	6
13.13.13.115	68:7F:74:26:3B:04	1
13.13.13.116	C0:C1:C0:01:25:B3	1
13.13.13.117	C0:C1:C0:01:27:C9	6
13.13.13.118	C0:C1:C0:01:26:D6	1

Then for each node we calculate: $I_l(ap_i) = \sum_{\forall ap_j \in N(ap_i)} I_l(ap_i, ap_j)$

where $N(ap_i)$ are the list of nodes that ap_i can see. Finally we can calculate the $L_{ran}(G, C)$ value as the sum of all $I_l(ap_i)$.

When each deployment start we have that:

$L_{ran}(G, C) = 35,65429$

The results obtained after run the testbeds are shown in Table 5.8

Table 5.8: TestBed Results

Strategy	Result
Centralized	12,91714286
Coordinated	11,92571429
UnCoordinated	11,19714286

We can observe a decrease of measured value from around 35 to 11 or 12. All implementations behave similar with a better performance of the uncoordinated deployment.

Now lets see each scenario in more detail. In Figure 5.1 we can see the results of the uncoordinated approach. In the horizontal axis are the ip (to distinguish each) of the access points of the lab. The green bar is the interference level at the beginning, the blue bar is the interference level at the end of the run. The black X is the average of the interference at the beginning and the red square is the average of the interference level at the end. When the run ends the node 111 has the max value: $1,905714286$ and the node 116 has the min value: $0,28$, with an average value equal to $1,119714286$. The difference between max value and the average, and the min value and the average is around 0.80.

In Figure 5.2 we can see the results of the coordinated approach. In this case when the run ends the node 110 has the max value: $2,54$ and the node 116 has the min value: 0, with an average value equal to $1,192571429$. The difference between max value and the average, and the min value and the average is around 1.20.

Finally in Figure 5.3 we can see the results of the centralized approach. In

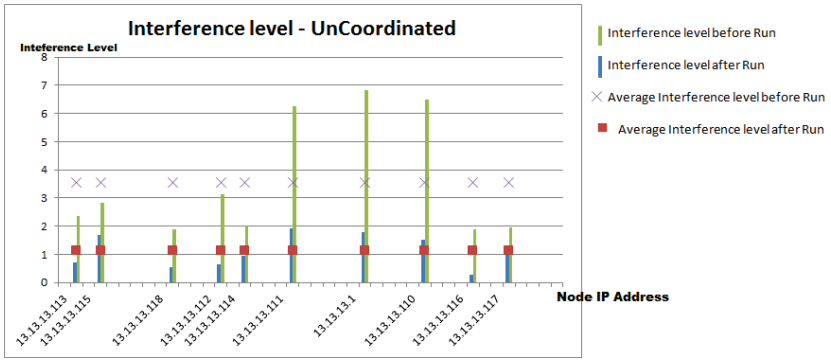

Figure 5.1: TestBed Graph for the Un-Coordinated Solution.

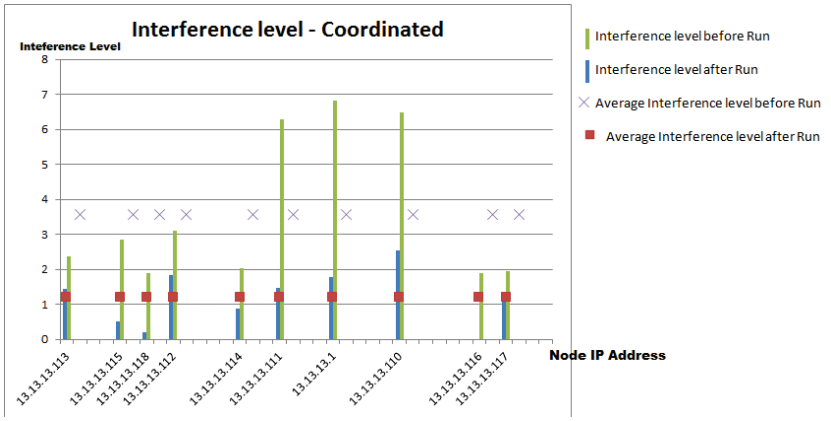

Figure 5.2: TestBed Graph for the Coordinated Solution.

this case when the run ends the node 111 has the max value: $2,591428571$ and the node 1 has the min value: $0,334285714$, with an average value equal to $1,291714286$. The difference between max value and the average, and the min value and the average is around 1.10.

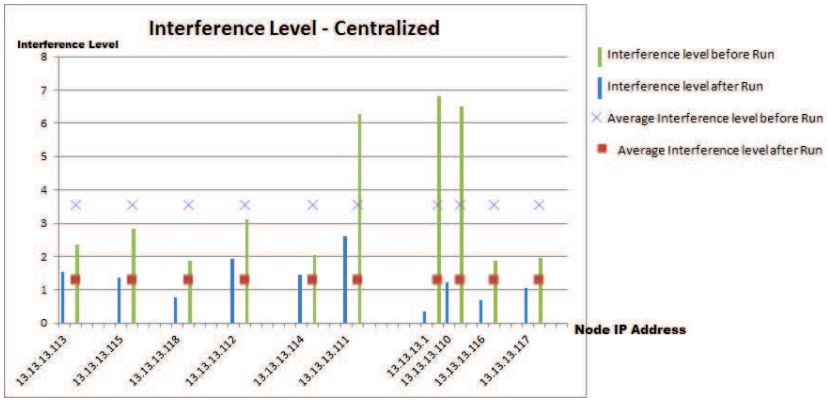

Figure 5.3: TestBed Graph for the Centralized Solution.

TestBed results shows that the uncoordinated solution outperform the others solutions in the L_{ran} value. In the case of the centralized implementation, although it has the greater value for the L_{ran} compared with the coordinated implementation, the interference level is better distributed between the nodes. Intuitively, we should expect that the centralized solution must produce the lower value of the L_{ran} because the central node has the global view of the network. Then, a natural quetion is, why the uncoordinated solution produce better results? Our gess is that the problem is the output produced by the *iwlist* Linux command. The implementation of the *iwlist* is driver specific as we explain later in section A.2.4. Two consecutive runs of the

```
iwlist wlan0 scan
```

(used in the *get_cells_in_range* utility in the testbed) can produce different outputs.

For example we did two consecutive executions of the

```
iwlist wlan0 scan
```

command in a Terminal window and we get for the first run,

```
wlan0     Scan completed :
          Cell 01 - Address: 00:25:12:6D:17:3D
                    Channel:5
                    Frequency:2.432 GHz (Channel 5)
                    Quality=37/70  Signal level=-73 dBm
                    Encryption key:on
                    ESSID:"TuleWiFi"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
                    Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s
```

```
                    36 Mb/s; 48 Mb/s; 54 Mb/s
          Mode:Master
          ......
Cell 02 - Address: 54:E6:FC:A1:B2:E4
          Channel:1
          Frequency:2.412 GHz (Channel 1)
          Quality=34/70  Signal level=-76 dBm
          Encryption key:on
          ESSID:"ER"
          Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                    12 Mb/s; 24 Mb/s; 36 Mb/s
          Bit Rates:9 Mb/s; 18 Mb/s; 48 Mb/s; 54 Mb/s
          Mode:Master
          ........
Cell 03 - Address: C8:6C:87:A0:4A:41
          Channel:5
          Frequency:2.432 GHz (Channel 5)
          Quality=30/70  Signal level=-80 dBm
          Encryption key:on
          ESSID:"tatocine"
          Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s
                    11 Mb/s; 12 Mb/s; 18 Mb/s
          Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
          Mode:Master
          .......
Cell 04 - Address: 98:FC:11:BD:66:B4
          Channel:9
          Frequency:2.452 GHz (Channel 9)
          Quality=31/70  Signal level=-79 dBm
          Encryption key:on
          ESSID:"Gaby"
          Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                    9 Mb/s; 12 Mb/s; 18 Mb/s
          Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
          Mode:Master
          .........
```

and for the second run,

```
wlan0     Scan completed :
Cell 01 - Address: 00:25:12:6D:17:3D
          Channel:5
          Frequency:2.432 GHz (Channel 5)
          Quality=37/70  Signal level=-73 dBm
          Encryption key:on
          ESSID:"TuleWiFi"
          Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
          Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s
                    36 Mb/s; 48 Mb/s; 54 Mb/s
          Mode:Master
```

```
          .........
Cell 02 - Address: 00:23:54:DB:0A:10
          Channel:1
          Frequency:2.412 GHz (Channel 1)
          Quality=36/70  Signal level=-74 dBm
          Encryption key:on
          ESSID:"wifigonza"
          Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                    9 Mb/s; 12 Mb/s; 18 Mb/s
          Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
          Mode:Master
          .......
Cell 03 - Address: 54:E6:FC:A1:B2:E4
          Channel:1
          Frequency:2.412 GHz (Channel 1)
          Quality=36/70  Signal level=-74 dBm
          Encryption key:on
          ESSID:"ER"
          Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                    12 Mb/s; 24 Mb/s; 36 Mb/s
          Bit Rates:9 Mb/s; 18 Mb/s; 48 Mb/s; 54 Mb/s
          Mode:Master
          ........
```

Note that for the second run we get only three cells while for the first run we get four cells. Moreover, in the second run cell number 02 is *wifigonza* and this AP does not appear in the first run, on the other hand cells number 03 and 04 of the first run do not appear in the second run.

We believe that this intermittence is the cause that the coordinated approach has value lower than the others two implementations. But the results are very similar in the three cases. So we will do more case studies during simulation phase.

Finally, before we end the section related to the testbed we want to show other graphics. This is to show how the interference affects throughput. To do this we use MGEN [10] to generate network traffic and to log it. We put all access points on channel 6 and connect two notebooks using wireless interface to two different access points. In the two access points that we use we start the following mgen script:

```
mgen event "listen tcp 5000" output log.drc
```

this is to start listen on port 5000 for tcp connections. The *log.drc* is just a log file.

Then, in the notebooks already connected to each AP we use the following mgen script to send tcp traffic to the access points:

```
0.0 ON 1 TCP DST 13.13.13.110/5000 PERIODIC [-1 1024]
10.0 OFF 1
```

where 13.13.13.110 is the ip of the access point. This command send 1024 byte packets as fast as possible using tcp for ten seconds.

After that we run one of our implementations, so the access points are assigned new channels. In particular the access points we use to connect the notebooks changes to channel 1 and channel 11. Then we run the same scripts mentioned above.

We use TRPR[14] trace analyzer to analyze output produced by MGEN and creates output suitable for plotting:

```
trpr mgen input <mgenLogFile> auto X output <plotFile>
```

the $< mgenLogFile >$ is substituted by *log.drc* the output of the MGEN and $< plotFile >$ is an output file. We use gnuplot[5] to display a graph of TRPR analysis results.

The results of this experiment are as follow: In Figure 5.4 you can see the throughput of the first notebook using the shared channel and in Figure 5.5 you can see the throughput of the first notebook using the non shared channel.

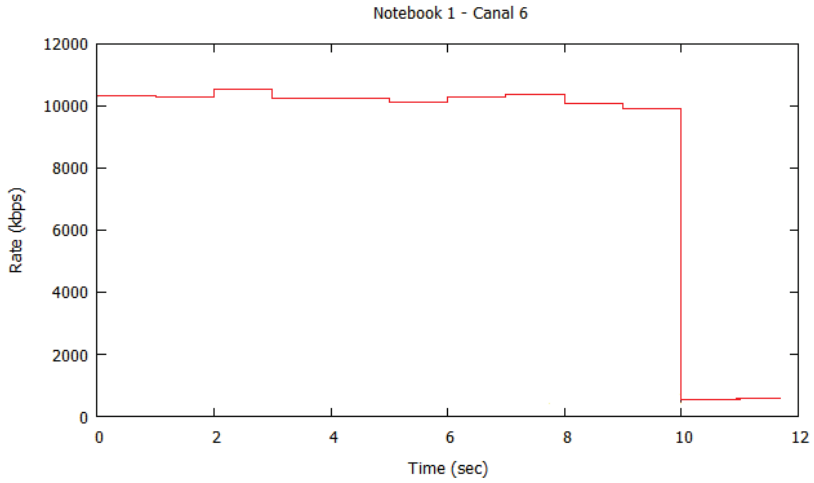

Figure 5.4: Throughput Notebook 1 on a Shared Channel.

While you can see a similar result on the throughput of notebook 1 in both situation, note that for notebook 2 (figures 5.6 and 5.7) there is a poor throughput in the case of a shared channel. This suggest that is important to have a good channel assignment strategy.

5.5 Motivation for Scalability Evaluation

After having proved the feasibility of a practical deploy in a laboratory and getting the first measures of each implementation we wish to measure the behavior in a network with more quantity of nodes. We can assume that the local and uncoordinated implementation will scale because that solution will operate in a node without sending neither receiving messages from/to other nodes. We just need to install the local and uncoordinated solution in each node of the network and run it. For the oscillation issues observed we found that we could

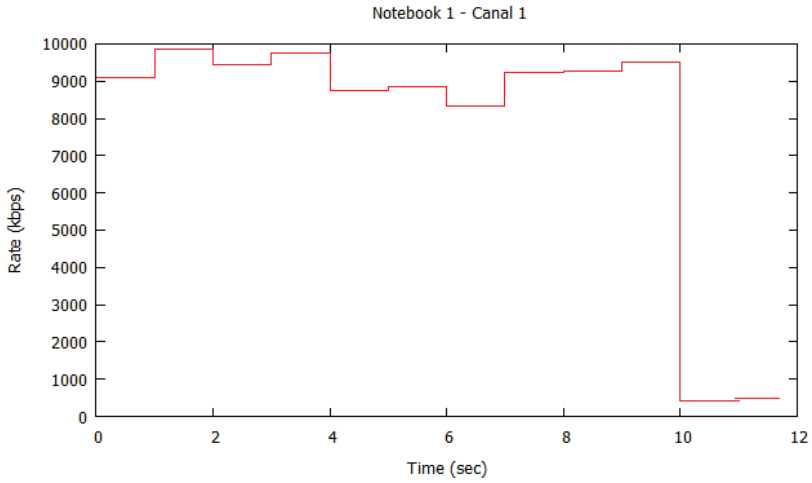

Figure 5.5: Throughput Notebook 1 on a Non-Shared Channel.

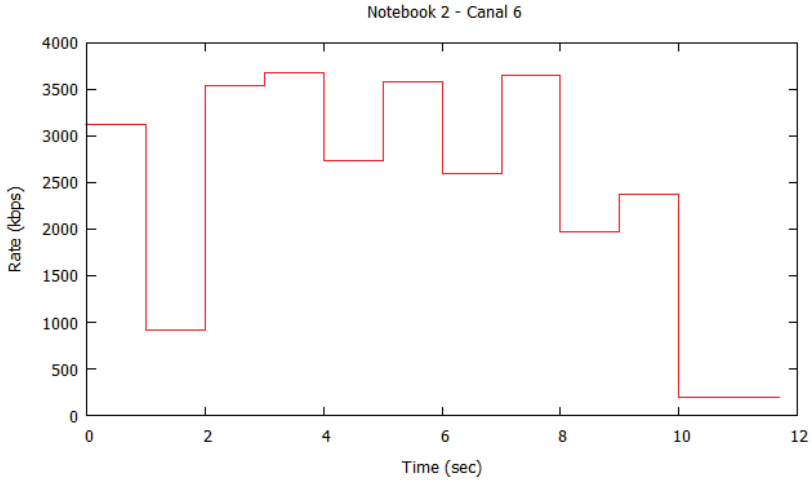

Figure 5.6: Throughput Notebook 2 on a Shared Channel.

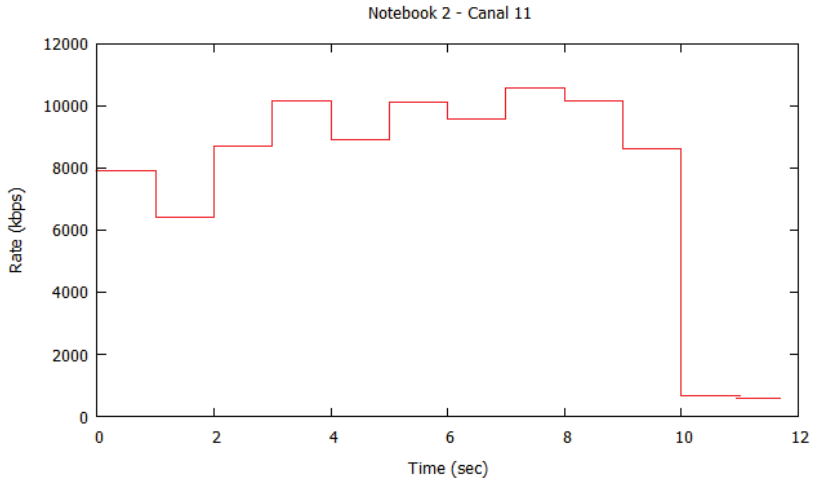

Figure 5.7: Throughput Notebook 2 on a Non-Shared Channel.

take control of it with a feedback control technique as demonstrated in Chapter 4. So, one of the goal of the simulation is to measure scalability issues for the coordinated and centralized solutions because both of them exchange messages between nodes. As we mention before one of the assumption of our work is that we are working with tens of nodes (not hundreds or thousands), this mean we are thinking in a single building, so we experiment in a simulation with 20 and 30 nodes. Also, we added some neighbors node that are not running our solution but that have a channel already assigned and interferes with our nodes.

Another goal of the simulation is to take measures and evaluate the interference level in each of the implementations. We will observe how channels are assigned between nodes in a network with tens of nodes and how this channels distribution impact on interference level. We will be calculating L_{ran} in all simulation scenarios and we will compare its value.

The remain of the chapter is as follow. The next section present the simulator we used and how we embed the RAN System into it. Section 5.7 details the environment under which the simulation run. Finally, we show simulation results.

5.6 Simulator

For simulation we use DisJ simulator [2]. The DisJ is an Eclipse plug-in.

In DisJ simulator you can visually set up a network graph as shown in Figure 5.8 (DisJ make use of Graphical Editor Framework [4] in order to use graphical editor).

Then you attach a protocol to the graph and run the simulation. We added a Wake-Up protocol to start the simulation. This protocol can be described as follow: "very often, in a distributed environment, we are faced with the following situation: A task must be performed in which all the entities must be involved; however, only some of them are independently active (because of

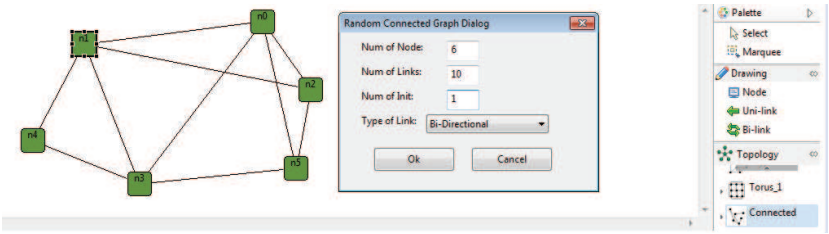

Figure 5.8: Defining a Graph Topology in DisJ.

a spontaneous event, or having finished a previous computation) and ready to compute, the others are inactive, not even aware of the computation that must take place. In these situations, to perform the task, we must ensure that all the entities become active. Clearly, this preliminary step can only be started by the entities that are active already; however, they do not know which other entities (if any) are already active. This problem is called Wake-up (Wake-Up): An active entity is usually called *awake*, an inactive (still) one is called *asleep*; the task is to wake all entities up; see Figure 5.9"[41].

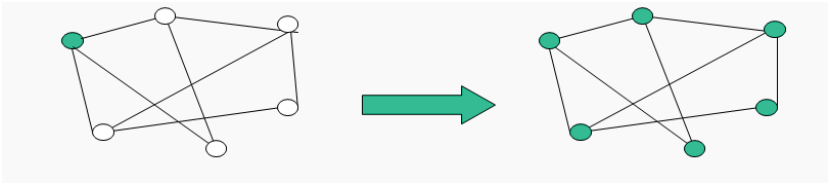

Figure 5.9: Wake Up with one Initiator. Source [1].

In our implementation of the protocol there is only one entity that has the information, so the computation will be started by this entity. The protocol is outlined as follow:

1. Initiator node when the simulation start → start rnr, start lupa, send(wake up) to N(x), become done
2. Any node except the initiator receiving wake up → start rnr, start lupa, become done, send(wake up) to N(x)
3. Initiator node receiving wake up → do nothing
4. Any node in done state → do nothing

where N(x) means the neighbors of x. Note that all nodes are wake up and during wake up it start rnr and lupa like if it is a access point. The same rnr and lupa implementation that run inside the access points run as process in the computer running the simulation. The only things that change is in the configuration that point to the fourth utilities.

5.7 Simulation Environment

The simulation was run using a notebook with the characteristic depicted in Figure 5.10. Note that the operating system we use to run the simulation is Windows 7 Enterprise with Service Pack 1 installed.

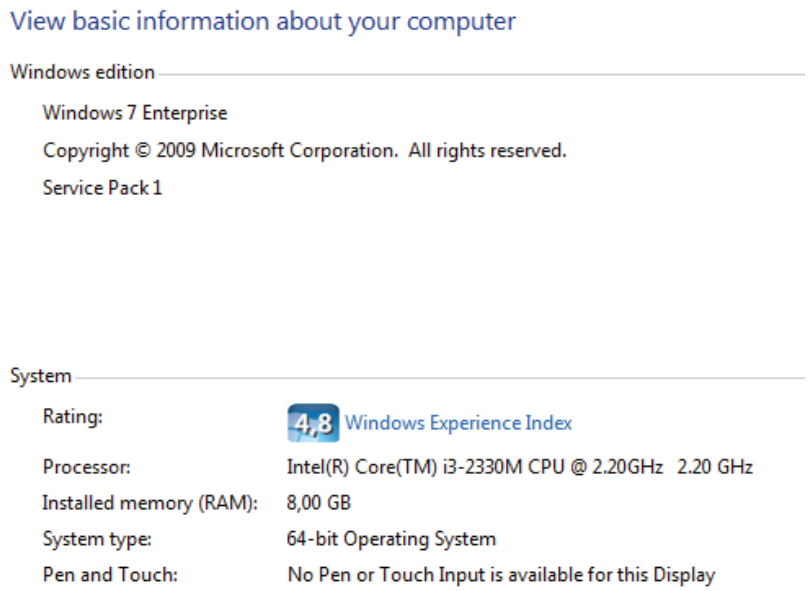

Figure 5.10: Notebook Used to Run the Simulations.

Eclipse is an open platform for tool integration built by an open community of tool providers. Operating under a open source paradigm and it can be downloaded at *www.eclipse.org*. Eclipse version is Juno and the java version is 7 update 25. As we mention before in order to use Graphic Editor in Eclipse^{TM}we have to install Graphical Editor Framework (GEF) library. GEF is an open source project under support of Eclipse Foundation. After Eclipse is installed, we need to download DisJ and add DisJ the installed Eclipse plugins.

5.8 Simulation Results

We could observe in section 5.4 that the uncoordinated implementation behaves very well. That solution will scale because each AP will run its own RAN System without interacting with others RAN. We also showed in chapter 4 a mechanism that help us handle the inherent oscillatory problem of that solution. This way the focus of the simulations are in the centralized and coordinated strategies. We tested both implementations in four different scenarios:

1. A graph with 20 nodes with all nodes running our algorithms (all nodes are managed by us).
2. A graph with 25 nodes with only 20 nodes running our algorithms (managed by us) and 5 nodes in the neighborhood.

3. A graph with 30 nodes with all nodes running our algorithms (all nodes are managed by us).

4. A graph with 40 nodes with only 40 nodes running our algorithms (managed by us) and 10 nodes in the neighborhood.

One of the way to define a graph in the DisJ Graph Editor is to use a topology library. This provides a ready-made set of topologies that can be edited such as Ring, Tree, Complete Graph, and more. We used the *connected* topology library where we can set the number of nodes and the number of links (see Figure 5.8.

The meaning of the defined graph is as follow:

Nodes Each node in the graph represent an AP.

Links A link between two nodes n_1 and n_2 indicates that n_2 signal is perceived at node n_1. This mean that if n_1 and n_2 are on the same or adjacent channel then they will interfere each other.

Distance between two connected nodes The distance between two nodes represents the quality of the signal perceived at each end. Each node in DisJ has a position represented by the x and y coordinates. This information let us calculate the distance between a pair of nodes. Then, we will represent the signal quality as a function of the distance between node n_1 and node n_2. In Appendix A.2 we will show the details. Note that we will consider only those nodes that can interfere which is the same to consider those nodes that are connected by a link.

State of a node Each node in DisJ is of type *distributed.plugin.runtime.engine.Entity*. We must define a set of States for Entity in any distributed protocol. DisJ library has provided standard programming interfaces for user to deal with states. For instance, *getState()* is for checking a current state of an entity, and *become()* is for setting a state of an entity to be at a given state. In our implementation, the state of a node is equivalent to the channel in which the AP is operating. With the *getState()* operation we can query the access point in which channel it is operating and with the *become()* operation we can implement the change of the channel (refer to Appendix A.2 for more details).

Color of a node In order to see the state change of a node from the simulator, we can specify a color for each state that defined in the protocol. So in the figures that follow we set a color for each state of a node: green for channel 1, yellow for channel 6 and red for channel 11.

5.8.1 Graph with 20 Nodes

In Figure 5.11 you can see the state of the graph after the centralized run completes and in Figure 5.12 you can see the state of the graph after the coordinated run completes.

In Table 5.9 you can see the channel assigned to each AP in the runs which do not have neighbours. In the left column are the node names, then there is the channel assigned in the centralized simulation for 20 nodes, the next one is the assignment results for the centralized simulation with 30 nodes. The other two columns are the same but for coordinated simulation.

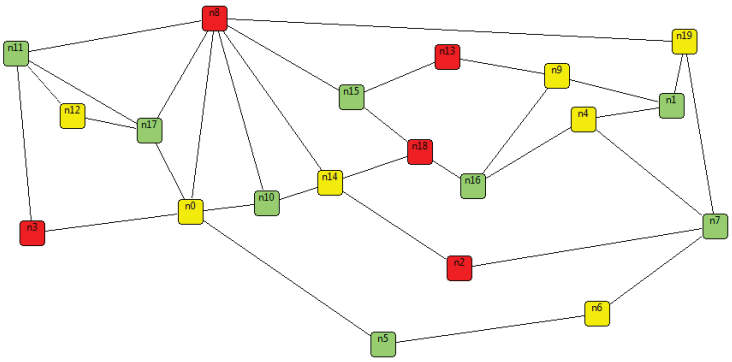

Figure 5.11: Simulation Results with 20 Nodes - Centralized.

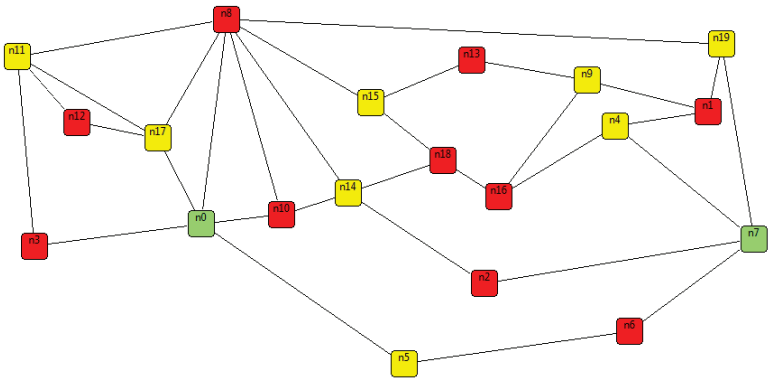

Figure 5.12: Simulation Results with 20 Nodes - Coordinated.

Table 5.9: Results 20 and 30 nodes without neighbours

Node	Cent	Cent	Coord	Coord
n0	6	11	1	1
n1	1	1	11	6
n2	11	1	11	1
n3	11	1	11	11
n4	6	1	6	11
n5	1	11	6	6
n6	6	1	11	11
n7	1	11	1	6
n8	11	1	11	11
n9	6	6	6	11
n10	1	1	11	11
n11	1	6	6	11
n12	6	1	11	6
n13	11	6	11	6
n14	6	1	6	11
n15	1	1	6	11
n16	1	6	11	6
n17	1	1	6	11
n18	11	6	11	11
n19	6	11	6	6
n20		6		11
n21		11		11
n22		1		11
n23		11		6
n24		6		11
n25		6		6
n26		6		11
n27		11		1
n28		11		11
n29		1		11

From the information in Table 5.9 we have that eight nodes are in channel 1, seven nodes are in channel 6 and five nodes are in channel 11. From the information in Table 5.10 we have that ten nodes are in channel 11, eight nodes are in channel 6 and two nodes are in channel 1.

The most important result is that **nodes (APs) in the centralized solution do not have interference**. For example, lets look at node $n0$. $n0$ was assigned channel 6. The neighbors of $n0$ are: node $n10$ with channel 1, node $n5$ with channel 1, node $n3$ with channel 11, node $n17$ with channel 1 and node $n8$ with channel 11. As another example look at node $n15$. $n15$ was assigned channel 1. The neighbors of $n15$ are: node $n18$ with channel 11, node $n8$ with channel 11 and node $n13$ with channel 11.

Over the entire graph there are not neighbours sharing the same color.

Now, consider the coordinated solution. There are six nodes (APs) with the same color as at least one neighbor (in the same channel). The nodes are $n10$, $n11$, $n8$, $n18$, $n17$ and $n16$. For example, lets look at node $n10$ that was assigned channel 11. One of its neighbors is node $n8$ that is in channel 11 too. As another example consider node $n17$ that was assigned channel 6. Its neighbor node $n11$ is in channel 6 too. The interference level of the entire system is $4,77$ with an average value of $0,238$ per node.

5.8.2 Graph with 30 Nodes

In Figure 5.13 you can see the state of the graph after the centralized run completes and in Figure 5.14 you can see the state of the graph after the coordinated run completes.

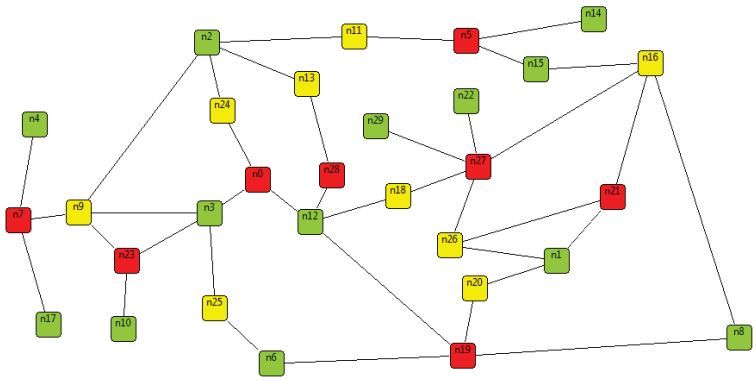

Figure 5.13: Simulation Results with 30 Nodes - Centralized.

Here again we can see that **nodes (APs) in the centralized solution does not have interference**. While in the coordinated solution we have six APs that present interference. The total interference level in the system is $2,1380952381$.

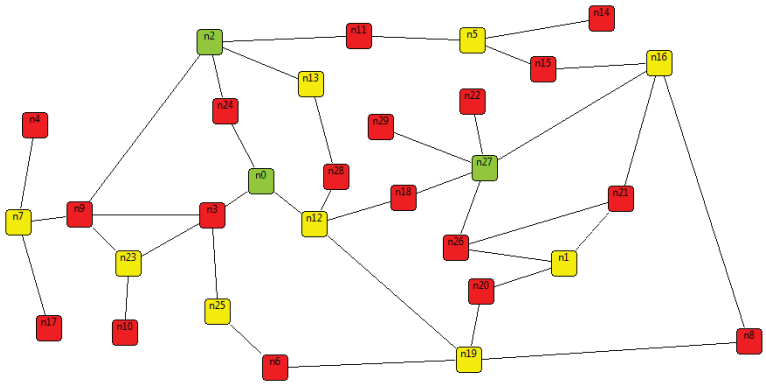

Figure 5.14: Simulation Results with 30 Nodes - Coordinated.

5.8.3 Graph with 20 Nodes plus 5

In Figure 5.15 you can see the state of the graph after the centralized run completes and in Figure 5.16 you can see the state of the graph after the coordinated run completes.

In this graph nodes name that start with the letter m are the neighbors node (nodes not running RAN System). These nodes are: $m20$, $m21$, $m22$, $m23$ and $m24$. To these nodes we have assigned a channel by hand (we set the state of the node hard coded in DisJ simulator). $m20$ and $m23$ have channel 11, $m21$ and $m24$ have channel 1 and $m23$ channel 6. For these nodes the channels are fixed, but they produce interference.

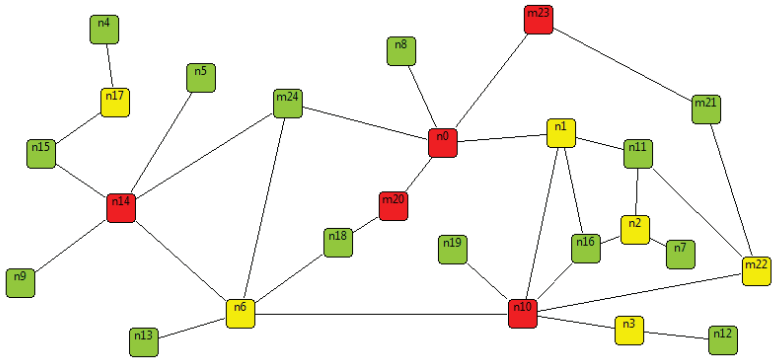

Figure 5.15: Simulation Results with 20 nodes plus 5 neighbors - Centralized.

In Table 5.10 we list the channel assignment after simulation run when we add neighbor nodes to the graph, ie nodes that are not running RAN System. One thing to note here is that the coordinated solution does not assign channel 1 to any node. In this simulation, centralized results get an interference value greater than zero. But if you look carefully in the Table 5.10 and graph in

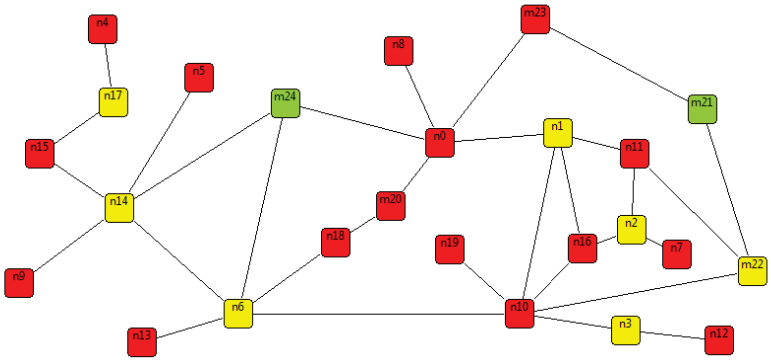

Figure 5.16: Simulation Results with 20 Nodes plus 5 Neighbors - Coordinated.

the Figure 5.15 you can see that only one node (node $n0$) has interference. In Table 5.10, considering the second column for $n0$ you can see that the channel assigned is 11.

The neighbors of node $n0$ are $n1$ in channel 6, $n8$ in channel 1. But also has three "external" neighbor nodes. Nodes $m20$ in channel 11, $m23$ in channel 11 and $m24$ in channel 1. Is the interference produced by the nodes $m20$ and $m23$ that cause the system to have an interference greater than zero, measured in node $n0$. All others node in our domain, does not have interference.

While the centralized approach has a good result as expected, because it has the knowledge of the entire network to run the backtracking algorithm, the coordinated approach has demonstrated some problems. As we mention before, if you look at fourth column in Table 5.10 you can see that the result only assign channels 6 and 11 in the coordinated solution. This leads that seven nodes present interference producing a total interference value equal to $6,3571428571$. The nodes with interference are: $n10$, $n0$, $n8$, $n6$, $n19$, $n14$ and $n16$. Looking at node $n10$ we can see that the channel assigned is 11. Their neighbors are node $n1$ in channel 6, node $m22$ in channel 6, $n3$ in channel 6, $n19$ in channel 11, $n6$ in channel 6 and $n16$ in channel 11. This mean that node $n10$ has interference from nodes $n19$ and $n16$. And the other way is also true. $n19$ has interference from $n10$ although with different quality and $n16$ has interference from $n10$ too.

Lets review our final simulation.

5.8.4 Graph with 30 Nodes plus 10

In Figure 5.17 you can see the state of the graph after the centralized run completes and in Figure 5.18 you can see the state of the graph after the coordinated run completes.

In this simulation we have similar results to previous one. In the centralized approach the total measured interference level is $3,9857142857$. A very low level considering the amount of nodes in the environment (forty nodes in total). In the third column in Table 5.10 you can see the channel assignment as a result of the centralized simulation. In thirty nodes managed by us, only four nodes

Table 5.10: Results 20 and 30 nodes with neighbors

Node	Cent	Cent	Coord	Coord
n0	11	6	11	11
n1	6	1	6	11
n2	6	1	6	11
n3	6	1	6	11
n4	1	11	11	6
n5	1	6	11	6
n6	6	6	6	6
n7	1	6	11	6
n8	1	1	11	11
n9	1	11	11	6
n10	11	6	11	11
n11	1	1	11	11
n12	1	1	11	11
n13	1	1	11	11
n14	11	6	6	6
n15	1	6	11	11
n16	1	1	11	11
n17	6	1	6	11
n18	1	11	11	6
n19	1	1	11	11
n20		6		6
n21		1		11
n22		6		6
n23		11		11
n24		1		11
n25		6		6
n26		1		11
n27		1		11
n28		1		11
n29		1		11

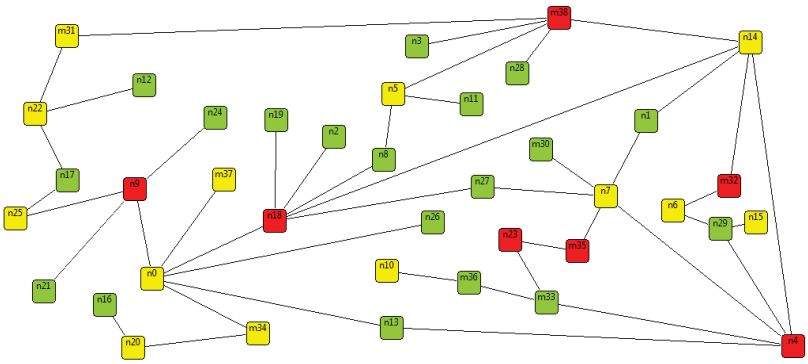

Figure 5.17: Simulation Results with 30 Nodes plus 10 Neighbors- Centralized.

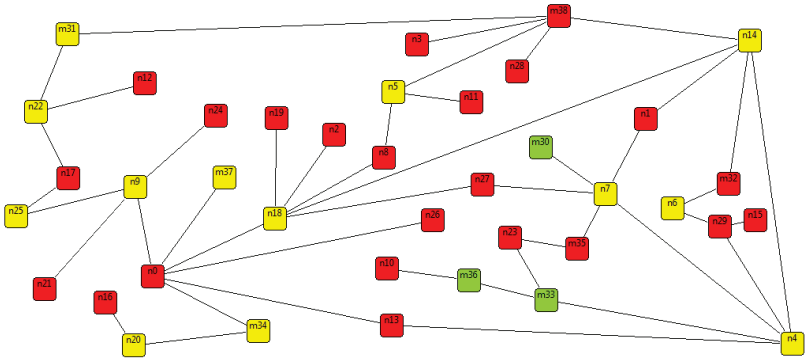

Figure 5.18: Simulation Results with 30 Nodes plus 10 Neighbors- Coordinated.

result with interference: $n23$, $n22$, $n0$ and $n20$.

In contrast, the coordinated solution present bad results. The total measured interference level in this case is $11,1714285714$. A much higher value than $3,9857142857$. Again, if you look at the fifth column in Table 5.10 you can see that the algorithm only assign channels 6 and 11 in the coordinated solution. This yields that more than eleven nodes have interference.

In the following section we present a chart comparing the behavior of the coordinated solution versus centralized solution in the simulations.

5.8.5 Summary of Simulation Results

In this section we present a summary of the fourth simulation scenarios comparing the results of the coordinated implementation against the centralized implementation.

In chart 5.19 we present an average interference level in each node. The vertical axis is the measured interference level.

The horizontal axis present four points, each one representing a simulation scenario: the simulation with twenty nodes (section 5.8.1), the simulation with twenty nodes and five "external" nodes (section 5.8.3), the simulation with thirty nodes (section 5.8.2) and the simulation with thirty nodes and ten "external" nodes (section 5.8.4).

In each point of the horizontal axis there are two bars. The blue color corresponds to the results in the centralized simulation and the red color corresponds to the results in coordinated simulation.

Clearly the centralized solution outperform the coordinated solution. Moreover, in chart 5.20 where we present the total interference level of the entire network there is a clear gain if we use the centralized approach.

We can conclude that if we have the opportunity to connect the network is worth to attach a computer and run the centralized implementation that outperform the coordinated simulation. If we cannot attach a computer to the network then the coordinated solution can be deployed with the knowledge that it not produce optimal results. But it decrease the interference level with respect to the initial.

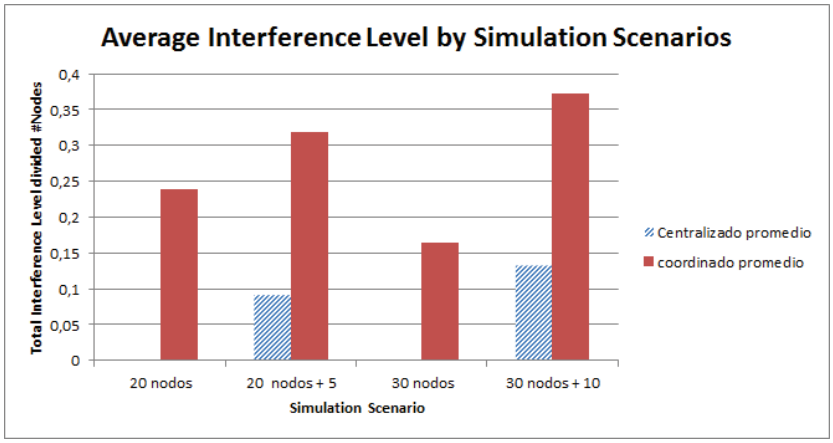

Figure 5.19: Average Interference Level.

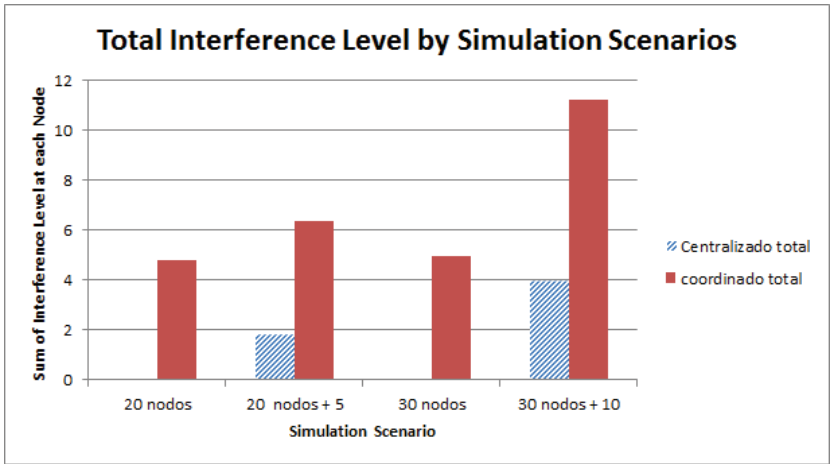

Figure 5.20: Total Interference Level.

5.9 Scalability conclussions

Usually, in distributed computation two types of measure are used: the *amount of communication activities* and the *time* required by the execution of a computation. They can be seen as measuring costs from the system point of view (how much traffic will this computation generate and how busy will the system be?) and from the user point of view (how long will it take before I get the results of the computation?).

To measure the amount of communication activities, the most common function used is the number of message transmissions **M**, also called message cost. So in general, given a protocol, we will measure its communication costs in terms of the number of transmitted messages.

The other important measure of efficiency and complexity is the total execution delay, that is, the delay between the time the first entity starts the execution of a computation and the time the last entity terminates its execution. Note that time is here intended as the one measured by an observer external to the system and will also be called real or physical time. In general, there is no assumption about time except that communication delays for a single message are finite in absence of failure. In other words, communication delays are in general unpredictable. For this reason, we only say here that the simulations takes less than five minutes to produce results. Hence, we will calculate the communication costs of the algorithms by determining the number of *message transmissions*.

First of all, let us determine the number of message transmissions for the coordinated strategy. In this calculation we will omit the *trap* notification messages send by the monitoring agent, rmoon, to its own pdp because it is internal to a node, and will not influence the cost.

In this solution, each service (pdp, pep and rmoon) running in each of the Access Points send a suscribe message to receive messages routed to them. So, we have: $3*|V|$ where $|V|$ is the number of Access Points running the algorithms.

Looking at the leader node we can observe that, first it send a broadcast message, and then for each node the leader send a notification message requesting to assign a channel. There is also a notification message for his own pep. Then, we have: $1 + |V|$, where 1 is for the broadcast message.

Now, considering the rest of the Access Points, we can observe that each node respond to the broadcast message ($|V| - 1$) with the level of interference measured by the node and then, each node reply to the assign channel message with an *OK* notification message ($|V| - 1$). We can consider the notification message from each pdp to it's own pep instructing to change channel: $|V| - 1$

So, the total number of messages transmitted in the coordinated solution is:

$$M[Coordinated] = 3 * |V| + 1 + |V| + 3 * (|V| - 1) = 7 * |V| - 2$$

Hence the total number of messages transmitted in the coordinated solution is $O(|V|)$.

Then, lets examine the total number of messages transmitted in the centralized solution. In this solution, the central node broadcast a message to all Rmoon services in the network. Then, all Rmoon services send the response back to the central node. Thus, we have $1 + |V|$, where 1 is for the broadcast message.

Finally, after computing the best channel assignment for each node, the central send to each pep in every Access Points the channel that the node must operate. For this last communication, the number of messages is $|V|$

So, the total number of messages transmitted in the centralized solution is:

$$M[Centralized] = 1 + |V| + |V| = 2 * |V| - 1$$

In this case, the total number of messages transmitted in the centralized solution is $O(|V|)$.

The difference between the two approaches is in the coefficient of $|V|$. In conclusion, the cost of both algorithms have the same order, and are determined by the number of Access Points.

Chapter 6

Conclusions and Future Work

This chapter presents the main conclusions of this thesis and proposes guidelines for future works.

6.1 Conclusions

The performance of dense wireless networks depends on the channel assignments between neighboring access points (APs). The limited number of non-overlapping channels may lead to severe interference scenarios if no appropriated spectrum planning is employed. The IEEE 802.11b/g standards provide only a limited number of non-overlapping channels, creating the need to design a channel assignment strategy to re-use these channels. Additionally, in dense urban areas it is usual to find wireless networks scenarios with interfering APs belonging to different administrative domains. Moreover, in many practical scenarios, like residential deployments in dense urban areas, the interfering WLANs belong to different administrative domains

We did an in depth review of the state of the art in chapter 2 where we showed that this topic is important not only from the academic point of view but also is important to the industry. Main companies in the field, like Cisco, Aruba and Meru have guidelines of how to setup a high density wireless network. Then we presented a detailed view of the academic research where there are a variety of research lines. A channel assignment strategy can be classified as static or dynamic. The dynamic strategy assumes that the network interface can switch the communication channel using a coordinated or uncoordinated approach. From a different perspective the strategies could also be classified as a centralized implementation with one node having the knowledge of the entire network or as a distributed solution.

We formalized the problem deriving an objective function:

$$\begin{aligned} &Minimize \\ &L_{ran}(G,C) = \sum_{\forall e=(ap_i,ap_j)\in E \wedge ap_i \in V'} I_l(ap_i, ap_j) \end{aligned}$$

where V' is a set of AP belonging to our administrative domain and I_l is the interference that ap_i perceived the signal emitted from ap_j.

Afterwards, we presented three different solutions to the problem of channel assignment. We can make a correspondence between our proposed strategies and the classification we did in the state of the art study:

- RAN System is a rule based system similar to the one on section 2.2.6.
- based on what we found in the literature as *Least Congested Channel* we developed an uncoordinated solution, similar to the 2.2.6.
- coordinated solution with a philosophy similar to the 2.2.7.
- centralized solution that is in the same direction as those presented in 2.2.3.

We did a testbed deployment of the three strategies with ten access points and we obtained similar results. We showed in table 5.8, in section 5.4 the calculated interference level in each implementation. With the uncoordinated solution performing a little better than the others. The value of the L_{ran} for the uncoordinated solution was below the values for the coordinated and centralized solution. But the difference between results in the three strategies are not significant. We think that this could happen because of the intermittence in the output produced by the *iwlist* command.

During the testbed of the uncoordinated implementation we observed that this solution present oscillatory issues. For this reason in chapter 4 we presented a feedback control system applied to the uncoordinated strategy. This way we could demonstrate that is viable to control the oscillatory issues in autonomic manner. We need to set a desired values for the amount of channel changes per unit of time (for example one hour) and the system will adapt a *threshold* value to meet the objective. We also showed testbed results to evaluate the proposal getting promising results that must be verified with more scenarios to get more statistics and thus get more robust conclusions.

Then we did a thorough review of the coordinated and centralized strategies using DisJ simulator. We have found that the centralized solution outperform coordinated in all cases.

We can summarize a comparison between the strategies as follows:

- UnCoordinated.
    1. There is no need of a communication infrastructure between Access Points.
    2. Good results in testbed, but present oscillatory issues.
    3. The oscillatory issues can be controlled by using a feedback control technique.
    4. Is scalable, because each node run in isolated mode.
    5. No point of failure. If one node fail, that node will keep operating in the same channel without affecting the rest.
- Coordinated

1. Needs a communication infrastructure between Access Points.
2. Produce results not so good as the centralized solution.
3. Relies on leader node that can be determined dynamically.
4. One point of failure. But, although not implemented, we can recover by recalculating the leader.

- Centralized

1. Needs a communication infrastructure between Access Points.
2. Produce best result because of the knowledge of the composition of the network by the central node.
3. Relies on a central node that must be setup beforehand.
4. One point of failure.

So the conclusion is that if there is a chance to connect the APs then the best solution is to use the centralized implementation. It has good results and does not present oscillatory problems. If APs cannot be connected then the uncoordinated is a good alternative because we could demonstrate that the oscillatory issues can be controlled.

6.2 Future Work

Although the coordinated solution prove to be a valid alternative further inquiries and studies are needed to improve and attain better results. We suggest that new research pursue new veins of study such as the adjustment in the algorithm in order to achieve lower values that minimize the function L_{ran} results. One thing that must be added in the protocol is that when the leader instruct a node to assign a channel then the leader must send the list of APs (their mac address) that already have the channel assigned. In the current implementation a node when instructed by the leader scan the environment an select the channel with less interference, but if the leader sends to the access point the list of nodes that already have been switched their channels then the device can select the channel with more intelligence. This way we believe we can increase the performance of the coordinated solution.

Another issue to improve that is related to the coordinated implementation is the leader election. As we mention in section 3.4.1 leader in RAN System is done manually during set up. We could implement one of the algorithm described in the book [41]. If -as the research group- we will be using RAN system for other purpose this can be a good enhancement to the system.

In future research it is also necessary to tune our testbed implementation, considering the intermittence of the output of the *iwlist* command. It might be useful to do a cache of the output for some period of time. If one AP appears listed in the output we can renew the time to live in the cache, but if the amount of time elapsed and the AP that is already cache does not appear again, then we can remove it from the cache. With this we are sure that the centralized approach will outperform the uncoordinated solution.

At the end of the chapter 4 we showed promising results of the feedback control technique applied to our un-coordinated solution to bound the oscillatory

issues. We did this in a small and unique testbed environment. We believe that to get more accurate results more scenarios must be deployed on the testbed.

Also, we can do a real deployment and test the system with many users. For example if possible we can try to deploy the solution in a classroom faculty in the Computer Networks course and ask the students to bring their laptops, tablets or cell phones, and monitor the association of clients to APs. We can set up a tool to measure the throughput, the packet loss and the likes.

Finally, there is another member of the Mina Research group [9] that is doing a research in dense wireless network by handling the transmission power of the APs. An interesting work would be to integrate both results. In this respect, we can implement a combined solution by trying to assign the best channel distribution controlling the transmission power of the APs in order to reduce interference. Hopefully with this combined approach we can approximate to an optimal solution.

Appendix A

Utilities Implementation

As was mentioned in 3.2.2 four operations needs to be available both in the testbed and the simulation. These operations must have the same output because we wish the same version of the RAN System to run in the testbed and in the simulation, with the only change presented in a configuration file where we point to the correct implementation. In this chapter we will show the implementation in the testbed using Linux commands available in openwwrt and the implementation in the simulation where we create java programs that mimic Linux commands and can be called from a shell line using java command line.

A.1 TestBed Utilities Implementation

The implementation of the fourth operations in the testbed are explained in the following subsections

A.1.1 GetAddress

We create a shell script file named *get_address.sh* with the following command:

```
ifconfig wlan0
```

The output of the above command has the format:

```
wlan0     Link encap:Ethernet  direccinHW 68:a3:c4:85:84:31
          Direc. inet:198.18.0.4  Difus.:198.18.0.255  Msc:255.255.255.0
          Direccin inet6: fe80::6aa3:c4ff:fe85:8431/64 Alcance:Enlace
          ACTIVO DIFUSIN FUNCIONANDO MULTICAST  MTU:1500  Mtrica:1
          Paquetes RX:2339 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:1847 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:1955831 (1.9 MB)  TX bytes:367147 (367.1 KB)
```

Lua implementation parse the output looking for the mac address as the identification of the node, this is the piece of information:

```
HW 68:a3:c4:85:84:31
```

A.1.2 GetChannel

We create a shell script file named *get_channel.sh* with the following command:

```
iwlist wlan0 channel
```

The output of the above command has the format:

```
wlan0     13 channels in total; available frequencies :
          Channel 01 : 2.412 GHz
          Channel 02 : 2.417 GHz
          Channel 03 : 2.422 GHz
          Channel 04 : 2.427 GHz
          Channel 05 : 2.432 GHz
          Channel 06 : 2.437 GHz
          Channel 07 : 2.442 GHz
          Channel 08 : 2.447 GHz
          Channel 09 : 2.452 GHz
          Channel 10 : 2.457 GHz
          Channel 11 : 2.462 GHz
          Channel 12 : 2.467 GHz
          Channel 13 : 2.472 GHz
          Current Frequency:2.432 GHz (Channel 5)
```

Lua implementation parse the output looking for last line of the output.

A.1.3 GetCellsInRange

We create a shell script file named *get_cells_in_range.sh* with the following command:

```
iwlist wlan0 scan
```

The output of the above command has the format:

```
wlan0     Scan completed :
          Cell 01 - Address: 00:25:12:6D:17:3D
                    Channel:5
                    Frequency:2.432 GHz (Channel 5)
                    Quality=61/70  Signal level=-49 dBm
                    Encryption key:on
                    ESSID:"TuleWiFi"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s
                    Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s
                              36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=0000022f26857718
                    Extra: Last beacon: 672ms ago
                    IE: Unknown: 000854756C6557694669
                    IE: Unknown: 010482848B96
                    IE: Unknown: 030105
                    IE: Unknown: 0706555920010B1E
                    IE: Unknown: 2A0100
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: 32080C1218243048606C
                    IE: WPA Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (1) : TKIP
                        Authentication Suites (1) : PSK
          Cell 02 - Address: A0:EC:80:50:68:A6
                    Channel:1
                    Frequency:2.412 GHz (Channel 1)
                    Quality=29/70  Signal level=-81 dBm
                    Encryption key:on
                    ESSID:"Mathias"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                              24 Mb/s; 36 Mb/s; 54 Mb/s
```

```
                Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
                Mode:Master
                Extra:tsf=000003129fcb8c55
                Extra: Last beacon: 880ms ago
                IE: Unknown: 00074D617468696173
                IE: Unknown: 010882848B962430486C
                IE: Unknown: 030101
                IE: Unknown: 0706555320010B1E
                IE: Unknown: 2A0104
                IE: Unknown: 2F0104
                IE: IEEE 802.11i/WPA2 Version 1
                    Group Cipher : CCMP
                    Pairwise Ciphers (1) : CCMP
                    Authentication Suites (1) : PSK
                IE: Unknown: 32040C121860
                IE: Unknown: 2D1A1E181BFFFF000001000000000000000000000000000000
                IE: Unknown: 3D16010D160000000000000000000000000000000000000000
                IE: Unknown: DD6A0050F204104A000110104400010210 3B000103104700
                IE: Unknown: DD090010180203F02C0000
                IE: WPA Version 1
                    Group Cipher : CCMP
                    Pairwise Ciphers (1) : CCMP
                    Authentication Suites (1) : PSK
                IE: Unknown: DD180050F20201010000003A4000027A4000042435E0062322F00
        Cell 03 - Address: ......
```

Lua implementation parse the output then iterates over each Cell. In each iteration it takes the lines:

```
Cell 02 - Address: A0:EC:80:50:68:A6
                   Channel:1
                   Frequency:2.412 GHz (Channel 1)
                   Quality=29/70  Signal level=-81 dBm
```

then it store the values in a table like structure with the identification, the channel and the quality of the signal.

A.1.4 SwitchChannel

We create a shell script file named *switch_channel.sh* with the following command:

```
uci set wireless.radio0.channel=$1
uci commit wireless
wifi
ifconfig wlan0 down
ifconfig wlan0 up
```

first we set the new channel in the wireless configuration of the access points with the help of *uci* then we down the wireless interface and start it again so the access point start operating in the new channel.

A.2 Simulation Utilities Implementation

As we mention before, the idea as that the same Lua components that run inside an access point must run inside the simulation. To do this we need to implement a set of operations that produce the same output as the operations presented in section A.1.

There are a couple of conventions for writing a distributed algorithm with DisJ that a programmer has to follow. The entry class for the protocol must extend *distributed.plugin.runtime.engine.Entity*. State for Entity in distributed

protocol must be **public static final int** and the name of the state must begin with *state* case insensitive.

DisJ library has provided standard programming interfaces for programmer to use in their algorithms. In particular they have the following methods:

1. **public String getName():** return the name of an entity.
2. **public int getState():** is for checking a current state of an entity.
3. **public void became(int state):** is for setting a state of an entity to be at a given state.
4. **public List**< *String* > **getPorts():** is list of neighbor nodes name of an entity.
5. **public int getLocationX():** get X coordinate of a node represented in Graph Editor
6. **public int getLocationY():** get Y coordinate of a node represented in Graph Editor
7. **public void receive(String incomingPort, IMessage message):** to receive a message
8. **public void sendTo(String portLabel, Serializable message) :** send a given message to a given neighbor

The operations are written as java programs that can be executed as shell script, this way in a configuration file we will have for the testbed:

```
command_Switch_Channel = "/sbin/switch_channel.sh "
command_Get_Cells_In_Range = "/sbin/get_cells_in_range.sh"
command_Get_Channel = "/sbin/get_channel.sh"
command_Get_Address = "/sbin/get_address.sh"
```

and for the simulation:

```
command_Switch_Channel = "java -cp SimulatorHelper.jar
   uy.edu.fing.mina.agriad.SimulatorHelper.SwitchChannel 10181 "
command_Get_Cells_In_Range = "java -cp SimulatorHelper.jar
   uy.edu.fing.mina.agriad.SimulatorHelper.GetCellsInRange 10181"
command_Get_Channel = "java -cp SimulatorHelper.jar
   uy.edu.fing.mina.agriad.SimulatorHelper.GetChannel 10181"
command_Get_Address = "java -cp SimulatorHelper.jar
   uy.edu.fing.mina.agriad.SimulatorHelper.GetAddress 10181"
```

The last argument of the java program call is a socket port number, because each node graph when they are awaken start a server socket to listen for request.

Then from lua script file we just call:

```
local t = run_shell(configuration.command_Get_Address)
```

We need to parse t to get the Address. Again, to have only one implementation of lua file both commands the one for the testbed and the one for the simulation must produce the same output.

To implement the commands for the simulation we define the states of the entities as the channel of the AP, so we define:

```
public static final int STATE_CHANNEL_UNO = 1;
public static final int STATE_CHANNEL_SEIS = 6;
public static final int STATE_CHANNEL_ONCE = 11;
```

A.2.1 GetChannel

The java program connects (as we explained this connection is via socket) to the node graph and ask for the channel. The entity using the API calls the method **public int getState()** to get the channel and send it back to the java program that produce the same output as *GetChannel* for the testbed except that in the last line:

```
Current Frequency:2.432 GHz (Channel 5)
```

it change the values according to node sate.

A.2.2 GetAddress

The java program connects to the node graph and ask for the address. The entity using the API calls the method **public String getName()** to get the address and send it back to the java program that produce the same output as *GetAddress* for the testbed except that in the line:

```
wlan0     Link encap:Ethernet  direccinHW 68:a3:c4:85:84:31
```

it change the value of the mac address for the node name:

```
wlan0     Link encap:Ethernet  direccinHW nodename1
```

A.2.3 SwitchChannel

The java program connects to the node graph and ask the entity to change channel passing through the socket interface the new channel number to switch.

The entity using the API calls the method **public void became(int state)** to change channel. For DisJ this is only a change in the state of the entity.

A.2.4 GetCellsInRange

The three commands presented before were very easy to implement and there is a good analogy between the simulation and real implementation. The identification of an access point is your mac address while the identification of an entity is its node name. The channel the access point is operating on is the state of an entity and changing the channel in a access point is equal to change the state in an entity.

The problem arise in this more complex command. To implement this command we study the source code of iwlist implementation and look for the behavior when it is invoked with the *scan* parameter. *iwlist* and the other wireless tool provide a common front end to different wireless device drivers that support *Linux Wireless Extension (WEXT)*. Each driver will register handlers with WEXT that implement the device specific operations defined by this interface.

iwlist make use of the traditional *ioctl*. First it open a socket:

```
/* Create a channel to the NET kernel. */
 sock = socket(AF_INET, SOCK_DGRAM, 0);
```

Then it start scanning:

```
/* Initiate Scanning */
 struct iwreq wrq;
 /* Set device name */
 strncpy(wrq.ifr_name, "wlan0", IFNAMSIZ);
 /* Do the request */
 ioctl(skfd, SIOCSIWSCAN, &wrq)
```

and finally it get the results

```
/* Try to read the results */

 wrq.u.data.pointer = buffer;
 wrq.u.data.flags = 0;
 wrq.u.data.length = buflen;
 /* Set device name */
 strncpy(pwrq.ifr_name, "wlan0", IFNAMSIZ);
 /* Do the request */
 ioctl(skfd, SIOCGIWSCAN, &wrq)

 if(wrq.u.data.length)
 {
//print results.
 }
```

where SIOCSIWSCAN and SIOCGIWSCAN are defined in wireless.h as follow:

```
#define SIOCSIWSCAN 0x8B18 /* trigger scanning (list cells) */
#define SIOCGIWSCAN 0x8B19 /* get scanning results */
```

The important point to note here is that the *ioctl* calls are implemented in the drivers module so the actual function call for *ioctl* using *SIOCGIWSCAN* will invoke a function defined in the drivers module. In other words the *ioctl* calls are implemented in the WEXT module but the code that handles this command is implemented in the device driver. When a user space application makes an *ioctl*, WEXT looks up the device drivers handler and runs it.

Note that the device is free to implement the scan how ever it chooses. For example, it can passively listen for beacons or actively scan by sending out probe requests.

Because of this we choose our own implementation of the *iwlist wlan0 scan*. The only constraint is to keep tied with the output.

The cells in range of a node (an entity in DisJ) are the result of invoking **public List**< *String* > **getPorts()**. The meaning of this is that the cells that can interfere with the node (the nodes that are seen by the access point) are the list of neighbor of the entity.

This is a direct relationship with the real implementation. As we said for the channel and node states, mac address and node names then we can say that

the interferers access points are the list of neighbor of the node. The hard part is to implement for each cell listed in the command result the following output line:

```
Quality=29/70  Signal level=-81 dBm
```

Our implementation in the simulation is a function of the distance between the nodes in the graph drawn in DisJ. Remember from the API of DisJ the methods **public int getLocationX()** and **public int getLocationY()** to get the node position. When a node is requested for his cells in range it in turn request each neighbor (using **public void sendTo(String portLabel, Serializable message)** asking for their coordinates. The *portLabel* parameter in DisJ terminology is the name of a neighbor node and the *message* parameter usually is a any *String*. Then the node can compute the distance to each neighbor and this way it can send a response with the list of cells and the quality of the signal.

Bibliography

[1] Course notes for design and analysis of distributed algorithms. http://people.scs.carleton.ca/~santoro/DADA-CourseNotes.html.

[2] Disj (simulator for distributed algorithms). http://people.scs.carleton.ca/~santoro/DADA-Simulator.html.

[3] Fap web. http://fap.zib.de. [Online; accessed 2-September-2013].

[4] Gef (graphical editing framework). http://www.eclipse.org/gef/.

[5] Gnuplot. http://www.gnuplot.info/.

[6] Hollywood's failed wi-fi rankles residents. http://articles.sun-sentinel.com/2012-07-02/news/fl-hollywood-failed-wi-fi-20120630_1_wi-fi-wireless-hollywood-transmitters-without-signal-interference. [Online; accessed 12-September-2013].

[7] How many wireless clients an access point can support. https://supportforums.cisco.com/docs/DOC-5500. [Online; accessed 28-December-2013].

[8] Meru networks. http://www.merunetworks.com/microsites/meltdown/index.php. [Online; accessed 2-September-2013].

[9] Mina research group. http://www.fing.edu.uy/inco/grupos/mina/.

[10] Multi generator (mgen). http://cs.itd.nrl.navy.mil/work/mgen/.

[11] Open source implementation of the lightweight directory access protocol. http://www.openldap.org/.

[12] The openwrt project. http://openwrt.org/.

[13] Smarta. http://blizzard.cs.uwaterloo.ca/tetherless/index.php/SMARTA.

[14] Trace plot real-time. http://pf.itd.nrl.navy.mil/proteantools/trpr.html.

[15] N. Ahmed and S. Keshav. Smarta: a self-managing architecture for thin access points. In *Proceedings of the 2006 ACM CoNEXT conference*, CoNEXT '06, pages 9:1–9:12, New York, NY, USA, 2006. ACM.

[16] 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.

[17] J. Antoniou, A. Pitsillides, and L. Libman. A game theory-based approach to reducing interference in dense deployments of home wireless networks. In *Proceedings of the 2011 IEEE Symposium on Computers and Communications*, ISCC '11, pages 976–982, Washington, DC, USA, 2011. IEEE Computer Society.

[18] Javier Baliosian and Joan Serrat. Finite state transducers for policy evaluation and conflict resolution. In *Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, pages 250–259, June 2004.

[19] Javier Baliosian, Jorge Visca, Eduardo Grampín, Leonardo Vidal, and Martín Giachino. A rule-based distributed system for self-optimization of constrained devices. In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, IM'09, pages 41–48, Piscataway, NJ, USA, 2009. IEEE Press.

[20] Ioannis Broustis, Konstantina Papagiannaki, Srikanth V. Krishnamurthy, Michalis Faloutsos, and Vivek Mhatre. MDG: measurement-driven guidelines for 802.11 WLAN design. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 254–265, New York, NY, USA, 2007. ACM.

[21] Cisco, editor. *High Density Client Deployment Guide for Cisco Wireless LANs*. Cisco, 2011.

[22] G. Colombo. A genetic algorithm for frequency assignment with problem decomposition. *International Journal of Mobile Network Design and Innovation*, 1(2):102–112, 2006.

[23] Yixin Diao, Neha G, Joseph L. Hellerstein, Sujay Parekh, and Dawn M. Tilbury. Using mimo feedback control to enforce policies for interrelated metrics with application to the apache web server. In *In Proceedings of the Network Operations and Management Symposium 2002*, pages 219–234, 2002.

[24] Athanasios D. Panagopoulos Dimitris E. Charilas. A survey on game theory applications in wireless networks. `http://koti.mbnet.fi/~laitinej/jouni/School/artikkelit/Asurveyongametheoryapplicationsinwirelessnetworks.pdf`. [Online; accessed 9-September-2013].

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

[26] Jim Geier. wi-fi planet: Assigning 802.11b access point channels. `http://www.wi-fiplanet.com/tutorials/article.php/972261`. [Online; accessed 3-October-2013].

[27] Antonis M. Hadjiantonis and George Pavlou. Policy-based self-management of wireless ad hoc networks. In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, IM'09, pages 796–802, Piscataway, NJ, USA, 2009. IEEE Press.

[28] Joseph L. Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M. Tilbury. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.

[29] A. W. J. Kolen. A genetic algorithm for frequency assignment. *Statistica Neerlandica*, 61(1):4–15, 2007.

[30] Kin K. Leung. Frequency assignment for IEEE 802.11 wireless networks. In *Proc. 58th IEEE Vehicular Technology Conference*, pages 1422–1426, 2003.

[31] Vivek Mhatre, Konstantina Papagiannaki, and François Baccelli. Interference Mitigation Through Power Control in High Density 802.11 WLANs. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 6-12 May 2007, Anchorage, Alaska, USA*, pages 535–543. IEEE, 2007.

[32] Uruguay Mina Research Group. Rural Ambient Network project. `http://www.ran.org.uy/wiki/field.php`.

[33] A. Mishra, V. Brik, S. Banerjee, Aravind Srinivasan, and William A. Arbaugh. A client-driven approach for channel management in wireless lans. *IEEE Infocom*, 6, 2006.

[34] Arunesh Mishra, Suman Banerjee, and William Arbaugh. Weighted coloring based channel assignment for wlans. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(3):19–31, July 2005.

[35] Tnia L. Monteiro, Guy Pujolle, Marcelo Eduardo Pellenz, Manoel Camillo Penna, and Richard Demo Souza. An optimal channel assignment strategy for wlans using distributed optimization. In *NOMS*, pages 286–292. IEEE, 2012.

[36] Rohan Murty, Jitendra Padhye, Ranveer Chandra, Alec Wolman, and Brian Zill. Designing high performance enterprise Wi-Fi networks. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 73–88, Berkeley, CA, USA, 2008. USENIX Association.

[37] Wireless Sensor Net. Types of interference. `http://wirelesssensornet.blogspot.com/2010/05/types-of-interference.html`, 2010. [Online; accessed 2-September-2013].

[38] Juniper network. Understanding channel auto-tuning on wireless networks. `http://www.juniper.net/techpubs/en_US/junos-space-apps12.3/network-director/topics/concept/auto-tune-wireless.html`, 2014. [Online; accessed 7-June-2014].

[39] Aruba Networks. High-density wireless networks for auditoriums validated reference design. `http://www.arubanetworks.com/pdf/technology/DG_HighDensity_VRD.pdf`. [Online; accessed 2-September-2013].

[40] Sujay Parekh, Joe Hellerstein, T. S. Jayram, Neha Gandhi, Dawn Tilbury, and Joe Bigus. Using control theory to achieve service level objectives in performance management, 2001.

[41] Nicola Santoro. *Design and Analysis of Distributed Algorithms (Wiley Series on Parallel and Distributed Computing)*. Wiley-Interscience, 2006.

[42] Eduard Garcia Villegas, Elena Lopez-Aguilera, R. Vidal, and J. Paradells. Effect of adjacent-channel interference in ieee 802.11 wlans. In *Cognitive Radio Oriented Wireless Networks and Communications, 2007. CrownCom 2007. 2nd International Conference on*, pages 118–125, 2007.

[43] L. Wang and W. Gu. Genetic algorithms with stochastic ranking for optimal channel assignment in mobile communications. *Lecture Notes in Computer Science*, 3314:154–159, 2004.

[44] Wikipedia. Plagiarism — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/List_of_WLAN_channels`, 2004. [Online; accessed 22-July-2013].

[45] Wikipedia. Plagiarism — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Prisoner_dilemma`, 2004. [Online; accessed 22-July-2013].

[46] Xiaonan Yue, Chi-Fai Michael Wong, and Shueng-Han Gary Chan. Cacao: Distributed client-assisted channel assignment optimization for uncoordinated wlans. *IEEE Transactions on Parallel and Distributed Systems*, 22:1433–1440, 2011.