UNIVERSITY OF CALIFORNIA
RIVERSIDE

# Resource Allocation Techniques to Improve the Performance of Wireless Networks

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Mustafa Yasir Arslan

September 2012

# ACKNOWLEDGMENTS

I would like to start by expressing my gratitude to my Ph.D. advisor, Dr. Srikanth V. Krishnamurthy, whose role in this dissertation was much more than just “advising” it. Throughout my studies, he has shown incredible support and guidance, provided me with his vision and helped me approach several research problems in the right way. He has also been a close friend, who has listened to my daily problems and often showed me the way out of them, which kept me back up on my feet after every obstacle. I would also like to thank Dr. Michalis Faloutsos for his support and for showing me that research can indeed be fun and entertaining. I would like to thank Dr. Harsha V. Madhyastha for his invaluable collaboration and his close interest in my work during the short time that we have worked together. I would also like to thank Dr. Ertem Tuncel for participating in my defense committee. In short, I feel privileged to be recognized as a doctor by such a distinguished committee. In addition, I wish to thank Dr. Karthikeyan Sundaresan and Dr. Sampath Rangarajan for giving me a chance to have an internship at NEC Laboratories America Inc., and for their excellent mentoring and constant support. I would also like to thank my friends and colleagues who have made this long journey a lot more enjoyable and memorable: Arda Atali, Dr. Ali Umut Irturk, Efe Karasabun, Berk Atikoglu, Dr. Konstantinos Pelechrinis, Dr. Ioannis Broustis, Marios Kokkodis, Dr. Theodoros Lappas, Dr. Marios Iliofotou and many more.

My extended family during my Ph.D. deserves a special mention as well: Muzaffer Akbay and Busra Celikkaya for being a lot more than a close friend, Makbule Koksal for her awesome cooking and in general for caring so much about me, Abdurrahman Koksal for advising me in every problem I had, Attila Koksal, Asli Koksal and Omay Koksal for being my stress-busters.

Finally, I am fortunate to be a member of the Arslan family who have raised me to be the person I am, supported me in every phase of my life and loved me from the deepest of their hearts. This dissertation would certainly not be possible if I did not have their support, warmth and encouragement. I thank and love them all.

*To my love Yasemin and my family*

# ABSTRACT OF THE DISSERTATION

Resource Allocation Techniques to Improve the Performance of Wireless Networks

by

Mustafa Yasir Arslan

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, September 2012
Dr. Srikanth V. Krishnamurthy, Chairperson

Wireless networks are increasingly becoming important in enabling convenient Internet access for users. Many users have WiFi networks in their homes and have access to cellular networks (e.g., 3G, 4G) when they are on the go. Unfortunately, wireless networks do not offer performance guarantees. The download and upload rates achieved with these networks fluctuate and may often degrade to the point where they become practically unusable. One of the major factors that causes this performance degradation is interference, which can be defined as the unwanted noise generated by other devices in the wireless spectrum.

The root cause of interference lies within the fact that the open nature of wireless communications enables multiple stations to access the spectrum simultaneously. To eliminate interference, wireless communication systems offer multiple orthogonal channels [1]. If multiple stations access the spectrum using these orthogonal channels, interference is very likely to be avoided (assuming that the signal "leak" between orthogonal channels is negligible), as opposed to the case where stations use the same channel. Therefore, one needs to determine the specific access patterns (i.e., which channel is used by which transmitter at what point in time) to systematically coordinate the transmissions of multiple wireless stations.

[1]These are channels with non-overlapping frequency spans and hence, they are called orthogonal.

While a large body of previous work exists to coordinate the use of orthogonal channels between IEEE 802.11 a/b/g stations, we show that they are not able to address the interference problem for IEEE 802.11n devices due to the use of wide channels in 802.11n systems. Thus, we propose a WLAN management system – called *ACORN* – that not only allocates orthogonal channels to 802.11n access points but also makes intelligent user association decisions to significantly improve aggregate WLAN throughput. Later, we study the interference in cellular networks, in particular OFDMA femtocell systems. We experimentally characterize the interference between OFDMA femtocells and propose design guidelines that help in building resource management solutions to alleviate interference. We observe that resource management solutions for OFDMA femtocells need to implement unique functionalities due to the fundamental differences between OFDMA access technology and 802.11 WiFi systems. Leveraging the guidelines revealed by our study, we later propose *FERMI* – a resource management solution that mitigates interference between OFDMA femtocells and at the same time leverages spatial reuse opportunities. We implement *FERMI* on a WiMAX femtocell testbed and show that it provides throughput benefits as much as *7x* over a baseline scheme and it is closer to the optimal solution than its simpler theoretical alternatives.

In addition to the multi-cell interference setting addressed by our above-mentioned contributions, we study beamforming in a single cell scenario. Beamforming is a technique that allows focusing the energy of transmissions in a particular direction to improve overall signal quality. Since OFDMA schedules multiple users in the same frame (and this is in contrast to WiFi), designing data scheduling solutions and at the same time benefiting from beamforming is challenging. We propose *iBUS* – a system that addresses uplink data scheduling with beamforming. We show that *iBUS* is provably efficient in striking the right tradeoffs inherent in this setting. Evaluations from both a prototype system implementation and simulations yield that *iBUS* improves the throughput by more than 40% compared to an omni-directional scheduling scheme.

Finally, we study distributed computing on smartphones. We identify that there are a large number of phones that are left idle (often when being charged overnight) for long periods of time. We envision that an enterprise could use such idle smartphones (e.g., phones of its employees) for enabling parallel computing on them. We believe that smartphones offer precious computing resources that have not previously been tapped into from a distributed computing perspective. In addition to the computing capacity being offered, our cost analysis and projections show that smartphones are also an energy-efficient alternative to using dedicated servers for computing purposes. However, enabling distributed computing on a large number of phones is complicated due to the unique set of challenges that smartphones present, such as heterogeneity in CPU speed and variability in network bandwidth. We design and implement *CWC* – a distributed computing infrastructure using smartphones. Our extensive evaluations demonstrate that *CWC* can schedule tasks for $1.6x$ faster completion that alternative mechanisms.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The growing popularity of wireless networks have made them accessible to an increasing number of users over the last decade. Today, a typical network scan produces results where many WiFi access points (APs) are present and contend for access to the wireless spectrum (an example can be seen in Figure 1.1). While theoretical data rates are constantly increasing with the introduction of latest wireless technologies (e.g., IEEE 802.11n and 4G LTE, WiMAX), the "crowded" nature of the wireless spectrum results in interference between stations and thus, introduces a bottleneck in wireless data rates achieved in practice. In addition to the contention between multiple access points, a single access point is often inhabited by multiple users (i.e., clients) and thus, the capacity that it offers is also subject to contention among its clients. In summary, the contention for wireless capacity exists both in the *inter-cell* setting as well as the *intra-cell* setting [1]. If not coordinated properly, this contention can drastically reduce wireless communication performance.

In this dissertation, we first present resource allocation techniques that coordinate the sharing of the wireless spectrum to address two distinct, but equally important problems: **(i):** interference between IEEE 802.11n WiFi APs and interference between OFDMA femtocells, and **(ii):** uplink data scheduling across the clients of a single OFDMA femtocell [2]. In addition to spectrum man-

[1]We loosely use the terms "cell", "access point", "base station" interchangeably in the rest of this dissertation.

[2]The term "downlink" represents the data flow from the access points to the clients and "uplink" represents the opposite direction.

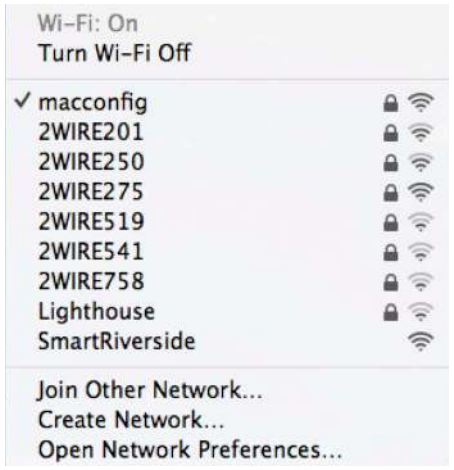


Figure 1.1: A typical residential WiFi network scan profile where there are multiple access points in the spectrum.

agement, we also present resource allocation techniques to carefully allocate computing resources in a distributed computing infrastructure on smartphones. We show that our techniques offer significantly higher performance than their state-of-the-art alternatives. Next, we detail our contributions in solving each of these problems.

## 1.1 Resource Management and Its Application to Alleviate Interference

Due to the broadcast nature of the wireless spectrum, the transmissions of a particular station on a given channel is "heard" by other stations (tuned to the same channel) that are close enough to the transmitter (e.g., stations within the communication range of the transmitter). While this forms the basis for wirelessly communicating with intended recipients, it inevitably exposes the communication to the noise generated by other wireless devices operating on the same channel. This creates two problems for WiFi networks: **(i):** since WiFi uses carrier sense multiple access (CSMA), the transmitters may back off (i.e., defer their transmission after sensing the channel to be “busy”) and **(ii):** the intended signal and the noise may collide at the receivers, degrading the quality of decoding. Both of these cases result in a decrease in the achievable data rate of a particular wireless link. Even though OFDMA systems are not vulnerable to the former (since they do not employ CSMA), collisions due to interference still exist and degrade the link throughput.

### 1.1.1 Interference Mitigation in WiFi

WiFi systems support two mechanisms in their basic toolkit to alleviate interference. First, they partition the spectrum into multiple orthogonal channels (e.g., IEEE 802.11a has 12 orthogonal channels in North America). These channels are arranged so that they have different central frequencies and their frequency span (i.e., bandwidth) do not overlap with each other. If links use these orthogonal channels, they do not interfere (assuming the leakage between adjacent orthogonal channels is not significant). Second, even if two WiFi links are tuned to the same channel, carrier sensing provides some level of arbitration since stations defer their transmissions in the presence of other transmissions. However, carrier sensing comes at the cost of reduced data rate since the channel is accessed less frequently as compared to the case where all links are tuned to orthogonal channels. Thus, much of the previous work (e.g., [2, 3]) has proposed algorithms that assign orthogonal channels to interfering APs for legacy WiFi systems (IEEE 802.11a/b/g). In these systems, the channels are of fixed bandwidth (e.g., 20 MHz for 802.11a and 22 MHz for 802.11g).

The recent IEEE 802.11n standard introduces the option of using wider channels to significantly improve the data rates. With 802.11n, an AP can either use a 20 MHz channel or a 40 MHz channel by combining two adjacent 20 MHz channels. While the common wisdom has been in the favor of using 40 MHz channels [4] [3], we show that wider channels may sometimes degrade throughput in practice. Our key observation is that when a WiFi AP transmits using some power $P$, the energy is evenly split across the sub-carriers forming a channel. The use of wider channels with 802.11n results in *less energy per sub-carrier* since a 40 MHz channel has around twice the number of sub-carriers in a 20 MHz channel. While certain links benefit from the increased capacity of wider

[3]In theory, doubling the bandwidth should double the raw data rate.

channels, the reduction in sub-carrier energy can indeed result in degraded throughput for some links. Thus, assigning channels in 802.11n systems requires careful selection of the channel width, in addition to simply assigning orthogonal channels. Stated otherwise, it is imperative to design channel allocation solutions that identify the set of links that can benefit from wider channels and consider the application of wider channels exclusively for such links. The above-mentioned studies targeted for legacy WiFi systems do not account for the bandwidth vs. energy tradeoffs with wider channels and thus, are not suitable for 802.11n WLANs.

### 1.1.2 Our Contributions on WiFi Interference Mitigation

We conduct experiments to investigate the use of wider channels in practical 802.11n WLAN deployments. Our experiments reveal that wider channels do not always provide throughput benefits and for some links, they can even degrade the throughput. Such adverse affects of wider channels have not been previously demonstrated for 802.11n systems. Leveraging this key observation, we design and implement *ACORN* [5]. *ACORN* is a resource management system for 802.11n WLANs that **(i):** assigns orthogonal channels to interfering APs together with determining the set of APs that should use wider channels and **(ii):** involves a user association mechanism that carefully addresses the bandwidth vs. energy tradeoffs inherent in wider channels. We implement *ACORN* on a 802.11n testbed and show that the joint application of these two mechanisms significantly improve the throughput (as high as *6x* per AP) in comparison to state-of-the-art solutions proposed by prior studies.

### 1.1.3 Interference Mitigation in OFDMA Femtocells

Femtocells are small cellular base stations that are deployed indoors (e.g., residences, malls, enterprise settings). These small cells use the same access technology and standards as macrocells (traditional outdoor cell towers). The latest cellular systems (e.g., 4G LTE and WiMAX) use Orthogonal Frequency Division Multiple Access (OFDMA) as their access technology. Similar to

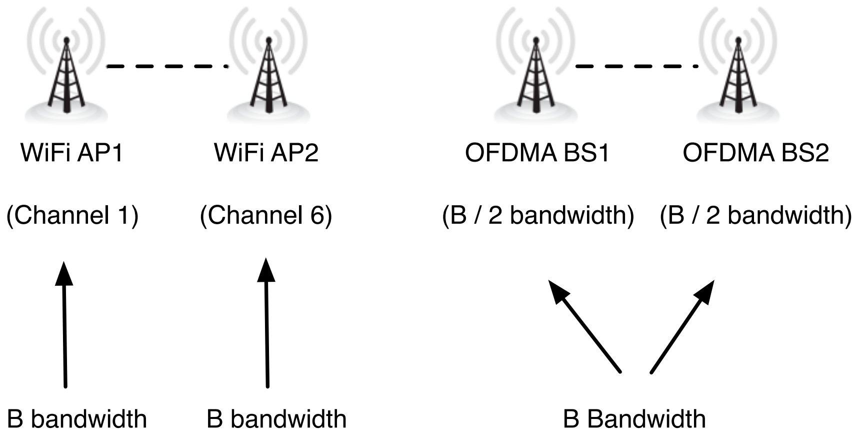


Figure 1.2: Channel allocation comes at the cost of reduced bandwidth for OFDMA femtocells. In contrast, WiFi systems have the same available bandwidth on each of the orthogonal channels.

WiFi systems, OFDMA femtocells also suffer from interference when deployed in close proximity to each other. Before detailing how interference manifests in OFDMA femtocells, we first describe three key differences between OFDMA access technology and WiFi.

1. In contrast to the unlicensed spectrum and the availability of multiple channels in WiFi, OFDMA spectrum is licensed. Cellular operators typically purchase one frequency band that is used by both femtocells and macrocells. The notion of orthogonality is supported by having multiple orthogonal fragments of the purchased band. These fragments are called *sub-channels*.

2. OFDMA does not employ carrier sensing. If two femtocells use overlapping sub-channels, they directly project interference on each other's clients (i.e., there is no busy channel back off).

3. In a single OFDMA frame, the base station schedules multiple clients. In contrast, WiFi frames contain data only to/from multiple clients.

These differences make interference management both unique and challenging in OFDMA femtocell systems. We describe this with a simple example in Figure 1.2. On the left side of this example, we have two interfering WiFi APs tuned to channel 1 and channel 6 in the 2.4 GHz band [4]. Since WiFi spectrum consists of multiple channels of equal bandwidth and each AP is allowed to use only one channel, both APs continue to have the same raw wireless bandwidth ($B$) irrespective of the specific channel that they are tuned to [5]. In other words, discovering a set of orthogonal channels for interfering APs suffices for interference mitigation in WiFi. For OFDMA on the other hand, one needs to allocate orthogonal parts of the spectrum to interfering femtocells. Given that there is only one frequency band, the band needs to be partitioned across interfering cells and this comes at the cost of reduction in the channel capacity for each cell [6]. Note also that OFDMA schedules data to multiple clients in the same frame. While some of these clients may be subject to interference (and require the assignment of orthogonal spectrum partitions), some clients may experience no (or negligible) interference from other cells. Thus, blindly assigning orthogonal parts of the spectrum unnecessarily reduces the channel capacity for the latter class of clients. WiFi systems do not bear such performance costs since each AP transmits to a single client at a time, using the entire channel bandwidth assigned to it.

In addition to the technical differences between WiFi and OFDMA, there also exists deployment differences between femtocells and macrocells (even though they implement the same set of OFDMA standards). Macrocells are typically deployed in a well-planned fashion, in which the operators conduct a significant amount of site surveys (signal strength evaluations) and frequency planning to determine the placement of the cell towers. Femtocells on the other hand, are purchased and deployed by end-users. While this presents a cost-effective opportunity for the operators to increase their cellular coverage, it also makes femtocell deployments uncontrolled and ad-hoc.

Prior studies that focused on interference mitigation in OFDMA systems leverage the planned deployments of macrocells in their solutions. Since cell boundaries are known, the determination of which set of clients are subject to interference is an easier task as compared to femtocell de-

[4]There are only three orthogonal channels ($1, 6$ and $11$) available in the 2.4 GHz band.

[5]Note that this is only theoretical bandwidth. If an AP uses a channel occupied by another AP, the practical data rate can be worse than the theoretical maximum.

[6]We assume equal partitioning for simplicity. It is easy to see that the capacity reduction occurs for any arbitrary partition.

ployments. Although there has been recent studies addressing interference mitigation for OFDMA femtocells (e.g., [6]), they are mostly theoretical studies (with simplifying assumptions) and thus, have limited applicability in practice.

### 1.1.4 Our Contributions on Femtocell Interference Mitigation

Given the fundamental differences between OFDMA and WiFi, we first conduct an experimental study [7] to understand the design choices that need to be made for interference mitigation in OFDMA femtocells. In these experiments, we compare and contrast alternative resource management strategies and derive a set of guidelines that help in designing efficient interference management solutions for OFDMA femtocells. Leveraging these guidelines, we design and implement *FERMI* [8], which is a standards-compatible resource management solution that effectively mitigates interference and at the same time leverages spatial resource opportunities in a OFDMA femtocell network. Although the objective of femtocell interference mitigation is similar to previous studies, *FERMI* distinguishes itself with being practical, scalable and compatible with the standards. Previous studies are mostly theoretical and have simplifying assumptions, which in turn limits their adoption in practice. *FERMI* incorporates the following key features to offer a full-fledged resource management solution.

1. It categorizes clients to determine the severity of interference projected by neighboring cells. This determines whether to isolate the resources or reuse them for each client.

2. Given clients with different requirements (isolation vs. reuse), it involves intelligent scheduling mechanisms for graceful existence of both types of clients in the same frame.

3. It allocates and assigns sub-channels to interfering femtocells subject to a max-min fairness model. This is achieved in a scalable and efficient way with clever use of chordal graphs.

## 1.2 Resource Management and Its Application to Data Scheduling in Femtocells

Beamforming is a signal processing technique that helps focusing the transmitted/received energy to/from particular directions. This is in contrast to omni-directional communications, where the signals radiate and lose energy as they travel in all directions (hence, the term "omni") in space. By focusing the beams, a transmitter can boost the signal strength at a particular receiver (called transmit beamforming) or similarly it can increase its received signal power from transmitters situated at particular directions (called receive beamforming).

While beamforming has been extensively studied for WiFi systems (e.g., [9]), the fundamental differences between WiFi and OFDMA render such studies inapplicable to OFDMA femtocells. As alluded to previously, OFDMA schedules multiple clients in the same frame as opposed to WiFi systems that schedule one client per frame. This creates an interesting tradeoff for receive beamforming in OFDMA. In particular, OFDMA benefits from scheduling uplink transmissions from multiple clients by allocating fewer frame resources to each client and the resulting per-resource power increase from it [7]. However, this makes it more challenging to leverage the directionality gains of receive beamforming; it now becomes harder to find a common beam that works well for all multiplexed signals coming from different directions. Thus, there is an evident need for novel uplink scheduling mechanisms that strike the right tradeoff between the multiplexing gain on one hand, and the beamforming gain on the other.

### 1.2.1 Our Contributions on Femtocell Uplink Scheduling

To address the above-mentioned tradeoff, we propose *iBUS*. *iBUS* is an uplink scheduling system that involves simple, yet efficient algorithms with a worst case approximation ratio of $\frac{1}{2}$ in comparison to an optimal resource allocation [8]. A distinguishing feature of *iBUS* is its practicality; our

[7]This is similar to the per-subcarrier energy increase when wider channels are avoided in 802.11n as described in Section 1.1.1.

[8]We show the problem to be even hard to approximate in polynomial time.

design ensures that it is agnostic to specific OFDMA standards (i.e., can be incorporated in most modern OFDMA systems) and hence, is not limited by the beamforming support defined in these standards. We implement a prototype of *iBUS* using real OFDMA femtocells and clients to evaluate its benefits in practice. Our evaluations show that *iBUS* improves the uplink throughput by more than 40% as compared to an omni-directional scheduler.

## 1.3 Resource Management and Its Application to Distributed Computing on Smartphones

Smartphones are increasingly becoming popular due to a multitude of features that they offer such as mobile access to the Internet, entertainment applications (e.g., games) and their advanced sensors such as accelerometers and GPS. It is often projected that the number of smartphones will soon surpass the number of PCs shipped worldwide [10, 11]. The popularity of smartphones has also manifested for enterprises as there are a large number of companies that have started giving out smartphones to their employees for various reasons [12]. We argue that an enterprise can benefit from harnessing the computing power of such smartphones of its employees, to construct a distributed computing infrastructure.

Our motivations for enabling distributed computing on smartphones are two-fold: **(i)** smartphone CPUs are increasingly becoming faster and approaching the clock speed of a traditional PC CPU. A large number of smartphones, when allowed to work on some computation in parallel, could provide significant computing power that could outperform that of a typical server. **(ii)** Smartphones are much more (an order of magnitude) energy-efficient than typical servers. Thus, computations can be executed with high energy efficiency while harnessing the same (or even higher than) computing power of PCs. Furthermore, we identify that most of these smartphones are left idle for long durations (e.g., being recharged overnight) and thus constitute a sizable infrastructure that is waiting to be tapped into.

### 1.3.1 Our Contributions on Distributed Computing Using Smartphones

Building such a distributed computing platform using smartphones however, is complicated by many technical challenges. First, smartphones have heterogeneous computing capabilities. This makes it harder to perform efficient task scheduling (e.g., load balancing) than typical servers which often come in preset configurations. Second, the fluctuations in wireless bandwidth requires the existence of scheduling mechanisms that are aware of the time it takes to copy a task to a smartphone and get the results back from it. Third, smartphones are highly personal devices. Thus, using them for distributed computing purposes should come at no (or negligible) distraction to the user experience.

To address the above-mentioned challenges, we develop *CWC* – a distributed computing infrastructure using smartphones. Our contributions in building *CWC* is as follows.

1. We envision running tasks on phones only when they are being charged so as to minimize concerns for battery life. Thus, we examine charging behaviors of 15 volunteers and identify that there is significant idle charging time in smartphone usage.

2. We design a scheduler that minimizes the makespan of a set of jobs waiting to be scheduled on smartphones. We show that optimally allocating jobs to smartphone resources is NP-hard. Our scheduler is based on a greedy algorithm and we experimentally validate that it outperforms its simpler alternatives.

3. We implement an approach to efficiently migrate tasks when their execution fails before completion. In our implementation, tasks are suspended if phones are plugged off the power source during execution. Thus, tasks are migrated to other phones that are plugged for charging and resumed from the point that they failed.

4. We show that blindly running CPU-intensive tasks on a smartphone (when it is being charged) can delay a full charge duration by as much as 35%, thus significantly deteriorating user

experience. We design and implement a CPU throttling mechanism, which minimizes the adverse impact of running tasks on the time required to charge a smartphone.

5. We implement a prototype of *CWC* on a testbed consisting of 18 Android smartphones. We quantify the practical performance of our scheduler and its task migration capabilities and show that *CWC* achieves a makespan that is $1.6x$ shorter than that of simpler heuristics.

## 1.4 Outline of this dissertation

The rest of this dissertation is organized as follows. In chapter 2, we detail the design of *ACORN* – our framework to manage user association and channel selection functions in 802.11n WLANs. *ACORN* is the first system that intelligently uses the wide channels feature in 802.11n and thus, outperforms previous WLAN management solutions that are agnostic to the tradeoffs inherent in using this feature. In chapter 3, we describe our experimental study that reveals an interesting set of observations and guidelines for efficient interference management solutions for OFDMA femtocells. Chapter 4 presents *FERMI* – our interference management solution for OFDMA femtocells. *FERMI* mitigates interference in a femtocell network and at the same time leverages spatial reuse opportunities to provide significant throughput benefits. In chapter 5, we describe the design of *iBUS*, which is an uplink data scheduling solution that considers the application of beamforming in OFDMA femtocells. Finally in chapter 6, we present *CWC* – our distributed computing solution on smartphones. The performance of *CWC* as revealed by our prototype evaluation demonstrates that our vision is viable in practice.

# Chapter 2

# ACORN: An Auto-configuration Framework for 802.11n WLANs

The wide channels (WC) feature combines two adjacent channels to form a new, wider channel to facilitate high data rate transmissions in MIMO-based 802.11n networks. Using a wider channel can exacerbate interference effects. Furthermore, contrary to what has been reported by prior studies, we find that WC does not always provide benefits in isolation (i.e., one link without interference) and can even degrade performance. We conduct an in-depth, experimental study to understand the implications of WC on throughput performance. Based on our measurements, we design an auto-configuration framework called ACORN for enterprise 802.11n WLANs. ACORN integrates the functions of user association and channel allocation, since our study reveals that they are tightly coupled when WC is used. We show that the channel allocation problem with the constraints of WC is NP-complete. Thus, ACORN uses an algorithm that provides a worst case approximation ratio of $O(\frac{1}{\Delta+1})$, with $\Delta$ being the maximum node degree in the network. We implement ACORN on our 802.11n testbed. Our evaluations show that ACORN (i) outperforms previous approaches that are agnostic to WC constraints; it provides per-AP throughput gains ranging from *1.5x* to *6x* and (ii) in practice, its channel allocation module achieves an approximation ratio much better than the theoretically predicted $O(\frac{1}{\Delta+1})$.

## 2.1 Introduction

The IEEE 802.11n [13] is based on the use of MIMO technology and promises drastically improved throughputs as compared to legacy 802.11 systems (a / b / g). In order to achieve high data rates ($>$ 100 Mbps), 802.11n supports a feature called wide channels (WC). With WC, two adjacent channels can be combined to form a new, wider frequency band; this is expected to support transmissions at higher bit rates (i.e., bandwidth). The default channel width is 20 MHz and it is increased to 40 MHz when WC is activated. To realize increased data rates, 802.11n uses 114 subcarriers with 40 MHz channels (as compared to 56 subcarriers with 20 MHz channels) in a given OFDM symbol.

The use of WC however, has associated caveats. Communication links operating on 40 MHz channels project increased interference on other links and can negatively impact the total network throughput [14]. On the other hand, prior studies have only reported results suggesting that WC would yield significantly higher throughput in settings *without* interference [14, 15]. Surprisingly, our experiments suggest the contrary in quite a few cases; the throughput performance with WC (for a single link in isolation) can be even worse than that with a single 20 MHz channel. We carefully examine the reasons behind this observation by conducting both physical layer (PHY) as well as higher layer experiments. Our key findings from the experiments are summarized as follows.

**Wide channels without interference:** First, with a fixed transmission power ($T_x$), there is about a 3dB decrease in the signal strength per subcarrier when WC is employed. Thus for a fixed $T_x$, the Bit Error Rate (BER) and the Packet Error Rate (PER) with WC is always greater than or equal to that without WC. Therefore, the throughput observed with WC is almost always "less than double" of that without WC [1]. Second, using a higher number of subcarriers with WC increases the likelihood of experiencing errors. We find that links of poor quality (i.e., low SINR) are the ones that are most affected by these factors; the throughput on such links with WC is worse than without WC. More importantly, as discussed later, the existence of a single low-SINR client in a cell (that uses WC) can degrade the long-term throughput of the entire cell due to the 802.11 performance

[1] In an ideal setting, one would expect that doubling the channel width would yield twice the throughput.

anomaly [16]. One might argue that these adverse affects of WC can be alleviated by using a higher $T_x$ at the APs. However, note that $T_x$ cannot be increased beyond a specified maximum value; this value is the same for both 20 and 40 MHz channels as mandated by the 802.11n standard [13]. In addition, increasing $T_x$ may project additional interference on other links if not properly executed.

**Wide channels and interference:** The use of WC projects interference over a larger spectral width (40 MHz as opposed to 20 MHz). Thus, the wider channels need to be carefully assigned to cells in a dense enterprise WLAN setting. Moreover, note that due to the 3 dB reduction in the per-carrier signal strength, transmissions with the wider bands are more susceptible to interference.

The above observations suggest that WLANs should employ WC with care. In this chapter, we detail the design of ACORN, an $\underline{\text{A}}$uto-$\underline{\text{CO}}$nfigu$\underline{\text{R}}$ation framework for enterprise 802.11$\underline{\text{N}}$ WLANs. ACORN jointly performs the functions of channel allocation and user association. As discussed above, the existence of poor quality clients impacts the cell-wide performance; therefore, intelligent user association is critical in facilitating the throughput gains from WC. In brief with ACORN, clients make association decisions not in a selfish / greedy manner, but by considering the impact of their decision on cell throughput. APs make decisions on whether or not to use WC by considering factors such as interference, towards achieving network-wide performance gains. To our best knowledge, ACORN is not only the first system to address the application of WC in enterprise WLANs but also the first system to consider the use of two distinct channel widths while performing joint channel allocation and user association.

The main contributions of our work are the following:

- We conduct extensive PHY layer experiments using WARP [17] and higher layer experiments on our 802.11n testbed, to validate that the performance can indeed degrade in settings without interference if WC is used.

- Leveraging the insights from our experiments, we design ACORN - an auto-configuration system for 802.11n WLANs. ACORN jointly executes user association and channel allocation with WC.

- We show that the channel allocation module of ACORN achieves a theoretical worst case approximation ratio of $O(\frac{1}{\Delta+1})$ as compared to an optimal algorithm, where $\Delta$ is the maximum node degree in the network (the problem is shown to be NP-complete).

- We implement ACORN on our 802.11n testbed and show that (i) it provides significant performance benefits (ranging from *1.5x* up to *6x*) over the state-of-the-art schemes designed for legacy 802.11 (which typically employ bands of a single width) and (ii) in practice, ACORN's channel allocation performs better than the theoretical worst case approximation.

- We evaluate ACORN with both synthetic UDP / TCP traffic as well as realistic HTTP workloads. We show that ACORN improves performance for all the considered traffic types.

The rest of the chapter is organized as follows. In Section 2.2 we provide brief background on the 802.11n PHY and discuss related work. Section 2.3 describes our PHY and higher layer measurements that provide an understanding of the behavior with WC. We present the design of, and analyze the algorithms included in ACORN in Section 2.4. Section 2.5 describes the system implementation and our experimental evaluation. We discuss ACORN's applicability to other deployments in Section 2.6. Our conclusions form Section 2.7.

## 2.2 Background and Related Studies

In this section, we first provide brief background on 802.11n. We then describe related previous work.

### 2.2.1 802.11n Background

802.11n technology offers significant improvements over the legacy 802.11a / b / g protocol family. At the PHY layer, 802.11n utilizes MIMO communication with OFDM. Two modes of MIMO operation are feasible with 802.11n: **(i)** *Spatial Division Multiplexing (SDM)*, which transmits multiple independent data streams and thus achieves higher data rates and **(ii)** *Space Time Block Coding*

*(STBC)*, which targets higher reliability and range by transmitting a redundant copy of the signal. Typically, vendors implement rate adaptation algorithms with 802.11n, which choose the mode of MIMO operation based on the link quality and other parameters. The MIMO operation together with the WC feature contributes to the throughput benefits offered by 802.11n.

### 2.2.2 Related Work

**802.11n Experimental Studies:** In [14], Shrivastava *et al.* identify that 802.11n throughput performance is limited by the CSMA / CA access policy inherited from the 802.11 legacy protocol family. In addition, they provide insights on increased interference due to WC, using a previously proposed model [18]. They recommend using WC on the 5 GHz band (in contrast to 2.4 GHz) since it offers more orthogonal channels and hence less potential interference. However, their conclusions cannot explain why an isolated link does not always enjoy a higher throughput with WC, as compared to the case without WC, which we show to be the case in practice. Visoottiviseth *et al.* [15] compare the performance between commodity 802.11g and 802.11n devices. However, this work does not examine WC in depth.

The above studies have either reported that WC decreased the throughput due to increased levels of interference [14] or that WC increased the throughput when used for a single link in isolation [14, 15]. On the contrary, we experimentally show that WC can degrade the performance even for a single link, in addition to the degradation for multiple interfering links.

**Orthogonal Frequency Division Multiplexing (OFDM):** OFDM divides the allocated spectrum into smaller tones (subcarriers) that are orthogonal to each other. Each subcarrier carries data at lower rates; however together they maintain the total data rate. OFDM systems have the advantage of better coping with narrowband interference and fading [19]. There are many studies that try to improve OFDM performance. For example, Rahul *et al.* [20] implement a downlink OFDM PHY for WLANs that performs rate adaptation per subcarrier. Subcarrier power allocation is examined in [21]. In this work, we are not interested in changing the PHY layer functions of 802.11n OFDM; we examine the effects of the PHY layer (with WC) on the higher layers and propose a solution towards improving performance without requiring significant changes at the core PHY / MAC functionality.

**Channel Width Adaptation:** Commodity 802.11 hardware vendors have also implemented a function that uses channels of varying width. These systems can flexibly operate on $5, 10, 20$ and $40$ MHz bandwidth [22, 23]. In [24], the authors propose a channel width adaptation algorithm that can dynamically choose a bandwidth to satisfy an optimization criterion. Although the authors discuss the potential usage of their algorithm in a WLAN context, the algorithm is designed assuming two communicating nodes and its applicability in enterprise WLANs is not studied. [25] proposes a spectrum allocation solution that leverages the flexible channel widths, by considering the load in each AP. It only addresses the assignment of different channels with a given client population (i.e., user association is not studied). In contrast, we show that user association and channel allocation (using WC in our case) is tightly coupled. We examine the impact of different channel widths on 802.11n enterprise WLAN performance by jointly considering user association and channel allocation.

**Resource Allocation:** Resource allocation in WLANs usually refers to power / rate control and channel selection. User association is very tightly related with these functions; therefore, in many cases they are studied together.

Static channel allocation is shown to have associated fairness issues and thus, Mishra *et al.* [26] propose a dynamic algorithm based on frequency hopping. In [27], the same authors formulate the channel assignment problem as a graph coloring problem and solve it. Kumar *et al.* [28] formulate a utility optimization problem that accounts for user association. All of the above efforts however, study the problems of channel selection or user association independently. In [29] and [1] the problem of jointly performing frequency selection and user association is studied. Mishra *et al.* [29] provide a centralized approach, while Kauffman *et al.* [1] provide a distributed solution based on the Gibbs sampler. In [3], the authors study the interaction between channel allocation, power control and user association.

All of the above (independent or joint) network optimization solutions are designed for legacy 802.11 systems that use single channel widths. The complexities associated with WC render the above solutions inapplicable to 802.11n. To the best of our knowledge, we are the first to study the joint problem of channel selection and user association in 802.11n WLANs, considering two distinct channel widths.

## 2.3 Wide Channels are not the Panacea

In this section, we first analyze the behavior of WC at the PHY layer. Later, we examine the impact of these PHY observations on the higher layers.

### 2.3.1 Wide channels micro-effects

802.11n leverages the OFDM-based PHY to implement WC. OFDM is inherited from legacy 802.11 systems. 802.11a / g modulate 52 subcarriers in an OFDM symbol of $4\mu$sec duration. The OFDM subcarriers together form a 20 MHz channel (or band). Four subcarriers are for pilot tones; thus, 48 subcarriers effectively carry data in an OFDM symbol. With 802.11n, the number of modulated subcarriers is increased to 56; 52 subcarriers carry data and four are for pilot tones. This results in a maximum nominal bit rate of 65 Mbps for a single data stream [2]. WC utilizes two adjacent channels simultaneously and employs 114 sub-carriers (108 data and 6 pilot) over a total of 40 MHz bandwidth. As one might expect, the nominal bit rates with 40 MHz are slightly higher than double of their 20 MHz counterparts for the same modulation. However, there is an important factor that can negatively impact the achievable throughput in a 40 MHz channel; the SNR experienced is lower with WC than when a 20 MHz band is used. To understand why this is the case, let us first look at the impact of increasing the number of subcarriers on (a) the thermal noise and (b) the energy per subcarrier.

[2]This assumes a $800n$sec guard interval (GI). The option for a shorter GI ($400n$sec) is also available in 802.11n; this reduces the symbol duration to $3.6\mu$sec and further increases the data rate. For more details see [13] and [19].

*Impact of WC on thermal noise:* The total thermal noise in a 40 MHz channel is higher than that in a 20 MHz channel. The Wi-Fi noise floor, $N$, can be calculated as follows [30]:

$$N(in\ dBm) = -174 + 10 \cdot log(B) \tag{2.1}$$

where $B$ is the bandwidth of the channel in Hz. It is easy to see that the noise in a 40 MHz channel is about 3 dBm ($10log2$) higher as compared to a 20 MHz channel. If one assumes that the noise is uniformly distributed across the subcarriers, the noise per subcarrier can be expected to remain almost the same for both the 20 MHz and 40 MHz channels, and in theory there is just a 4% reduction.

*Impact of WC on subcarrier energy:* The 802.11n standard [13] mandates the use of the same maximum transmit power with and without WC. In an OFDM system, the transmit energy is uniformly distributed across the subcarriers. Since WC uses 114 subcarriers and the total transmit power remains the same, the energy per subcarrier is theoretically reduced by 48% (approximately halved) as compared to that of 20 MHz bands. Expressed in dB, this translates to about a 3 dB reduction in the energy per sub-carrier. This in turn, can have a negative impact on the performance.

Considering the two factors together, we postulate that:

- For the same transmission power ($T_x$), the SNR of a signal is reduced by 3dB with WC. A reduction similar to this postulated value (52%), is seen in the energy per subcarrier in our experiments (discussed below); the noise per subcarrier remains almost the same as predicted theoretically (4% reduction). Thus, **the SNR per subcarrier is halved**; combined with the increased probability of error due to the larger number of subcarriers (each subcarrier experiences a different fade), a BER increase is expected.

- In order to achieve the same SNR on a link, with both 20 and 40 MHz channels, one needs to increase the transmission power in the latter case. However, this might not be possible given power budget constraints; in addition, 802.11n dictates the use of the same maximum power with and without WC.

- One might expect these factors to primarily affect poor quality links since these links are the ones that are most susceptible to the lowered SNR due to the use of WC.

With additive white Gaussian noise (AWGN), one can deduce the above observations from Shannon's theorem on the capacity of wireless channels [30]. The capacity $C$ of an AWGN channel (in bits per second) with a bandwidth $B$ (in Hz) is given by:

$$C = B \cdot log(1 + SNR) \tag{2.2}$$

One can see that for low SNR values, the logarithmic term dominates. Thus, if increasing B decreases SNR (as is the case when we transition from a 20 MHz channel to a 40 MHz channel), there may be regimes where the capacity decreases. The validity of these theoretical assessments in real settings has to be investigated, since the noise may not be AWGN in practice. We next present our experiments for this purpose.

**Experimental validation:** We use the WarpLab framework with WARP [17] hardware to implement a basic OFDM system. We generate a random bitstream and modulate it using DQPSK. The Inverse Fast Fourier Transform (IFFT) is applied on the modulated I - Q (In-phase and Quadrature) samples. A cyclic prefix is then added. A Barker sequence is later prepended to facilitate the symbol detection at the receiver. These samples are transmitted over the air using 2 x 2 STBC (Space Time Block Codes with two antennas - Alamouti) [31]; we use the STBC mode of transmission since on poor quality links, the auto-rate function of our 802.11n cards induces operations in this mode. At the receiver side, the preamble sequence is detected and stripped. The cyclic prefix is removed and the remaining samples are fed into a Fast Fourier Transform (FFT) module. After demodulating the samples, the receiver obtains the bitstream. We implement the WC functionality by appropriately changing the subcarrier mappings, sampling rate and using a 128-point FFT (as opposed to a 64-point FFT with a 20MHz channel).

First, we investigate the spectral characteristics of the transmitted waveform. We obtain the power spectral density (PSD) of the transmitted signals. The same power $T_x$, is used for both 20

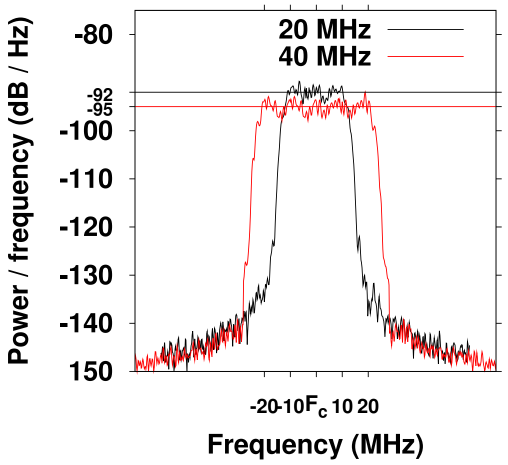
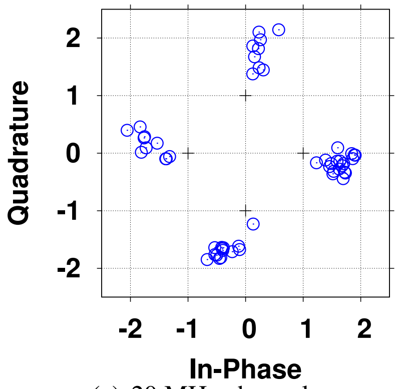
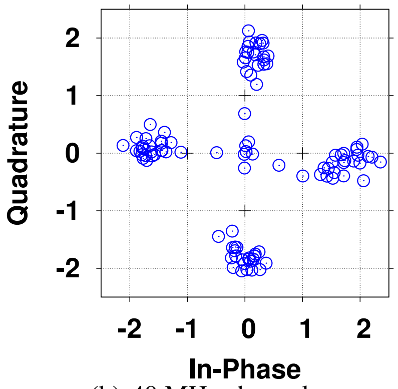


Figure 2.1: PSD estimate with different channel widths. WC results in a 3 dB reduction on the signal strength per sub-carrier.



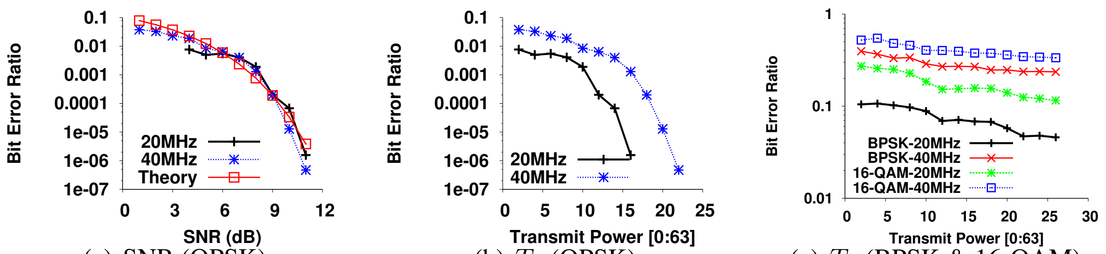
(a) 20 MHz channel



(b) 40 MHz channel

Figure 2.2: Received constellations with different channel widths. Although one can transmit more bits in a symbol with WC, the reception is more likely to be distorted (on the right) as compared to the case without WC (on the left).

and 40 MHz channels. PSD reflects the distribution of the energy with regards to the frequency content of the signal. Figure 2.1 depicts the PSD estimate with 56 and 114 subcarriers. It is evident that there is a 3dB reduction (from -92 dB to -95 dB) in the energy per subcarrier when we double the channel width. Note that in 802.11n, the central frequency $F_c$ is not the same for 20 and 40 MHz channels (we deliberately plot the PSD using the same $F_c$ for both bandwidths in Figure 2.1). In practice, when two 20 MHz channels (centered at $F_{c1}$ and $F_{c2}$ where $c1 < c2$) are merged together, the resulting channel is centered at $F_{c1} + 10$ [32].



(a) SNR (QPSK) (b) $T_x$ (QPSK) (c) $T_x$ (BPSK & 16-QAM)

Figure 2.3: BER for QPSK modulation with respect to SNR (a) and with respect to $T_x$ (b). For a given $T_x$, WC results in a higher BER (b and c).

To elucidate the effects of WC on a received signal, we present a typical sample from the received QPSK constellations for both 20 and 40 MHz channels in Figure 2.2. With 20 MHz the received symbols are mostly clustered around the actual transmitted symbol on the I - Q plane. With WC, there is a higher uncertainty due to the lowered energy per subcarrier. The signals are more vulnerable to fading and more likely to be erroneously decoded. The higher baud error rate (error rate of QPSK symbols) results in a higher BER.

To look at the implications of our last observation, we measure the Bit Error Rate (BER) with 20 and 40 MHz channels. We use the OFDM reference design from [17]. On top of the OFDM PHY, we use the BERMAC implementation [17] to transmit packets and calculate the BER for various settings. We use a custom Java application that transmits back to back packets to an Ethernet switch using the *jpcap* API [33]. The BERMAC implementation loads specific buffers of the WARP boards (on both the transmitter and the receiver boards) with the packet transmitted by the Java application; thus, the receiving node knows the *correct* payload contents for the BER calculation. In our experiments, we transmit a total of 9000 packets with a packet size of 1500 bytes and collect the BER statistics from the receiving board. We calculate the BERs using our WARP boards with 20 and 40 MHz channels, with QPSK. Note here that these are uncoded BERs (no forward error correction (FEC) employed at the PHY). The BER results with respect to the measured SNR are presented in Figure 2.3 (a). As one can expect, for a fixed SNR, the BER does not depend on the channel width. We also plot on the same figure the theoretical bit error rates for the considered system from [30]; the theoretical BER formula depends only on the SNR per subcarrier and not on the bandwidth. We observe that the experimental curves fit well with the theoretical plots [3]. In particular, the coefficient of determination [34] is $0.8$ and $0.89$ for 20 and 40 MHz, respectively. Figure $2.3$ (b) presents the same set of BER measurements, but with respect to $T_x$. We notice that the wider channel exhibits a higher number of bits in error for a given $T_x$, thus corroborating our intuition (discussed earlier). We observe a similar BER increase with other modulations (see Fig. 2.3 (c)).

### 2.3.2 Effects of WC on higher layers

The PHY layer performance is not always directly exported to the higher layers (due to mechanisms such as FEC). Therefore, the performance degradation in terms of BER with a 40 MHz channel for a fixed transmission power, may or may not be reflected in the performance observed at the higher layers. To understand the effect at the packet level, we look at PER (Packet Error Rate). A small increase in the raw uncoded BER (when using a 40 MHz channel) might result in no change in the PER on a commercial coded system such as 802.11n. If so, the throughput enjoyed at the application layer is practically doubled (ignoring MAC overhead). On the contrary, if the PER is also increased, this will result in less than a two-fold increase in the throughput or in some cases, could even result in a throughput reduction. Assuming the throughput $T$ with transmission rate $R$ is roughly $T = (1 - PER) \cdot R$, the throughput with a 40 MHz channel will be lower than that with 20 MHz if the following holds:

$$\sigma = \frac{1 - PER_{20}}{1 - PER_{40}} > \frac{R_{40}}{R_{20}} \approx 2, \tag{2.3}$$

where $PER_x$ and $R_x$ are the PER and the nominal transmission rate with the $x$ MHz channels. Here, $\sigma$ is simply the ratio of packet delivery probabilities achieved without and with WC.

[3] We leverage this later in designing ACORN in Section 2.4.

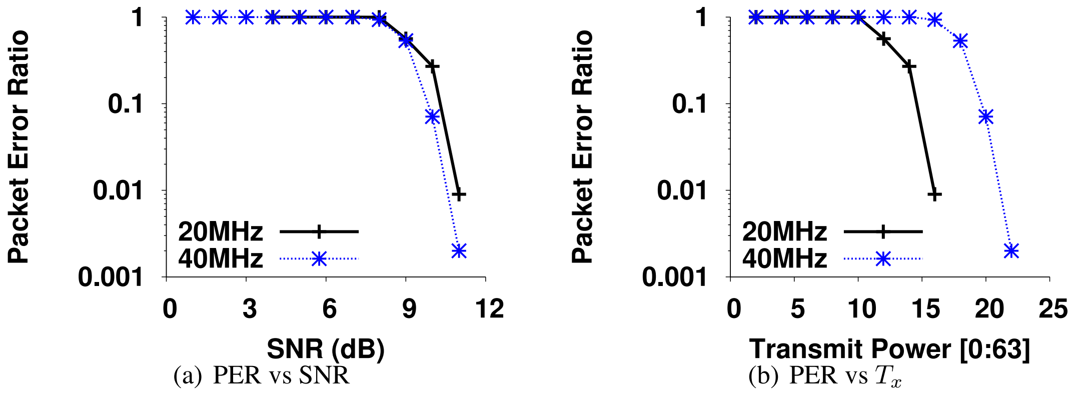


Figure 2.4: Uncoded PER for QPSK modulation. As expected, the higher BER with WC results in a higher ratio of packet losses at the MAC.

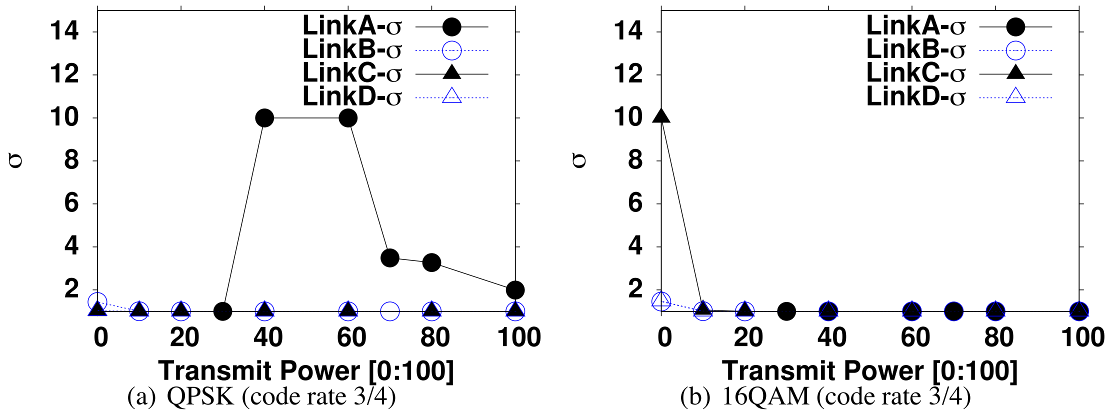
| $mod^{cod}$ | $QPSK^{3/4}$ | $16QAM^{3/4}$ | $64QAM^{3/4}$ | $64QAM^{5/6}$ |
|---|---|---|---|---|
| $\sigma \geq 2$ | -7dB | 3dB | 5dB | 8dB |
| $\sigma < 2$ | -4dB | 5dB | 7dB | 11dB |

Table 2.1: Experimental transition table for $\sigma$ values.

**PER performance:** Using the previous experimental setup and results, we first obtain the uncoded PER. Figure 2.4(a) presents the PER with respect to the SNR. As discussed, for a given SNR the BER does not depend on the channel width; thus, the uncoded PER is similar for the 20 and 40 MHz channels *for the same SNR*. However, for the same $T_x$, the PER with WC is much higher as compared to that without the feature (as seen in Figure 2.4(b)). Recall that with a 40 MHz channel, for the same $T_x$, the SNR per subcarrier is halved and this contributes to the performance degradation.

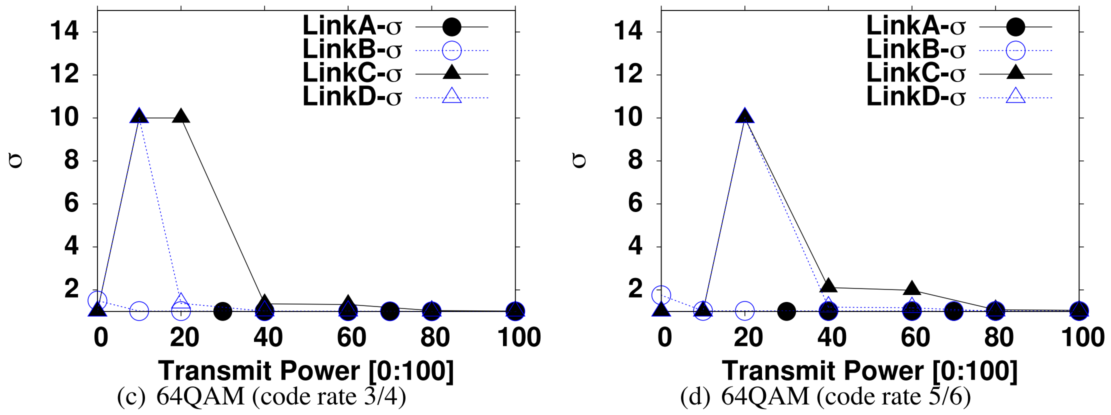
**Experiments with commodity 802.11n cards:** To examine the performance with commodity systems employing FEC, we conduct experiments on our 802.11n testbed. Our testbed consists of 18, 2 x 3 802.11n nodes, each equipped with a Ralink mini-PCI wireless card and three 5-dBi omnidirectional antennas. The testbed contains both indoor and outdoor links [35]. We use the 5GHz band for our experiments, avoiding interference from other WLANs in the 2.4GHz band.

With our initial 802.11n experiments, we examine the performance of the PER with a coded PHY layer. We tune the transmission power on our links in order to account for a large set of SNR



(a) QPSK (code rate 3/4)

(b) 16QAM (code rate 3/4)



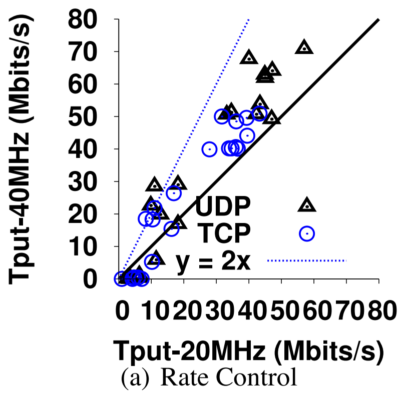
(c) 64QAM (code rate 3/4)

(d) 64QAM (code rate 5/6)

Figure 2.5: $\sigma$-values for different links, $T_x$, modulations (mod) and code rates (cod). For a given link, WC is beneficial ($\sigma < 2$) only beyond a certain power level. For lower power levels (lower SNR), WC hurts performance ($\sigma \geq 2$).

values, and we measure the PERs with both 20 and 40 MHz channels. In Figure 2.5, we plot the $\sigma$ values (recall Eq. 2.3) for various modulation schemes and code rates for four representative links (we omit BPSK since it exhibits a similar performance as QPSK). Whenever $\sigma$ is larger than 2, the throughput achieved with WC will be lower than that with a 20 MHz channel (Eq. 2.3). Table 2.1 presents the observed SNR values ($\gamma$) when we have a *crossover* value of $\sigma = 2$. The common trend identified among all the cases where $\sigma \geq 2$ is that this degradation in performance is observed for a certain range of transmission powers. This region maps to a 2 - 3 dB difference in SNR [4]. More specifically, for low $T_x$, the PER for both the 20 and 40 MHz channels is similar (and close to 1), resulting in $\sigma \approx 1$. As we increase the power, the PER with a 20 MHz channel

[4]When $\sigma$ is $> 10$, we cap its value at 10 for better visualization.



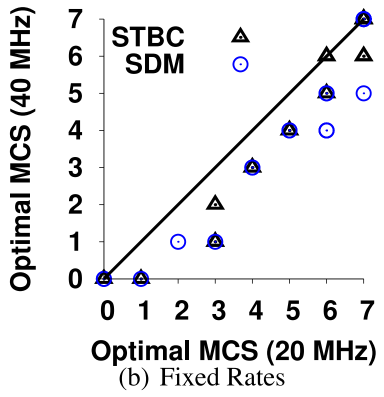


Figure 2.6: Throughput experiments (a) with rate control and (b) with fixed rates. Optimal MCS (b) denotes the MCS giving the highest UDP throughput for a link. WC primarily hurts performance for links with low SNRs even in the presence of rate control and FEC.

drops faster (with respect to $T_x$), since the SNR at the receiver is 3dB higher as compared to the WC case. Thus $(1 - PER_{20})$ increases faster than $(1 - PER_{40})$ and the ratio $\sigma$ can indeed assume values larger than 2; however, as we keep increasing $T_x$, the SNR with 40 MHz also increases and therefore, the PER performance with the two cases become similar to each other again (almost no packet losses) and $\sigma \approx 1$. Furthermore for a given link, the SNR value $\gamma$, at which we begin to see a PER decrease with 20 MHz (which results in $\sigma \geq 2$), is higher as the modulation becomes more aggressive (Table 2.1). The reason for this is that with more aggressive modulations there is a higher SNR requirement to correctly receive packets.

Note that, for some robust links (e.g., link B in Fig. 2.5) the PER is extremely low for both the 20 and 40 MHz channels and here WC will provide significant benefits in terms of throughput. For poorer links, the difference in the PER performance can be significant and inequality 2.3 might be satisfied. In such cases, a 20 MHz channel is preferable.

**Throughput performance:** From the perspective of the end-user, the achievable throughput at the application layer is what is important. To examine the application layer performance, we measure the achievable throughput with and without WC on our 802.11n testbed. Note that, so far we have examined the impact of WC using a fixed modulation. However in practice, rate adaptation mechanisms can potentially cope with the reduced SNR due to WC by adaptively using a more robust Modulation Coding Scheme (MCS [13]). For this reason, we use the rate control algorithm of our wireless cards to assess the performance in the presence of rate adaptation and FEC. This proprietary algorithm not only adjusts the rates in response to packet successes / failures but also picks the best mode of operation (SDM or STBC) based on the channel quality. We consider both UDP and TCP traffic generated by the *iperf* tool. The maximum transmission power is used and we consider all of our links (24 in total) to capture a wide variety of link qualities. Figure 2.6 (a) depicts the results. We see that in about 20% of our experiments, the use of the conventional 20 MHz channel yields higher throughput. What is more interesting is that the majority of these cases are clustered in the low throughput regimes. In addition, the throughput gains are very significant in these regimes (the points in the lower left corner of Fig. 2.6(a)); for example, 20 MHz channels achieve 5 to 6 Mbps while 40 MHz channels yield almost 0 Mbps throughput. The SNR values observed during these experimental trials were less than 6dB, which conforms with our previous BER / PER observations in Table 2.1. We observe that the rate adaptation mechanism does indeed lower the MCS to the most robust BPSK modulation. However, this is not sufficient to restore the throughput with WC for links with very low SNRs. In addition, approximately 30% of the TCP experiments yield better performance with 20 MHz as compared to only 10% with UDP. TCP is more sensitive to packet losses and as a result even a small PER increase due to WC can significantly degrade performance.

To understand the above observations, we next experiment with fixed transmission rates. For every link on our testbed, we find through exhaustive search, the MCS which gives the highest UDP throughput with and without WC, considering both modes of 802.11n (SDM and STBC). The results from Figure 2.6 (b) imply that the optimal modulation scheme with 40 MHz is almost always less 'aggressive' (i.e., smaller MCS) as compared with the one with 20 MHz. This results in less than a $2x$ throughput increase with WC as compared to 20 MHz channels. Figure 2.6 (a) verifies this, since the vast majority of the points lie on the right side of the line $y = 2x$. This is again an artifact of the increased BER and PER with WC.

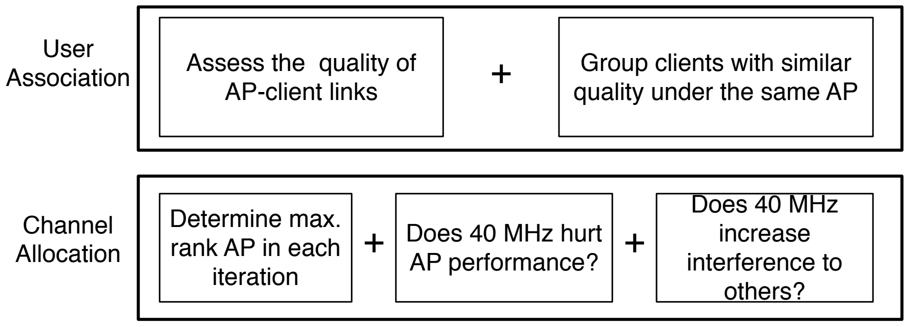

User Association
Assess the quality of AP-client links
+
Group clients with similar quality under the same AP
Channel Allocation
Determine max. rank AP in each iteration
+
Does 40 MHz hurt AP performance?
+
Does 40 MHz increase interference to others?

Figure 2.7: High level functions of ACORN.

To summarize, our experimental study shows that WC does not always increase throughput for links in isolation (such adverse effects of WC have not been reported in prior studies). If one were to also consider the increased interference from wider channels, it becomes evident that channel assignment algorithms should be carefully designed for 802.11n WLANs. In addition, user association is even more critical in the case of 802.11n than in legacy 802.11 systems; a poor client might hurt the throughput of the other clients of a cell that utilizes WC. Next, we describe the design of our 802.11n auto-configuration framework ACORN, which accounts for all of the above factors.

## 2.4 Harvesting ACORN

ACORN is designed based on the understanding developed with the experiments described in the previous section. It consists of two modules: (a) a user association module and (b) a channel assignment module that accounts for WC. The operations of ACORN, are briefly depicted in Figure 2.7. In short, the functions of the two modules are the following:

a) The **user association module** tries to group users (or clients) of similar link qualities within the same cell. The basis for this assignment is that the presence of a few poor quality users in a cell can degrade the performance drastically with 40 MHz bands. In such cases, even the good clients suffer due to the performance anomaly with 802.11 [16]. In brief, the distributed coordination function (DCF) used with 802.11 ensures equal long-term medium access opportunities. Since poor clients occupy the channel for longer periods, the good clients are hurt as well. This effect would cause an AP with an associated poor client, to suffer from degraded throughput if it were to apply WC. If instead it does not apply WC, the potential throughput gains for clients with high SNR links are lost. To address this, ACORN tries to ensure that each cell either has (preferably) all users with high quality links, or contains larger numbers of users with poor link qualities. In the former case, the cell can apply WC and in the latter it would simply use a 20 MHz channel.

b) The **WC module** exploits the application of the user association module. It assigns 40 MHz channels to those APs that achieve the highest improvements in throughput (clearly these are the APs that contain the most clients with good quality links). 20 MHz channels are assigned to APs that either suffer degradations in throughput with 40 MHz channels (due to the presence of poor clients) or do not achieve significant gains with WC.

Note that many of the previously proposed WLAN configuration schemes minimize the total transmission delay [1, 3]; this achieves fairness among the users. However, our objective is to ***maximize the total network throughput***. In other words, we tradeoff some level of fairness for significant gains in the total network-wide throughput. This is the current trend in many commercial platforms (e.g., 3G / CDMA), which typically employ schedulers that give priority to high quality links (e.g., PF scheduler) [36, 37]; they maximize the total network throughput, allowing for some hit in terms of fairness across the users. In addition, many studies align with this direction (for example [38] and [39]). We assume saturated downlink traffic for analytical tractability of ACORN's decisions. Most of the previous work also relies on this assumption [3, 40]. However, we show experimentally that ACORN helps even with unsaturated loads (e.g., TCP).

### 2.4.1 User Association

A newly arriving client $u$ usually has a set $A_u$, of serving APs to choose from. In order to pick the *best* AP for association, an objective must be in place. With ACORN, the decision is based on the pairing that achieves the maximum aggregate network throughput.

**Gathering information for user association decisions:** To achieve our goal, $u$ needs to know for every AP $i \in A_u$ the per client throughput of $i$, with and without $u$ associated with it ($X^i_{w,u}$

and $X^i_{wo,u}$, respectively). The client computes these values by obtaining a modified beacon. This beacon includes the number of clients associated with the AP (including $u$) $K_i$, the transmission delays per client $d^i_{cl}$, the aggregate transmission delay $ATD_i$ of the AP and the channel access time $M_i$ of the AP (if there is fully saturated traffic and no contention, $M_i = 1$). Client $u$ calculates the above quantities as (i) $X^i_{w,u} = \frac{M_i}{ATD_i}$ and (ii) $X^i_{wo,u} = \frac{M_i}{ATD_i - d^i_u}$ [1].

In order for the AP to be able to include these information in the beacon, the user has first to associate with it. Other, more simplistic approaches for AP selection, do not require prior client association with the APs. For example, affiliation decisions that are based on the received signal strength (RSS) of the beacons, do not require each user to associate with the APs in range first. However, it is shown that cross layer information is important for user affiliation to deliver good performance [40]; for instance simply looking at the RSS can lead to configurations with a few overloaded APs and other underloaded APs. In order to obtain the required information accurately, a user needs to associate with the AP and exchange traffic. Thus, we implement a similar approach to the one proposed in [1] [3] to obtain the information required by our algorithm. Note however that with ACORN the decision on which AP to affiliate with, is different.

**Associating with an AP:** Based on the information gathered from all the APs in range, $u$ picks the AP $i^* \in A_u$, which maximizes the following utility function with respect to $i$:

$$U_{asoc}(u,i) = K_i \cdot X^i_{w,u} + \sum_{j \in A_u, j \neq i} (K_j - 1) \cdot X^j_{wo,u} \tag{2.4}$$

The first term on the right hand side of Eq. 2.4 is the total throughput of the AP with which, $u$ associates. The second term is equal to the total throughput achieved by the other APs in the range of $u$. Note here that $K_j$ was defined as the number of clients associated with AP $j$, including client $u$. When a new client joins the cell, there may be a reduction in throughput due to the increased transmission delay. The goal is to minimize this reduction and preferably maintain the throughput at the level prior to the client's arrival. Eq. 2.4 minimizes the impact of a poor client $v$, in the network. $v$ affiliates with an AP serving similar quality clients, since this will minimize the total network throughput degradation (i.e., maximize the total network throughput) that could result from the 802.11 performance anomaly [16]. Clients with high quality links do not significantly affect the throughputs due to the anomaly when they associate with their best APs. The pseudocode for user association is given in Algorithm 1.

```
1: Input: $K_i, M_i, ATD_i, \forall i \in A_u$ and $d_v^i\ \forall i'$s clients
2: Output: AP $i$ that client $u$ associates with
3: for $i \in A_u$ do
4:     $X_{w,u}^i = \frac{M_i}{ATD_i}$
5:     $X_{wo,u}^i = \frac{M_i}{ATD_i - d_u^i}$
6:     $U_{asoc}(u,i) = K_i \cdot X_{w,u}^i + \sum_{j \in A_u, j \neq i} (K_j - 1) \cdot X_{wo,u}^j$
7: end for
8: return $i^* : U_{asoc}(u,i^*) \geq U_{asoc}(u,i), \forall i \in A_u$
```

**Algorithm 1**: User Association Algorithm

## 2.4.2 Wide Channels Decision/Allocation

Next, we describe the channel allocation module of ACORN. Table 2.2 enlists the notation used.

**Problem Formulation:** The channel allocation problem is cast as a graph coloring problem [27]. We apply the idea of the *interference graph (IG)* (as in [26, 27]). The set $V$ of vertices of the interference graph $G(V, E)$ are the APs. An edge $e_{ij} \in E$, if APs $i$ and $j$ interfere with each other. In the classic graph coloring problem [41], the objective is to assign colors from a given set of colors to the vertices of the IG, such that no adjacent vertices have the same color. Note here that, the notion of *color conflicts* differs in our case due to WC. For instance, let us assume colors $c_i$ and $c_j$, corresponding to channels $f_i$ and $f_j$ (20 MHz channels). Then the *composite color* $\{c_i, c_j\}$ corresponds to the 40 MHz channel derived from the combination of $f_i$ and $f_j$. With this set up, the basic colors $c_i$ and $c_j$ do not conflict; however, each of them conflicts with the composite color $\{c_i, c_j\}$. In the rest of the chapter, we will refer to both basic and composite colors simply as colors. We relax the constraint of the above graph coloring problem, since our objective is to assign colors to the vertices (i.e., channels to the APs), so as to maximize the total network throughput. Formally,

|  |  |
|---|---|
| $V$ | Set of Access Points |
| $Ch$ | Set of available 20 and 40 MHz channels |
| $F : V \to Ch$ | Channel assignment mapping |
| $f_i$ | channel assigned at AP $i$ |
| $X_i$ | Throughput of AP $i$ |

Table 2.2: Notations for channel allocation algorithm.

we seek to:

$$\max_F \quad Y = \sum_{i \in V} X_i(F) \tag{2.5}$$

**NP-completeness:** The graph coloring problem with the above objective is NP-complete. The classic, NP-complete, **decision graph coloring problem** [41], tries to answer the following: "Given a graph $G(V, E)$ and $k$ colors, can we color the vertices $V$ such that $\forall e_{ij} \in E \to f_i \neq f_j$?", where $f_i$ is the channel assigned to AP $i$.

The total aggregate network throughput $Y$, is upper bounded by $Y^* = \sum_{i \in V} X_i^{isol}$, where $X_i^{isol} = \max\{X_i^{isol-20}, X_i^{isol-40}\}$ is the maximum possible throughput for AP $i$ in an interference-free setting, achieved with either a 20 or a 40 MHz channel. Let $F^{'}$ be a solution to our problem, yielding a total aggregate network throughput of $Y^{'}$. The solution is optimal **iff** $Y^{'} = Y^* \Leftrightarrow G$ has a $k$-coloring. Thus, our problem is NP-complete.

**Our approach:** We design an algorithm that allows APs to decide upon the channel(s) to use. Later, we compute the approximation ratio of our algorithm relative to the optimal. Initially, all APs are assigned either a 20 MHz or a 40 MHz channel at random. The algorithm is iterative and executed with a periodicity of $T$. In every iteration, the AP that can provide the maximum increase in the aggregate throughput by switching channels, is allowed to switch. The algorithm terminates, after a number of iterations $K$, either when no further improvement can be provided or when the improvements provided are very small. A pseudocode for our algorithm is given in Algorithm 2. $Y_x^W$ represents the long-term aggregate network throughput achieved at period $W$ and after $x$ iterations of the algorithm.

```
1: **Input:** $Y_k^{T-1}, F^{T-1}$
2: **Output:** $F^T$
3: $Y_0^T = Y_k^{T-1}, k = 0$
4: **repeat**
5:   $label\ (1)$
6:   $AP = V, AP^{'} = \emptyset$
7:   **for** $i \in V$ **do**
8:     $k = k + 1$
9:     **for** $c \in Ch$ **do**
10:      $Tmp_i(c) = \sum_{a \in V} X_a(F^T_{j \in AP^{'}}, f_i = c, F^{T-1}_{j \in AP})$
11:    **end for**
12:    $pick\ \ c_i^* : Tmp_i(c_i^*) \geq Tmp_i(c), \forall c \in Ch$
13:    $rank_i = Tmp_i(c_i^*) - Y_{k-1}$
14:    **if** $\max_{i \in V} rank_i < 0$ **then**
15:      **if** $|AP^{'}| \leq 1$ **then**
16:        ***return*** $F = (F^{T-1}_{j \in V}, F^T_{j \in V})$
17:      **else**
18:        $GOTO\ (1)$
19:      **end if**
20:    **else**
21:      $\text{“}winner\text{”}\ is\ AP\ m : rank_m \geq rank_n, \forall n \in V$
22:      $f_m^T = c_i^*, AP = AP/\{m\}, AP^{'} = AP^{'} \bigcup \{m\}$
23:      $Y_k^T = \sum_{a \in V} X_a(F^{T-1}_{j \in V}, F^T_{j \in V})$
24:    **end if**
25:  **end for**
26:  **if** $AP \neq \emptyset$ **then**
27:    $GOTO\ \ (1)$
28:  **end if**
29: **until** $Y_k^T < \epsilon \cdot Y_{k-1}^T$
```

**Algorithm 2**: Wide Channels Selection Algorithm

In each iteration of the algorithm, the APs that have still not chosen new channel(s) (i.e., members of the set $AP$), estimate the aggregate throughput achieved with every possible channel, assuming that other APs keep their current allocation (line 10).

**Estimating throughput:** In order to estimate the throughput on a new channel, an AP needs to take into account (i) the number of APs already residing on this new channel and (ii) the quality of the links to its clients on the channel. The first requirement is possible either with help from an administrative authority or the Inter Access Point Protocol (IAPP) [42]. For the second requirement, we assume that the link quality on the different channels (of the same width) is not significantly different. Later in this section, we provide measurements that validate this assumption in indoor slowly varying settings, typical for enterprise 802.11n WLANs. However, this assumption does not hold when channels are of different width. In other words, the channel qualities to the clients may change if a channel of different width is used. To map the measured results from a 20 MHz channel on to a 40 MHz channel (or vice versa), we leverage the understanding obtained from our PHY layer measurements in Section 2.3.

We estimate the link quality on a channel of different width as follows. The input is the SNR at the current width. When we change the width (from 20 to 40 MHz and vice versa), there is a 3dB change in the SNR; this processing is performed by a *SNR calibration* module in our estimator. Using this calibrated SNR value, a *BER estimation* module calculates the theoretical coded BER (from [30]). Recall, from our measurements that one can expect a reasonable match between the values computed with the theoretical formulas in [30] and the experimentally observed BER.

Finally, using BER we estimate PER. We use the commonly used assumption (e.g., in [43]) of independent, uniformly distributed bit errors within a packet and compute PER as:

$$PER = 1 - (1 - BER)^L. \tag{2.6}$$

*Note here that ACORN does not require the exact BER or PER values; it only needs a coarse estimate of the link quality i.e., a reasonable classification of good and poor links.*

Once the performance on each of the available channels is estimated, the AP that can provide the maximum increase in the aggregate throughput by switching channels does so. This approach in essence, mimics the gradient descent algorithm; it greedily seeks to find the point that exhibits the maximum increase in the value of the objective function (the throughput).

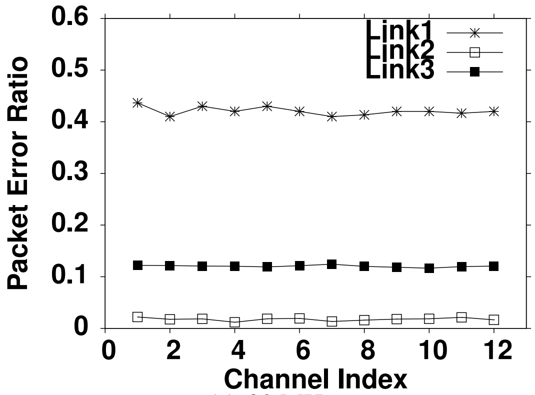
The same procedure is repeated considering the APs that have not had an opportunity to switch. When no improvement is possible or the improvement is incremental ($\epsilon$ - line 29), the algorithm stops. In our implementation, $\epsilon = 1.05$ (i.e., if there is a 5% or less increase in the total aggregate throughput as compared to the previous iteration, the algorithm stops).

**Approximation Ratio:** Gradient-based optimization can be trapped in a local extremum. Given also that the channel allocation problem is NP-complete, we are interested in finding the worst case approximation ratio of our algorithm.

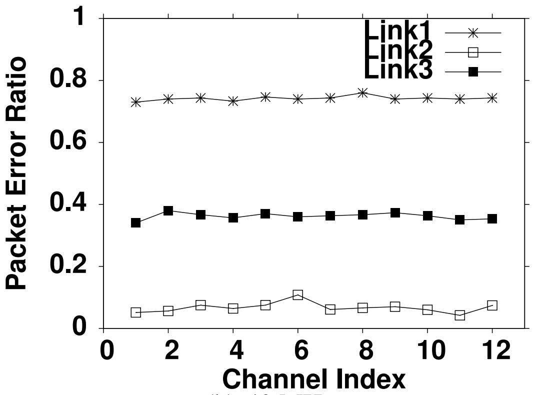
The maximum possible aggregate throughput is obtained when all APs operate in an interference-free setting, and is equal to: $Y^* = \sum_{i \in V} X_i^{isol}$ as mentioned before. The worst local extremum where our algorithm can be trapped, is the one in which every AP uses the same (20 or 40 MHz) channel. In other words, nodes are not just assigned conflicting colors, but are assigned exactly the same color. This is because, if they are assigned different conflicting colors (a composite color and a basic color) the achieved throughput will be higher (this is easy to verify). In this case, the throughput of every AP $u$ will be reduced by a factor of $\frac{1}{deg_u+1}$, assuming fully saturated traffic, where $deg_u$ is the degree of node $u$[5]. Consequently the long-term total network throughput will be $Y_{worst} = \sum_{i \in V} \frac{1}{deg_i + 1} \cdot X_i^{isol} \leq \frac{1}{\Delta + 1} \cdot \sum_{i \in V} X_i^{isol}$, where $\Delta$ is the maximum node degree in the network. Thus, our algorithm has a worst case approximation ratio of $O(\frac{1}{\Delta+1})$. As shown in Section 2.5, in practice, the channel allocation scheme performs much better.

**Link quality on different channels:** We assume that the quality of a link does not exhibit significant variations in terms of PER on different channels of the same width. To verify this, we conduct the following experiment. For all the links, we measure back-to-back, the PER on the dif-

[5]Note here that, the graph we consider is the interference graph of the network [27] with respect to the APs (vertices of the graph). Two APs interfere with each other either if they directly compete for the medium or if either competes with at least one of the other AP's clients.



(a) 20 MHz



(b) 40 MHz

Figure 2.8: Link quality on different channels (without interference) does not exhibit significant differences.

ferent channels, using the maximum modulation of 64-QAM (MCS = 15 [13]). Figure 2.8 presents the results from three representative links. Our measurements demonstrate that, indeed, the variations across the different channels are negligible (for both 20 and 40 MHz channels), making our assumption realistic. Note that since the results in Fig. 2.8 are with the most aggressive modulation (i.e., 64-QAM), the PER variations are even smaller for other less aggressive modulations. There are studies (e.g., [20]), that have reported variations in the link quality with different channels. However, these studies are on single antenna systems. In contrast, in our experiments, we utilize the MIMO PHY of 802.11n. The use of MIMO makes the performance stable and decreases variations across the different channels (arising primarily due to fading in single antenna systems). ACORN can easily be modified, such that each AP scans (one at a time) all the available channels and gets more accurate information regarding the link quality to its clients. However, this would add more complexity and increase the convergence time of the system.

**Periodicity of our algorithm:** The periodicity $T$ with which we apply channel allocation needs to be carefully chosen. If we apply it too often, then the hit in the throughput could be significant due to the overhead. If on the other hand, we activate channel allocation too infrequently, the topology might have significantly changed in the interim and the current allocation might provide poor performance.

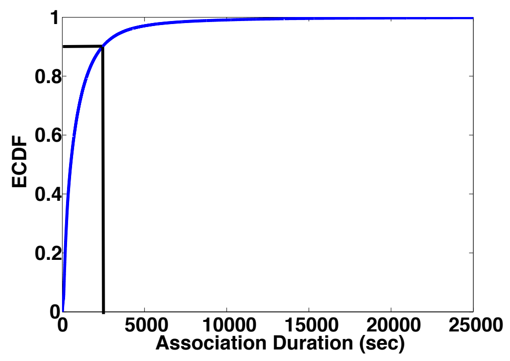


Figure 2.9: CDF of user association durations.

In order to assess this tradeoff, we use data collected from 206 different (commercial) APs, in a time period spanning more than 3 years from the CRAWDAD repository [44]. In particular, we extract the association duration of each user. Figure 2.9 depicts the CDF of the association duration. More than 90% of the associations last less than 40 minutes and the median is approximately 31 minutes. Based on these data, we run our channel allocation algorithm every 30 minutes.

### 2.4.3 ACORN and Other Rate Adaptation Algorithms

In our discussion, we have so far assumed that if an AP is configured with a particular channel width (whether 20 MHz or 40 MHz), that channel width is used for all of AP's clients. Our results in Section 2.5 also reflect this assumption since the Ralink RT2880 APs use a fixed channel width for all of their clients. However, other commercial 802.11n drivers such as *ath9k* [45] can adjust the channel width while performing rate adaptation for each AP-client link. Such drivers typically fall back to robust modulations in combination with a 20 MHz channel width for low-SNR clients, while continuing to use higher modulations with 40 MHz channel width for high-SNR clients. Next, we discuss how ACORN's functionality can be easily modified to account for such rate adaptation capabilities.

**User Association:** Recall that when a new user $u$ arrives, it considers joining an AP $i$ based on the per client throughput of $i$, with and without $u$ associated with it ($X^i_{w,u}$ and $X^i_{wo,u}$, respectively).

Previously, we assumed that when computing $X^i_{w,u}$, $u$ factors in the delay of its link to AP $i$ based on $i$'s current channel width and the resulting modulation (MCS) for the link between $u$ and $i$. For the case where, AP $i$ can adjust its channel width for each of its clients, $u$ simply needs to determine the best channel width (and the resulting MCS) for its link to $i$, when it computes $X^i_{w,u}$. Note that despite this small modification, the user association objective still remains the same as expressed in Eq. 2.4.

**Channel Allocation:** ACORN allocates additional channels (e.g., to use WC) to APs that provide the maximum increase in throughput due to the allocation of that particular channel, in an iterative fashion. With this, it prevents APs with low-SNR clients from using 40 MHz channels. Note that this is subject to the assumption that the channel width use by an AP is fixed (and the same) for all of its clients. However, even if an AP is capable of adjusting its channel width on a per client basis, ACORN (modified as discussed in the previous paragraph) will ensure that the 40 MHz channels are assigned to APs that have the maximum benefits in terms of throughput (predominantly clients with good channel quality). For APs with a large number of clients with poor link qualities, ACORN may assign 20 MHz channels. In dense deployments of WLANs (as is the case today), this is especially useful in opportunistically reusing the available spectrum to the maximal extent. In cells that have been assigned a 40 MHz of channel width, the corresponding APs can easily use a smaller 20 MHz bandwidth to serve the few clients that have poor channel qualities.

## 2.5 Evaluating ACORN

In this section, we detail the implementation of ACORN and its evaluation using our 802.11n WLAN testbed.

### 2.5.1 Implementation Details

We implement our algorithms using the Click modular router (v1.6) [46]. We implement a user-level utility that runs both at the APs and the clients. We keep track of the SNR, the nominal rate and the association time per client by using the functions implemented in our card's driver. The delay for each client is calculated and broadcast in a beacon as described in Section 2.4, along with the $M_a$ values (defined in Section 2.4.1), the number of clients and the aggregate transmission delay of an AP. The client receives the beacons from every AP in range and makes appropriate association decisions.

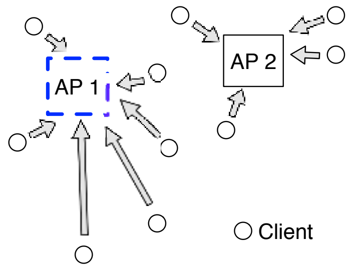
**Calculating per client transmission delay and $M_a$:** These metrics are not directly available since our hardware does not provide access to firmware. We implement a module where every AP calculates the delay of a client by utilizing our PER estimation procedure and the nominal rate for the client. We *estimate* $M_a$ for an AP $a$ by $\frac{1}{|con_a|+1}$ where $con_a$ denotes the set of neighboring APs that reside on the same channel as AP $a$. This estimation has very high accuracy when these APs can hear each other under saturated traffic. Accurate management and configuration of WLANs is of most interest in these regimes i.e., in dense deployments and with heavy loads.

### 2.5.2 Experimental Evaluations

**Comparison with legacy 802.11 WLAN configuration systems:** We start by randomly assigning initial channels to APs from the 5GHz band. Clients are then randomly activated one by one. Each client performs user association (i) as per **Algorithm 1** or, (ii) the algorithm described in [1]. The APs then perform channel selection either as per **Algorithm 2** or a modified version of [1]. We modify the frequency selection algorithm in [1] to implement a greedy strategy where APs aggressively use the (single width) 40 MHz channels[6]. Specifically, they scan 40 MHz channels and select the one that minimizes the total noise and interference. We simply refer to this scheme as "[1]".

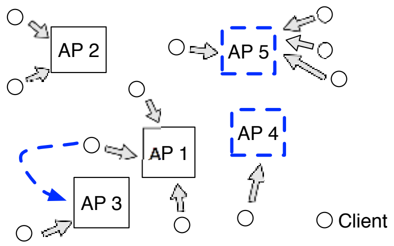
**ACORN significantly improves per-cell throughput in interference free settings:** In these experiments, we have saturated downlink UDP traffic (generated using *iperf*) from each AP to its clients. We evaluate ACORN on many different WLAN topologies. We employ all the twelve 20

[6]As one might expect, simply using the algorithm with 20 MHz channels results in lower rates and correspondingly lower throughput. For clarity and ease of discussion, we omit these results here.



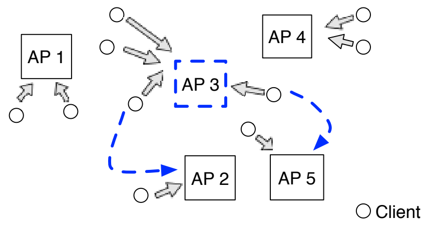
| | AP 1 | AP 2 | Total (Mbps) |
|---|---|---|---|
| ACORN (Tput) | 16.03 | 52.9 | 68.93 |
| [18] (Tput) | 3.15 | 53.1 | 56.25 |

(a) Topology 1



| | AP 1 | AP 2 | AP 3 | AP 4 | AP 5 | Total (Mbps) |
|---|---|---|---|---|---|---|
| ACORN (Tput) | 56.6 | 53.5 | 56.3 | 3.78 | 15.9 | 186.08 |
| [18] (Tput) | 55.8 | 54.1 | 20.4 | 0.56 | 6.35 | 137.21 |

(b) Topology 2



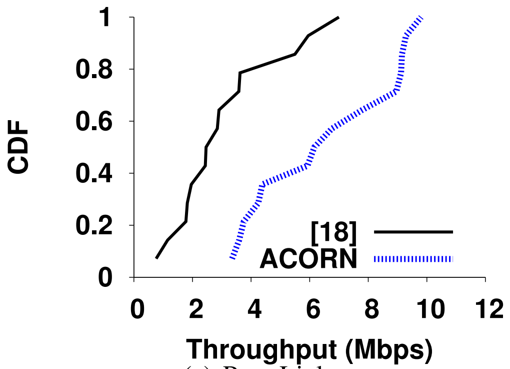
| | AP 1 | AP 2 | AP 3 | AP 4 | AP 5 | Total (Mbps) |
|---|---|---|---|---|---|---|
| ACORN (Tput) | 64.2 | 52.5 | 11.4 | 57 | 44.9 | 230 |
| [18] (Tput) | 64.3 | 35.3 | 1.09 | 57.7 | 42.4 | 200.79 |

(c) Topology 3

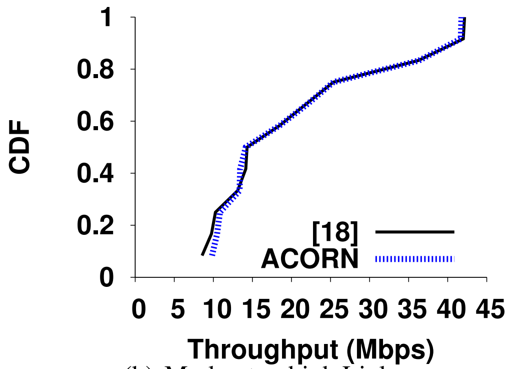
Figure 2.10: ACORN can provide throughput gains in interference-free deployments. Solid arrows depict the association relationship between clients and APs. Dashed APs use 20 MHz with ACORN and 40 MHz otherwise. Dashed arrows depict the different user affiliation decisions taken by [1].

MHz channels available in the 5GHz band with both ACORN and [1]. Figure 2.10 quantifies our per-AP throughput observations with a few sample topologies (also depicted). Topology 1 consists of 2 APs. This is a sparse WLAN where clients are connected with poor quality links with AP1 and good clients are associated with AP 2. We find that the user association with both ACORN and with [1] are identical. However, the use of the 20 MHz band provides a significant increase (*4x* throughput increase) for AP 1 because of its low-SNR client links. In fact, with the 3 dB reduction in SNR, we observe that the poor clients can hardly communicate with the AP when it uses WC with [1].

Topology 2 includes 5 APs. We observe that with ACORN, when poor clients associate with an AP, the AP uses a 20 MHz channel. The same APs apply WC greedily with [1]. The presence of poor clients reduces the cell throughput of the corresponding AP. These effects are seen with AP 4 and AP 5 of the topology and for these, ACORN provides significant throughput improvements (*6x* for AP 4 and *1.5x* for AP 5). We also observe that ACORN results in different user associations for APs 1 and 3; as discussed, ACORN tries to *group* clients with similar link qualities in the same cell. In contrast, [1] evenly divides the clients between these APs regardless of the specific client link qualities. Due to this, AP 3 achieves a higher throughput (1.8x) with ACORN since it serves



(a) Poor Links

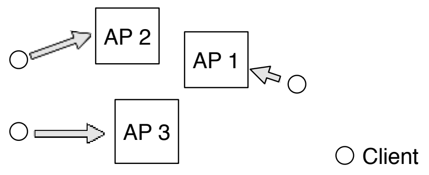


(b) Moderate - high Links

Figure 2.11: ACORN improves the median HTTP file transfer throughput by *1.5x* for poor quality links (left) while retaining the performance of good quality links (right).

only one good quality client. This results in more congestion at AP 1 with ACORN as compared to [1] since it has to serve more clients. However, interestingly, AP 1 can still achieve the *same total throughput with more clients*. We identify the reason behind this to be the following: since ACORN groups similar-quality clients in one cell, the aggregate throughput does not change despite the fact that *per-client* throughput is reduced; the performance anomaly of 802.11 does not take effect. Similar performance gains are also observed in Topology 3. Here AP 3 achieves a $\approx$*10x* increase when it avoids WC with ACORN. We again observe that ACORN chooses to group similar quality clients around AP 3, instead of balancing the load across the APs. As a result of this grouping, AP 2 with ACORN also achieves higher throughput since it serves a single good quality client. We observe a similar behavior with ACORN in a variety of other scenarios.

In addition to downlink UDP traffic, we evaluate ACORN against [1] with a HTTP workload. In these experiments, we configure the APs as HTTP servers hosting files. The clients download a file from their APs and we measure the throughput of the file transfer (using *wget* [47]). Fig. 2.11 plots the CDF of the file transfer throughputs of each AP. For clarity, we plot the results separately for two categories of links: a) links that achieve $< 8$ Mbps throughput with [1] (i.e., poor links) and b) links with $\geq 8$ Mbps throughput with [1] (i.e., moderate and high quality links). While maintaining a similar performance for cells with good client link qualities, ACORN provides significant gains



| X,Y,Z | 40,40,40 | 40,20,20 (ACORN) | 20,40,20 | 20,20,40 |
|---|---|---|---|---|
| Tput (Mbps) | 42.3 | 79.98 | 54.15 | 52.38 |

Figure 2.12: ACORN provides the highest throughput in settings with interference. X, Y and Z denote the channel widths (in MHz) used by APs 1, 2 and 3, respectively.

for cells with poor clients. For such clients, ACORN increases the median HTTP throughput by *1.5x*. These experiments demonstrate that the application throughput of clients also improves with ACORN for typical realistic traffic types.

**ACORN reduces interference:** In dense deployments where channel availability is limited, ACORN reduces interference at the neighboring APs and provides even higher improvements in throughput. To showcase this, we experiment with a representative scenario in Figure 2.12 where the number of APs is not small relative to the number of available channels (unlike in the previous experiments). We have 3 APs that contend for channel access and there are four 20 MHz channels made available. AP 1 serves a good quality client and APs 2 and 3 have poor clients associated with them. When all the APs aggressively use WC, they project interference on each other. In addition, APs 2 and 3 suffer because of the poor client links. Note that, with 4 channels, only one AP can use WC to achieve complete isolation. In such cases, it is essential for a channel allocation scheme to identify the best AP that should exclusively use WC. We observe that, ACORN identifies this AP and provides the highest throughput as compared to other possible allocations. It provides an almost *2x* improvement over the scheme that aggressively allows WC operations at every AP; the aggressive allocation causes increased interference and thus, lower throughput.

**Comparison with random manual configurations:** We compare ACORN against a large set of manual configurations in terms of assigned channels and user associations. The purpose of this experiment is to see the effectiveness of ACORN in yielding the highest network throughput.

| | ACORN | Random Configurations (Descending order) |
|---|---|---|
| Network Tput (UDP - iperf) | 259.2 | 201.63, 193.1, 188.56, 187.6, 184.62, 183.39, 169.62, 163.32, 160.47, 159.35 |
| Network Tput (TCP - iperf) | 178.93 | 161.7, 155.77, 134.78, 133.4, 130.64, 114.1, 109.4, 106.6, 103.41, 102.3 |

Table 2.3: ACORN achieves the highest network throughput (in Mbps) against 10 best (out of 50) manual configurations.

In this experiment (different from previous experiments), we measure the performance in terms of both UDP and TCP throughputs (both generated by *iperf*). The experiments evaluate ACORN's performance with unsaturated TCP traffic; since our analysis assumes saturated traffic, we wish to examine if ACORN works well with unsaturated traffic as well. Note that, we are mainly interested in scenarios that fully load the network (saturated conditions), since achieving balance and efficacy is most important in these regimes. However, these experiments (together with the HTTP workload evaluation) demonstrate the applicability of ACORN in more generic settings.

For a randomly picked topology, we first run ACORN and obtain the total network throughput. Next, we configure APs with random channels (both 20 and 40 MHz) and let each client associate with one of the APs in range with equal probability. We repeat this experiment for 50 different configurations and take the 10 best configurations. Table 2.3 tabulates our results. We observe that ACORN configures the network in a way that achieves the highest possible throughput as compared to what is achieved with these random configurations. We wish to point out that ACORN provides gains with TCP since congestion can still occur at shorter time scales; at such times, the use of ACORN provides benefits. The decisions relating to WC deliver higher performance since they are based on client link qualities and independent of the type of traffic. Although the set of random configurations is by no means exhaustive, the experiments do demonstrate the efficacy of ACORN in terms of yielding high throughputs.

**How close to the optimal is ACORN channel allocation in practice?:** Next, we perform experiments to examine the approximation ratios achieved by the ACORN channel allocation module in practice. We choose 3 APs that *compete* for channel access in each experiment (i.e., $\Delta = 2$); 9 such sets of different APs are considered. We then associate clients with each of these APs.

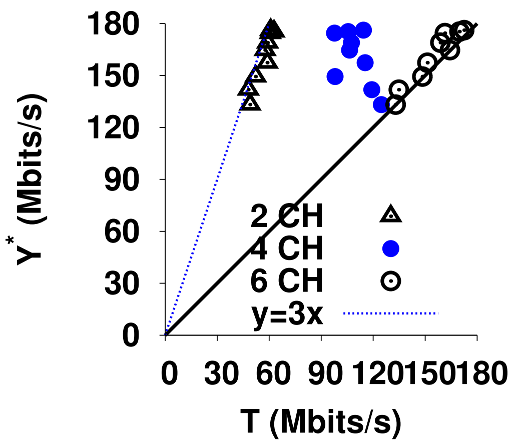


Figure 2.13: ACORN has approximation ratios better than $O(\frac{1}{\Delta+1})$ in practice.

We then run saturated UDP downlink traffic from each AP to its clients in *isolation* for both 20 MHz and 40 MHz channels. We calculate $Y^*$, which is the best possible aggregate throughput, as $\sum_{i=1,2,3} max(T^i_{20}, T^i_{40})$, where $T^i_x$ is the throughput obtained by AP $i$ using a channel width of $x$ MHz. Note here that this maximum is achieved when we completely isolate the 3 competing APs i.e., they do not contend with each other. Subsequently, we run the ACORN channel allocation algorithm with 2, 4 and 6 orthogonal channels made available. Note here that, 6 orthogonal channels are enough for all of the APs to simultaneously activate WC. Figure 2.13 depicts the total network throughput, $T$, obtained by ACORN in comparison with $Y^*$. Note that $Y^*$ computed as above, is a *loose* upper bound, since complete isolation of the 3 APs is not always possible with less than 6 orthogonal channels. With 2 channels, ACORN does not perform worse than what is theoretically predicted; the aggregate network throughput is $\frac{Y^*}{3}$, since the medium access is shared among the contending APs. In the case of 6 channels, ACORN can achieve $Y^*$, since channel allocation isolates every AP and configures the best channel width for each AP. We observe some cases where ACORN performs very close to the optimal (what is possible with 6 channels) even with only 4 channels. Examining the cases with care, we find that there is at least one AP $i$ such that $T^i_{20} > T^i_{40}$; ACORN accounts for this and configures the particular AP with a 20 MHz channel, leaving 3 channels for utilization to the other two APs.

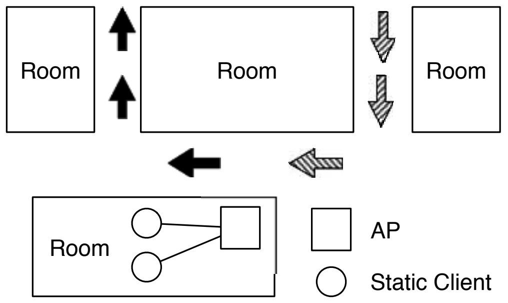
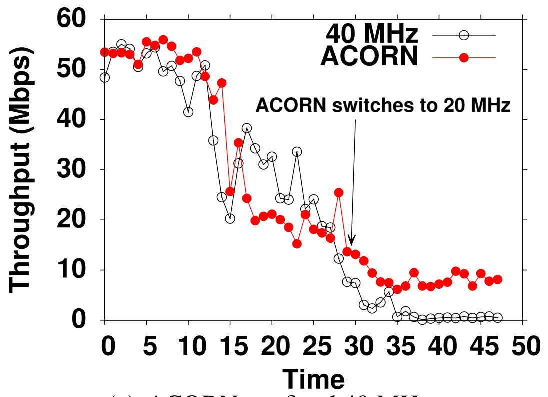
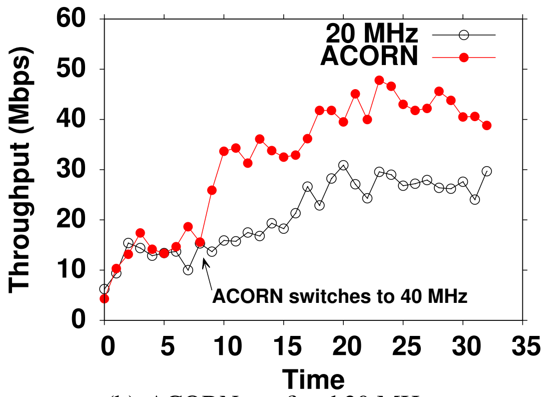


Figure 2.14: Trajectory of our mobile client.



(a) ACORN vs. fixed 40 MHz



(b) ACORN vs. fixed 20 MHz

Figure 2.15: ACORN tracks the link quality and selects the channel width that gives the higher throughput over fixed channel widths.

**Evaluating ACORN with mobility:** With ACORN, once an AP is assigned a 40 MHz channel, it can opportunistically use its allocated channels. In other words, the AP can opt out from using WC and only employ the 20 MHz channel (one of the two assigned). Since the other APs choose their frequencies based on the channels assigned to this particular AP, using either of the two 20 MHz channels will not change the interference on the neighboring APs. This mode of operation is particularly desirable in WLANs with mobile clients. Since an AP to client link quality can vary temporally, the AP can dynamically activate the desired width of operation (20 vs 40 MHz) with ACORN based on the measured link qualities of its clients. We experiment with a scenario that involves pedestrian mobility. We configure a laptop with the same card that we use in our testbed and use it as a mobile client. In this experiment, we have a single AP that has 2 static clients in addition to the laptop. First, we position the laptop close to the AP and start moving it *away* from the AP. Figure 2.14 depicts this trajectory with dark arrows.The AP transmits downlink UDP traffic to its clients and we record the aggregate throughput measured as a function of time. Figure 2.15(a) presents the time trace of the aggregate cell throughput during the mobile client's movement. We compare ACORN against a configuration that strictly uses a 40 MHz channel. We observe that ACORN uses the 40 MHz channel in the beginning and sustains this until the point where the link quality becomes poor for the mobile laptop (around 30 sec). From that point until the end of the experiment (the client stops at a location far from the AP), ACORN falls back to the 20 MHz mode and is able to sustain a cell throughput that is almost ten times that of a fixed 40 MHz channel. Note here that the poor quality link to the distant client affects the good clients as well due to the 802.11 performance anomaly [16]. In a similar experiment, we have our client moving *towards* the AP, the trajectory is depicted with striped arrows in Figure 2.14. We compare ACORN against a fixed 20 MHz configuration in Figure 2.15(b). We observe that ACORN uses the 20 MHz until it recognizes the improvement in link quality (SNR); at that time it switches to a 40 MHz channel (at around 10 sec) and is able to utilize the gains from WC.

## 2.6 Scope of ACORN

In our experiments, we evaluate ACORN in an enterprise setting where clients choose their AP from a set of candidate APs. In addition, user association and channel allocation are executed cooperatively so as to maximize the aggregate WLAN throughput. In residential settings, the clients typically have only one AP to choose from and the APs may not cooperate towards a common performance optimization. ACORN's benefits from user association will not be viable in such settings. However, ACORN can still be deployed with only the WC decision component. As shown with our experiments, ACORN provides significant gains from intelligent application of WC. Residential APs often have multiple clients due to the proliferation of mobile devices and wireless home entertainment systems. In such cases, it is important to prevent throughput degradation due to poor quality client links, by carefully applying WC. Thus, ACORN can also be applied to residential WLANs to provide potential throughput benefits. Stated otherwise, ACORN is modular in nature and can flexibly be deployed in various settings.

## 2.7 Conclusions

The wide channels (WC) feature allows 802.11n nodes to use a wider frequency band towards achieving high data rates. We show that applying WC has associated caveats and can degrade performance even in interference-free settings. We conduct extensive experiments to understand why this is the case. Based on the understanding gained, we design and implement ACORN, an auto-configuration framework for 802.11n WLANs. We show via extensive evaluations that ACORN provides significant per-AP throughput gains (as high as 6x) over what is achieved with prior approaches designed for legacy 802.11 systems that are based on using single channel widths. To the best of our knowledge, ACORN is not only the first system to consider the application of WC in 802.11n WLANs but also the first system to consider the use of two distinct channel widths while performing joint channel allocation and user association.

# Chapter 3

# Experimental Characterization of Interference in Multicell OFDMA Femtocell Systems

The increase in mobile Internet access is pushing broadband operators towards deploying smaller cells (*femtocells*) and sophisticated air interface technologies (Orthogonal Frequency Division Multiple Access or OFDMA). The expected high density of deployment and uncoordinated operations of femtocells however, make interference management both critical and extremely challenging. A significant challenge stems from the fact that femtocells have to use the same synchronous access technology as traditional macrocells. Given this, understanding the impact of the multitude of design choices (originally tailored to well-planned macrocellular networks) on interference management forms an essential first step towards designing efficient solutions for next-generation femtocell networks. This in turn is the focus of our work.

With the help of extensive measurements from our WiMAX femtocell testbed, we characterize the impact of various system design choices on interference in OFDMA femtocell deployments. The design choices that we examine are categorized across three broad dimensions: (i) *resource isolation* at the logical (MAC) level across femtocells, (ii) *resource mapping* via sub-channelization at the physical (PHY) level within each femtocell, and (iii) synchronization (or lack thereof) between interfering femtocells. Based on the insights from our measurements, we discuss several implications on the choice of system design features for femtocell operations. We also provide guidelines for efficient interference management in femtocell networks.

## 3.1 Introduction

The demand for higher data rates to cope with increased mobile data usage is driving broadband providers towards deploying smaller cells (called femtocells) with OFDMA [48]. Femtocells are much less expensive than their macrocell counterparts and operate at much lower powers [49]. They are installed in homes and enterprises, and use the same spectrum and access technology as macrocells, while connecting to the core network through cable or DSL backhaul.

Recent industry reports have shown that mobile access to the Internet is steadily increasing (expected to grow by a factor of 26 in 2015) and around 40% of mobile data originates indoors (i.e. homes and enterprises) [50]. Femtocells, being deployed indoors, present numerous advantages to mobile broadband customers and service providers. With femtocells, customer equipment (e.g. 4G phone, laptop) can save energy on the uplink and enjoy high throughput since it does not have to transmit to the macro base station that is typically miles away. The downlink throughput is also increased due to the short ranges of femtocells. In addition, the users can continuously experience the "5-bar" cellular coverage in areas where the macrocell signal is poor. Recent surveys have indicated that consumers find femtocells appealing due to above advantages [51]. This increases user satisfaction and reduces subscriber churn for service providers. In addition, the small size and low cost of femtocells offer cost-effective ways of providing coverage as opposed to deploying large and often expensive macrocells. The smaller cell sizes also increase the system capacity via spatial reuse.

While the above advantages motivate femtocells, they tend to get deployed in an unplanned manner by end users without direct coordination of the cellular operators, and hence they inevitably lead to equivalent unplanned interference which can potentially limit their performance. There are two potential interference scenarios for typical indoor femtocell deployments, (i) interference from / to macrocells and (ii) interference *between* femtocells. Although macro-femto interference has been studied (e.g. [52]), interference between femtocells has not received much attention before. If not properly understood, evaluated and accounted for, this can cause significant degradation in femtocell performance.

Resource management solutions to mitigate interference between femtocells cannot, however, be designed from scratch. These cells have to inter-operate with and use the same access technology as macrocells. There are three key aspects that make resource management both challenging and unique in femtocells: (i) Femtocell deployments are dense and uncoordinated unlike planned deployments of macrocells. Thus, unlike in macrocells where interference is localized at cell-edges, interference is less predictable and more pervasive across cells. (ii) Unlike in macrocells, it is harder to achieve synchronization across interfering femtocells (deployed by different users). The impact of this lack of synchronization on resource management is not well understood; one might however, expect that this would make resource management harder. (iii) Macrocell access technology dictates a *synchronous* channel access mechanism for femtocells (unlike *asynchronous* random access in WiFi). WiFi systems adopt carrier sensing that arbitrates access to the transmission medium. While this can help cope with some of the interference scenarios, a more widely-adopted approach is to tune interfering WiFi APs to orthogonal channels (of same bandwidth) in the unlicensed spectrum. Femtocells lack the carrier sensing feature and operate on a licensed spectral band (i.e. orthogonal channels are not available). Therefore, if interfering femtocells need to be tuned to orthogonal frequencies (for purposes of interference mitigation), they have to be assigned orthogonal sub-channels (*fragments* of the licensed spectrum). This reduces the bandwidth per femtocell and thus, the throughput. These factors necessitate the design of novel interference management solutions for femtocells. However prior to that, one has to understand the impact of various features of the macrocell access technology on interference among femtocells.

In this chapter, we present one of the first and most detailed measurement studies of OFDMA femtocell networks, performed using our WiMAX femtocell testbed. We provide an in-depth understanding of the impact of design choices made in the macrocellular context, on resource management in femtocells. These design choices are categorized along three broad dimensions. (i) Resource isolation across femtocells: isolation of transmissions between interfering cells can be either in the time or frequency domain. (ii) Resource mapping or sub-channelization in each femtocell: OFDMA sub-channels that provide the basic granularity of transmissions at the MAC layer, map onto a set of sub-carriers (tones) at the PHY layer. Varying the sub-carrier composition (grouping) of sub-channels using (a) contiguous or distributed grouping, and (b) different permutation sequences, can impact performance. (iii) Synchronization: lack of tight synchronization between femtocells can exacerbate interference given the rigid frame structure (designed for macrocells). Synchronization impacts both resource isolation and mapping.

Our study reveals several interesting (and sometimes surprising) effects. We carefully examine these and provide inferences that help understand the implications of the aforementioned design choices in the interference-rich femtocell environment. The study thus allows us to provide guidelines which would enable femtocells to effectively manage resources towards achieving both high per-cell and network-wide performance. Our key findings are as follows.

- Unlike in macrocell networks, isolation of transmission resources between interfering cells is critical towards alleviating pervasive interference that arises due to dense uncoordinated deployments.

- Surprisingly, resource isolation in the frequency domain increases the system capacity as compared to time domain resource isolation (reasons discussed later).

- A contiguous composition of subcarriers (as opposed to a distributed composition) at the PHY level reduces interference due to frequency offsets across cells. This can significantly aid performance.

- Permuting the subcarrier composition of sub-channels at the PHY level can provide only limited benefits. These benefits are due to the fact that different subcarriers are subject to different interference patterns.

- Frame-level synchronization between base-stations (BSs) yields predictable benefits from resource isolation. More importantly, there is still merit to isolating resources even in the absence of synchronization (discussed later).

The rest of the chapter is organized as follows. In §3.2, we provide brief background on WiMAX and OFDMA, and discuss related work. §3.3 describes our experimental platform and methodology. §3.4, 3.5 and 3.6 describe our measurements related to resource isolation, resource mapping and synchronization. The inferences that we draw from our measurements and guidelines for efficient femtocell network operations are in §3.7. In §3.8, we discuss how our work applies to OFDMA systems other than WiMAX. Our conclusions form §3.9.

## 3.2 Background and Related Work

### 3.2.1 Related Work

**Multicell OFDMA Systems:**While OFDMA broadband standards (WiMAX, LTE) have been drafted recently, related research has existed for quite some time. There are studies that address problems in multicell (e.g. [53, 54]) OFDMA systems. Several efforts have looked at the interference to cell-edge users in OFDMA macrocells. The *localized* interference coupled with *planned* cell layouts have aided various solutions (e.g. [55, 56]). Femtocells lack the desired features of localized interference, planned deployments and coordinated operations of macrocells. This necessitates novel resource management solutions. There have been some recent works [53, 6] that propose such solutions but are restricted to theory. In practice, OFDMA femtocells are under trial by several network operators and are yet to be deployed. To our best knowledge, there are no studies to date that shed light on real-life performance of OFDMA femtocells.

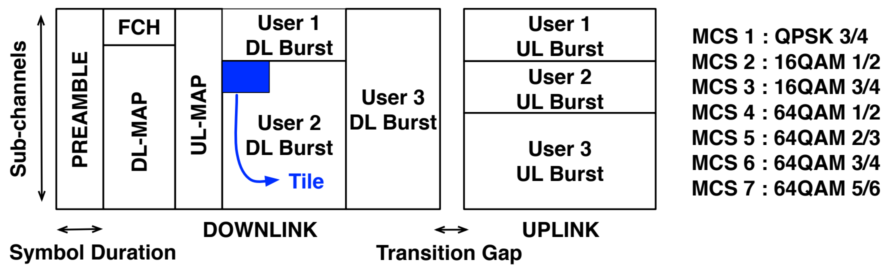


Figure 3.1: A simplified illustration of the WiMAX frame structure.

**WiFi and OFDMA:** Recent efforts show the benefits of OFDMA in the WiFi context by building practical systems that enable dynamic spectrum fragmentation [57, 58] and adaptive channel width [59]. [60] explores 802.11 with OFDMA. Since WiFi is based on asynchronous channel access, the main focus in all these efforts is enabling synchronization and alignment between a TX-RX pair (crucial for OFDMA). However, OFDMA is a mature technology where synchronization between a TX-RX pair is a standard feature. While the synchronization benefits are carried over to femtocells, this also limits their ability to manage interference effectively (i.e., lack of carrier sensing). This renders previous approaches addressing interference in WiFi (e.g. [2, 61]) inapplicable to OFDMA. Several design choices made for macrocells have to be revisited in a femtocell context.

### 3.2.2 WiMAX Preliminaries

While our study applies to OFDMA femtocells in general, our measurements are on a WiMAX (802.16e [62]) femtocell testbed. Hence, we overview of some of the relevant WiMAX components (more details in [62]).

**MAC Frame Structure:** With OFDMA, the spectrum is divided into multiple frequency tones (sub-carriers) and several sub-carriers are grouped to form a sub-channel. A WiMAX frame becomes a two-dimensional template that needs to be scheduled with data to multiple mobile stations (MSs) across both time (symbols) and frequency (sub-channels). The combination of a symbol and a sub-channel constitutes a ***tile***, which is the basic unit of resource allocation. Data to users are allocated as rectangular bursts of tiles in a frame.

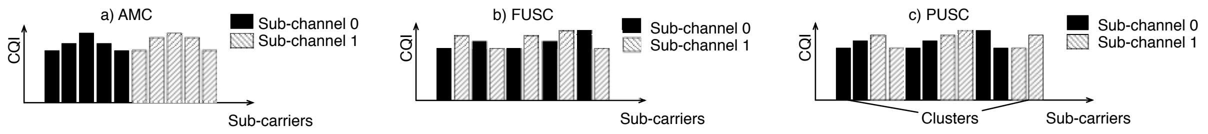


Figure 3.2: Illustration of sub-channel composition in WiMAX. Two (out of 30) sub-channels are shown for clarity.

In OFDMA macrocells, frames are synchronized both between the BS and MSs and across BSs. An example of a WiMAX frame is shown in Fig. 3.1; the transmissions from the BS to the MS (downlink) and those from the MS to its BS (uplink) are separated in time. The frame consists of the preamble, control and data payload. The preamble is used by the MS to lock on to the BS. The control consists of FCH (frame control header) and MAP. MAP conveys the location of the data burst for a MS in a frame. A BS schedules tiles both on the downlink and the uplink. The DL-MAP indicates the location of each burst in the frame, the MS it is intended for, and the modulation and coding scheme (MCS) in use. The UL-MAP indicates where the MS should place its data on the uplink frame. The different MCS levels in 802.16e are also shown in Fig. 3.1. While data bursts can be modulated using any MCS, the standard mandates using QPSK for the control part. In addition, the preamble is transmitted with a higher power compared to the other parts of the frame.

**PHY Resource Mapping:** While the MAC frame allocates resources at the granularity of tiles, transmitted bits map on to sub-carriers at the PHY. WiMAX offers options for the grouping of sub-carriers to form a sub-channel.

One aspect of grouping is related to the degree of contiguity in the sub-carriers. WiMAX provides three options here (see Fig. 3.2). ***FUSC (full usage of sub-carriers):*** sub-carriers composing a sub-channel are picked in a completely distributed manner from the spectrum; ***PUSC (partial usage of sub-carriers):*** sub-carriers are first grouped into clusters and distributed clusters are then grouped to form a sub-channel; ***AMC (adaptive modulation and coding):*** contiguous set of sub-carriers are grouped to form a sub-channel. AMC has perfect contiguity and hence retains the frequency selectivity of user channels. However, leveraging this diversity requires feedback of channel quality information (CQI) on all sub-channels from all users. FUSC loses frequency selectivity due to lack of contiguity, but requires only one CQI feedback (an average over the spectrum) for each user. PUSC strikes a good balance between frequency selectivity and overhead; it retains some contiguity via sub-carrier clustering and feeds back one CQI value similar to FUSC. PUSC is the default mode in 802.16e. The first generation of WiMAX devices implement PUSC and FUSC alone. Hence, we restrict our discussions to these modes.

Once the degree of contiguity is decided, the next step is to determine how the sub-carriers are picked. This is achieved by using pre-defined permutations in 802.16e. A total of 16 and 8 permutations are available with PUSC and FUSC, respectively. We defer a detailed discussion of these to §3.5.

### 3.2.3 Our work in perspective

Unlike in macrocells, interference is less predictable and more pervasive in dense, unplanned femtocell deployments. While interference has a direct impact on the decoding of the data bursts, it also significantly increases the vulnerability of the control part of the frame which is critical for decoding. To understand the impact of interference in femtocells, we perform extensive experiments on a WiMAX femtocell testbed. In particular, we seek the answers to the following:

- Does interference require resource isolation or can it be addressed by choosing the right MCS for each transmission (via link adaptation) ?
- If resource isolation is required, should it be achieved in the time or in the frequency domain?
- Is it sufficient to isolate resources for data bursts alone or is isolation also needed for the control part of a frame?
- Can resource mapping at the PHY aid resource isolation?
- What is the impact of the lack of synchronization across interfering BSs, on resource isolation and mapping?

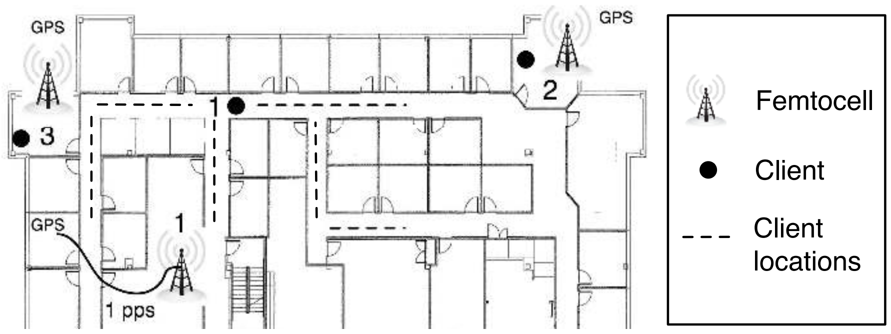


Figure 3.3: Topology of the WiMAX femtocell deployment. BS 1 is not placed near a window; therefore the 1 pps signal is provided with a 10 m. cable. Client $j \in$ cell $j, j \in \{1, 2, 3\}$

.

We believe that answering these will lead to guidelines for efficient interference management in femtocell networks.

## 3.3 System Description

**Experimental Setup:** Our testbed consists of three femtocells deployed in an indoor enterprise environment (Fig. 3.3) each with one client. We use PicoChip's [63] femtocells that run 802.16e (WiMAX). Our clients are black boxes that use commercial WiMAX cards (plugged into laptops) from Accton [64] [1]. Since WiMAX imposes a rigid frame structure (fixed locations for preamble and control), the measurements can be executed by mainly varying the positions of the data bursts. For this purpose, we modify the downlink scheduler code at the BS to perform measurements in a variety of scenarios. Note that any change in the BS scheduler (permitted by the standard [62]) is automatically conveyed to the clients via FCH and MAP. The cells operate on a 8.75 MHz bandwidth with the carrier frequency of 2.59 GHz. For this frequency, we obtained an experimental license from FCC to transmit WiMAX signals on the air.

**Synchronous Transmissions:** OFDMA uses synchronous channel access where a BS transmits a downlink frame *periodically* (5 ms for WiMAX). The clients align their receivers in time with the

[1]We use the terms client - MS and femtocell - BS interchangeably.

BS to receive these periodic transmissions. The failure of downlink frame delivery will cause a client to disassociate from the BS. If there is no data to be sent to the MS(s), the BS has to transmit *at least* the preamble and the control payload to keep the MS(s) in sync.

Although frames are synchronized between a client and its BS, they may not be synchronized across BSs. In other words, the start of a femtocell's downlink frame may not be aligned in time with another femtocell's downlink frame. We perform experiments with both synchronized and unsynchronized frames among the BSs. While each BS has its own internal clock, they are initially not synchronized with each other. To achieve synchronization across femtocells, we use external GPS modules (antennas placed close to a window) from TeraSync [65] to provide a 1 pps (pulse per second) signal to each BS, thereby aligning the start times of downlink frames. One particular challenge that arose in our efforts was that there was already an on-board clock oscillator; we needed to manually desolder the oscillator from the radio boards to use the external clock and the 1 pps signal from the GPS. Note that once synchronization is achieved, it will hold true for the uplink frames of BSs as well (given the fixed downlink frame size and transmit gap).

**Experimental Methodology:** We focus on cell 1 for our measurements. The other two BSs project interference (unless noted otherwise) on cell 1. Note that when BS 1 is transferring data to its client (client 1), even switching on other BSs (without any data transfer) will cause interference. This is because, every BS keeps transmitting the preamble and control information even when no clients are associated with it. However, when data transfer is also involved, the impact of interference is more severe. To understand these different aspects of interference, we pick the client in cell 1 and measure its throughput in the presence of either one or two other interfering cells, both with and without data transfers in the interfering cells (to clients 2 and 3). We consider downlink UDP traffic from the BSs to the clients generated by *iperf*. The traffic rate is set large enough to saturate the available tiles. A client's throughput is a function of the number of tiles in the allocated burst (symbols $*$ sub-channels) and the number of bits that the selected MCS encodes. Under these conditions, interference is projected by other BSs that transmit bursts (to their respective clients) using a set of tiles that collide with that of the BS - client pair under consideration.

**Generating Topologies:** OFDMA uses synchronous access as opposed to the asynchronous random access in WiFi systems. This fundamental difference reveals different dependencies among interfering entities in WiFi and OFDMA. In WiFi, interference can either hurt the reception at the client (due to collisions) or the transmission at the AP (due to carrier sensing). Since an OFDMA transmitter does not sense the medium, a nearby interferer does not preclude it from transmitting frames. Thus, interference in OFDMA solely hurts the client reception the extent of which is determined by parameters such as client location, propagation environment and transmission power of BSs. Therefore in measuring interference, we mainly vary the locations of clients (as opposed to moving the BSs) to generate a plurality of scenarios. Our deployment is a typical indoor environment with walls, doors, rooms and cubicles. Varying the locations of clients gives us a finer level of detail to cover a wide range of scenarios that include LOS and NLOS links. We call a given triplet $\{bs, cl, int\}$ a *topology* where $bs$ is the BS that the client is associated to, $cl$ is the location of the client and $int$ is the set of BSs that project interference on the client. Each measurement point corresponds to a topology and is obtained by running an experiment for 7 minutes, measuring the throughput and averaging it over several such runs.

**Link Adaptation:** The impact of interference on a client depends on the interference power received (i.e. location) and the MCS used for the transmission. The SNR required for correct decoding increases with higher MCS levels and thus, the impact of a given interference power is more severe with higher MCS. Note that femtocells are meant to be user-installed at homes and in enterprises. Hence, the baseline strategy is one where a femtocell operates on its entire spectrum, while performing link adaptation (MCS selection) for each of its clients. While link adaptation impacts client throughput in the presence of interference, current link adaptation solutions are sub-optimal. Therefore, we consider ideal link adaptation, whereby for each data point, we sequentially run the experiment over all MCS levels and record the one delivering the highest throughput for the given topology. Since we experiment in a slowly-varying static indoor environment, it is feasible to consider link adaptation as above.

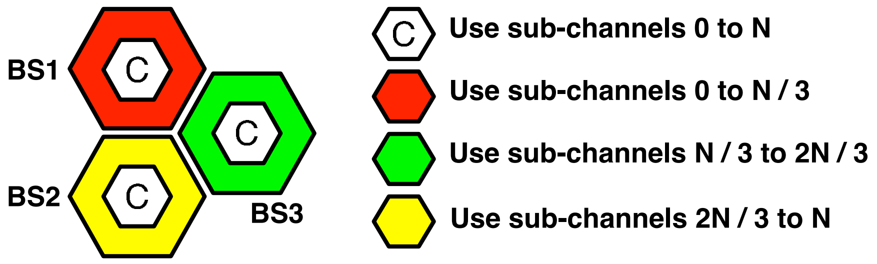

BS1
C
C
BS2
C
BS3
C
Use sub-channels 0 to N
Use sub-channels 0 to N / 3
Use sub-channels N / 3 to 2N / 3
Use sub-channels 2N / 3 to N

Figure 3.4: Hexagonal cell deployment. BSs transmit to center clients (C) using all sub-channels. Three separate orthogonal segments of sub-channels are used at cell edges.

Given that there are a multitude of design features that can have an impact on interference among femtocells, we categorize our study along the following three dimensions: (i) resource isolation, (ii) resource mapping, and (iii) synchronization. To isolate the benefits of resource isolation and mapping from the impact of synchronization, we consider the first two dimensions with synchronized femtocells.

**Scalability of Inferences:** Femtocells are constrained to inter-operate with the macrocell standards. Specifically, macrocells use 1:3 fractional frequency reuse (FFR) based on a hexagonal deployment as illustrated in Fig. 3.4. To support 1:3 FFR, the OFDMA (WiMAX) frame is segmented to form up to three non-overlapping sets of sub-channels. In macrocells, this segmentation is used for both the control and data parts of the frame resulting in perfect isolation of resources. This means at most two interferers can be accommodated on the same frame on orthogonal sub-channels. Thus, any study that considers segmentation for interference avoidance requires only two interferers to be considered. If there are more than two interferers, the interference effects can only be studied where some sub-channels are inevitably reused by multiple BSs for the control payload. With such reuse, interference on a BS to client link is quite disruptive even with two BSs causing interference (shown later); the effects of interference will only exacerbate with more interfering BSs and our inferences will still hold. Thus, we believe that our three cell system effectively captures all cases that arise in realistic femtocell deployments.

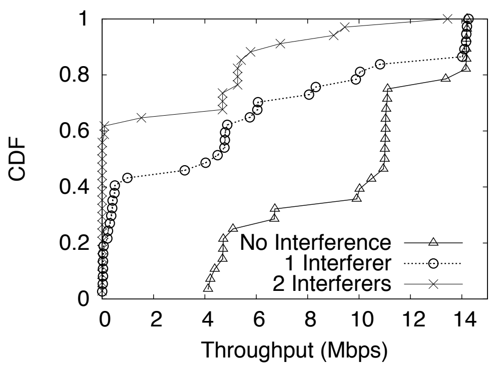


Figure 3.5: Prevalence of Interference

## 3.4 Resource Isolation

In this section, we examine the impact of various MAC level design choices on interference.

### 3.4.1 Prevalence of Interference

First we ask, *How prevalent is interference in femtocell deployments*? Client 1 is the desired client whose throughput is measured (see Fig. 3.3 and note that client $j \in$ cell $j$). While the locations of clients 2 and 3 are fixed (very close to their BSs), the location of client 1 is varied to generate a plurality of topologies. Either one or both interfering cells (cells 2 and 3) are activated and transfer data to their clients so as to interfere with client 1's reception. All cells use the available spectrum entirely. At each location, we first find the best MCS for client 1 without interference; this MCS is then fixed during interfering transmissions (no link adaptation). The reason for using this fixed rate is to see in what percentage of locations a given client is impacted by interference (where it could have achieved a high throughput in isolation). We consider cases with link adaptation later.

The CDF of the throughput at client 1 over numerous topologies is presented in Fig. 3.5. The throughput without interference is used as a benchmark and compared against that with one and two interfering cells. While the client sustains a median throughput of 11 Mbps with no interference, this decreases to 5 Mbps with one interfering cell, and is close to 0 Mbps with both interferers ($\approx 60\%$ of the locations have 0 Mbps). This clearly shows the pervasive impact of interference in indoor femtocell deployments. The maximum throughput in all cases is the same because these correspond to locations where client 1 is very close to its BS; here, it is effectively *shielded* from the interferers.

### 3.4.2 Resource Isolation vs. Link Adaptation

We now ask, *Does link adaptation by itself suffice in alleviating the impact of interference or is isolation of resources (tiles) needed?* Switching to a lower MCS via link adaptation can provide increased robustness to interference when the client's SNR is above the threshold required for that lower MCS level. If the received SNR is even lower than the most basic MCS requirement, this does not help. Isolating tiles (in conjunction with link adaptation) between cells helps alleviate the effects, but it comes at the cost of a reduced set of available tiles. In summary, the relative benefits of the two approaches depends on the severity of interference.

In a two-dimensional WiMAX frame (see Fig. 3.6), resource isolation can be either via (i) time-domain isolation or TDI (symbols) or via (ii) frequency-domain isolation or FDI (sub-channels). TDI isolates resources in time by offsetting the frames of multiple BSs and leaving empty (guard) tiles to prevent collisions. FDI for data (denoted FDIdata) allocates orthogonal sub-channels to multiple BSs only for data. The preamble and control parts use all available sub-channels. FDI for data and control (denoted FDIctrl) allocates orthogonal sub-channels for both the data and control parts. Note that with FDIctrl, due to the reduced number of sub-channels for the fixed-size control part, it takes a higher number of symbols to transmit this part. Thus, slightly fewer tiles are available for data bursts.

To understand the relative benefits of resource isolation and link adaptation, we experiment with two strategies: (i) two cells operating on all resources (baseline) and (ii) two cells operating on orthogonal sets, each containing half the sub-channels in a frame (FDIctrl). For both strategies, we measure the throughput at client 1 in the presence of the other interfering cell (cell 2) by cycling through all MCS levels and recording the MCS yielding the highest throughput. The results are measured over several topologies (not necessarily same as the locations in Fig. 3.5). From Fig. 3.7,

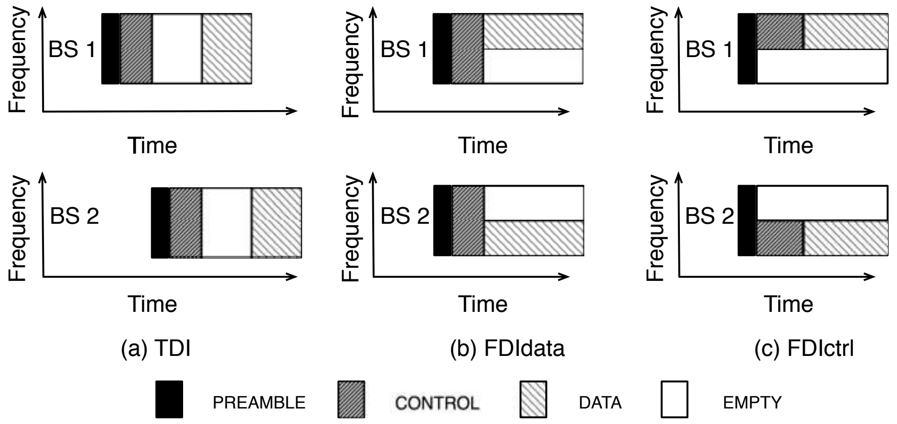


Figure 3.6: Illustration of TDI and FDI strategies.

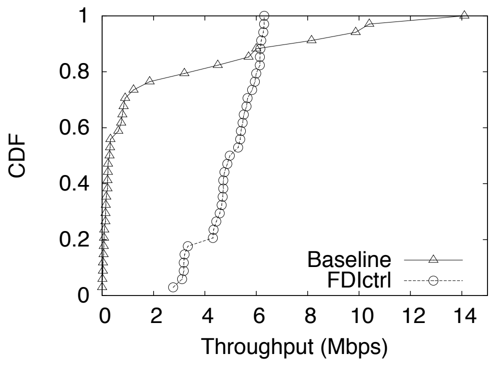
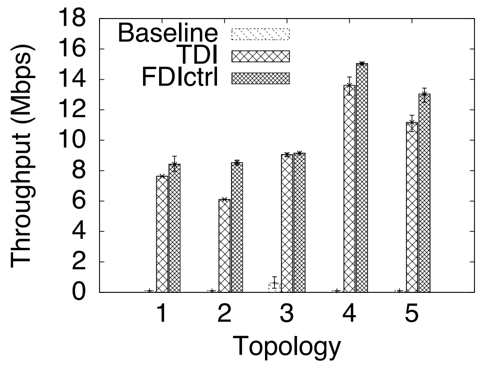


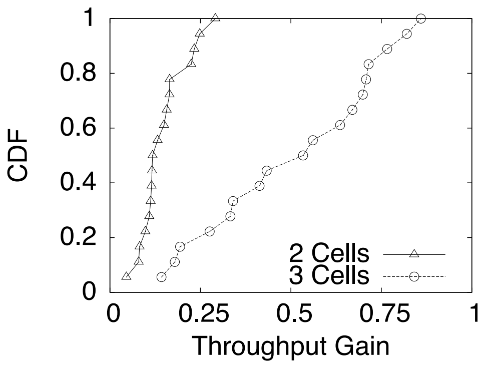
Figure 3.7: Isolating resources typically provides higher throughput gains than link adaptation with full set of resources.

we see that making tiles orthogonal limits the maximum (median) throughput that can be achieved to 6.3 Mbps (5 Mbps), because of using half of the frame. However, this is still much higher than that using all tiles with link adaptation. For a small fraction of topologies, the baseline outperforms FDIctrl; these are the locations where the SNR is high enough to support a lower MCS.

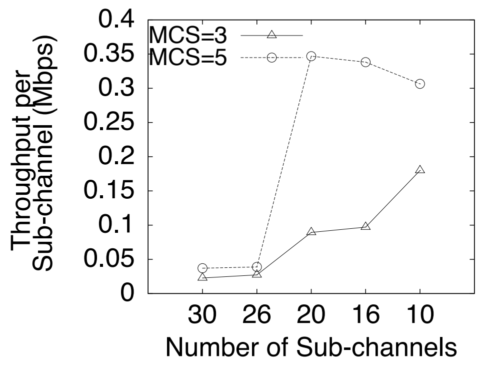
An effective resource isolation scheme has to opportunistically use all sub-channels and activate resource isolation only when it can provide benefits. Note that this is a fundamental difference between OFDMA and WiFi. In WiFi, interfering APs still use the entire spectrum albeit on an orthogonal channel. Thus, resource isolation (i.e. switching to a different channel) does not reduce bandwidth. On the other hand, resource isolation in OFDMA must be administrated with



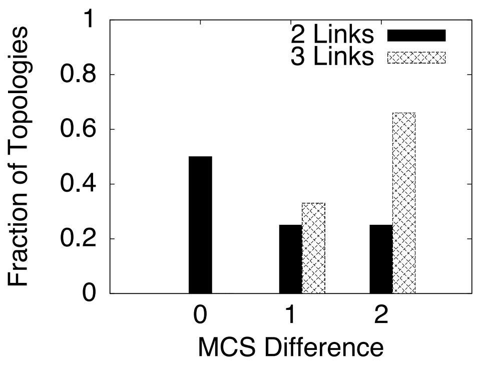
(a) Isolation for Two Cells



(b) FDIctrl vs TDI



(c) Per Sub-channel Throughput



(d) Average MCS Gain

Figure 3.8: Benefits of Frequency Domain Resource Isolation.

care. To summarize, *in femtocell deployments, link adaptation is typically insufficient to alleviate interference and has to be coupled with resource isolation. However, opportunities exist where just using link adaptation is beneficial. An effective resource management system must identify these opportunities and exploit them.*

### 3.4.3 Frequency vs. Time Domain Isolation

Next, our goal is to understand *whether resources should be isolated in time or in the frequency domain*. We consider three strategies: with the baseline strategy, BSs operate using all resources. With TDI and FDIctrl, BSs operate using half of the (orthogonal) available set of symbols and sub-channels respectively, in each frame (illustration in Fig. 3.6). All the strategies employ link adaptation via cycling through the MCS levels as discussed before.

We begin by considering two cells (cells 1 and 2). Aggregate throughput results of both clients 1 and 2 from a representative subset of topologies (created by varying the locations of both clients) are presented in Fig. 3.8(a). While resource isolation provides significant gains over the baseline, isolation in the frequency domain outperforms that in the time domain by about 20%. We repeat these experiments with all the three cells, each cell operating on a third of the resources during isolation. The CDFs of the aggregate throughput gain of FDIctrl over TDI are computed over several topologies and presented in Fig. 3.8(b). We observe that while the median gain of FDIctrl over TDI is about 17% for two cells, it increases sharply to about 60% for three cells.

A deeper look into this surprising observation reveals an inherent property leveraged by FDIctrl, referred to as *power pooling*. The energy transmitted by a BS is split over its constituent sub-channels in OFDMA. Hence, when a smaller subset of sub-channels are used, the average power per sub-channel increases, potentially allowing operations with a higher MCS than before. As more links are included, the number of (orthogonal) sub-channels available per link decreases; this however, increases the average power and hence the throughput per sub-channel. Since the total number of sub-channels remains the same in the network, the higher per sub-channel throughput translates to a higher network capacity. To illustrate, we consider cell 1 without interference and fix the MCS. Then we decrease the number of sub-channels and measure the throughput. The per sub-channel throughputs of client 1 at two different locations, one with MCS 3 and other with MCS 5 (see Fig. 3.1) are presented in Fig. 3.8(c) (results with other MCSs show similar trends but are not included to avoid cluttering the plot). We see significant gains due to power pooling on a subset of sub-channels. Reducing the set of sub-channels increases the average SNR on each sub-channel, allowing it to better sustain (decode) the MCS of operation. However, once the SNR is sufficient to sustain the MCS, any further power pooling will not yield additional gains (as with MCS 5). In Fig. 3.8(d) we plot the fraction of topologies over which the MCS supported by client 1 increases with FDIctrl as we go from two to three links. We see that a majority of the topologies benefit from an MCS gain of two levels in the three link case due to power pooling, thus resulting in the higher gains seen in Fig. 3.8(b).

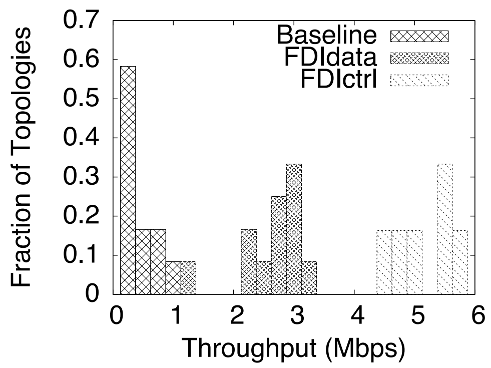
Note that the ability to pool the power of multiple cells (links) comes only with FDIctrl and not with TDI. Hence, *operating multiple cells on a subset of sub-channels but orthogonalizing them using FDIctrl is the key to alleviating interference and increasing network capacity via power pooling*. While one can completely isolate the tiles (for control and data) between cells in FDIctrl, achieving this with TDI is more challenging. Given that the control part has to immediately follow the preamble, TDI requires guard spaces over several OFDMA symbol durations to prevent collisions between interfering cells. With TDI, we measure the link throughputs without interference and take their time average to emulate perfect isolation in time domain. However, even with this best case scenario for TDI, the performance of FDIctrl is still superior.

### 3.4.4 Control vs. Data Isolation

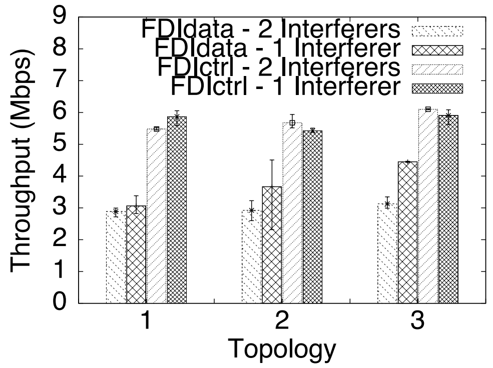
Unlike with TDI, the control payload can easily be isolated in FDIctrl. However, standards restrict the flexibility of isolation especially for the control part. For e.g., with WiMAX the control part can be only placed in three distinct sub-channel regions (segments) i.e., isolation is provided for at most three cells. If there are more cells in an interference domain, then the overlap of control parts cannot be completely eliminated. We seek to understand *"What is the impact of not isolating the control part and whether data isolation can still provide benefits in such scenarios?"*

We consider two cells (cells 1 and 2) and measure the throughput at client 1 with interference, with three strategies: baseline, FDIdata and FDIctrl (all employing link adaptation). The histogram of the results over several topologies is presented in Fig. 3.9(a). The median throughputs of the baseline, FDIdata and FDIctrl are around 0.3, 2.5 and 5 Mbps, respectively. This indicates that while complete isolation of both control and data parts can significantly alleviate interference, isolating sub-channels for the data part alone (with control parts experiencing interference) can still deliver significant benefits over the baseline strategy.

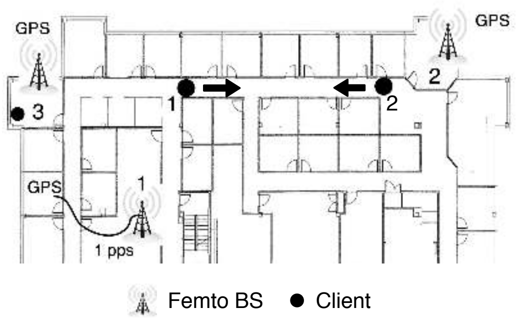
Next, we investigate the robustness of FDIdata and FDIctrl to increased interference. We now include cell 3 with the same set of sub-channels as cell 2 (already orthogonal to those in cell 1); thus, it only projects increased interference on the control payload for client 1. We present a subset



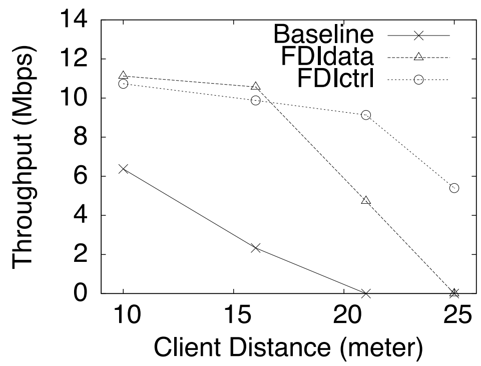
(a) One Interferer



(b) Two Interferers



(c) Topology

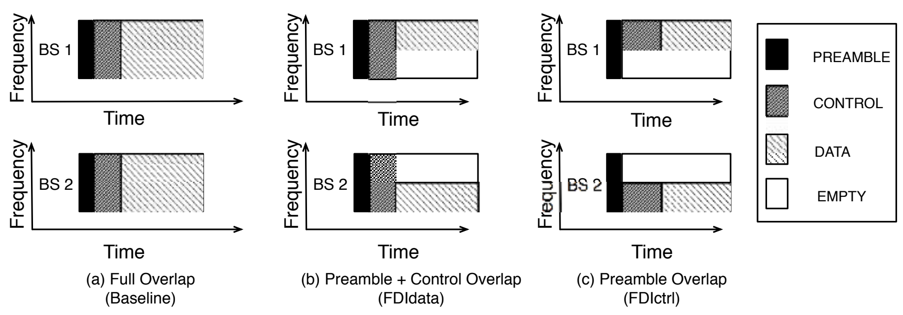


(d) Throughput Performance

Figure 3.9: Impact of Increased Interference on Resource Isolation Strategies.

of our measurements in Fig. 3.9(b). We see that increased interference does not have an impact on FDIctrl due to complete isolation (control + payload) of resources. However, increased interference on the control payload does degrade performance with FDIdata by 20-30%; however, FDIdata still delivers throughput gains over the baseline strategy.

Our next experiment is with two cells (cell 1 and 2) and their clients. Initially both clients are close to their BSs (see Fig. 3.9(c)). Gradually, we increase the distances of both clients from their BSs and move them closer to the interfering BS (as indicated by the arrows). The corresponding aggregate throughput for the three strategies shown in Fig. 3.9(d), clearly reinforce the benefits with FDIdata. We see that if the clients are close to the BS, the difference in throughputs with FDIctrl and FDIdata is insignificant since the control part is not susceptible to interference; however, as they move away, the control part is affected and the throughput with FDIdata decreases (still outperforms



Frequency
BS 1
Time
BS 2
(a) Full Overlap (Baseline)
(b) Preamble + Control Overlap (FDIdata)
(c) Preamble Overlap (FDIctrl)
PREAMBLE
CONTROL
DATA
EMPTY

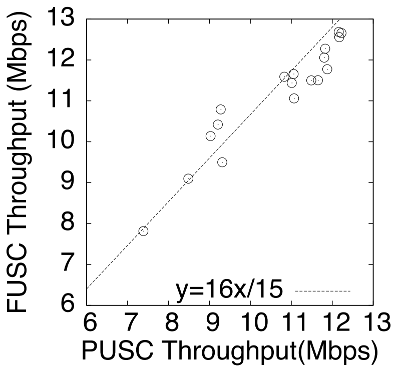
Figure 3.10: Possible interference scenarios.

baseline). In summary, *although perfect isolation of both data and control parts is best, if control part isolation is not feasible, data part isolation can still deliver significant throughput gains*.

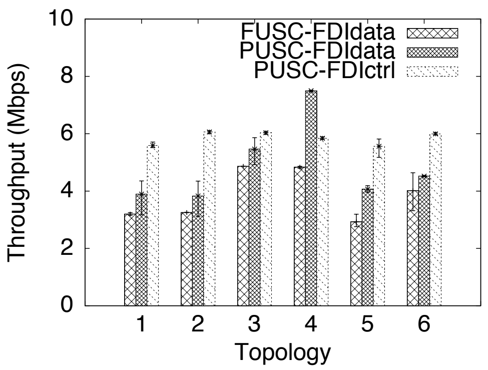
### 3.4.5 A Microscopic View

We have thus far presented macroscopic results on resource isolation. Next, we provide some of the microscopic insights from our experiments. Our clients' connection manager reports statistics including signal strength, erroneously decoded packets and the cause of errors. The causes of errors prove especially helpful in analyzing a scenario.

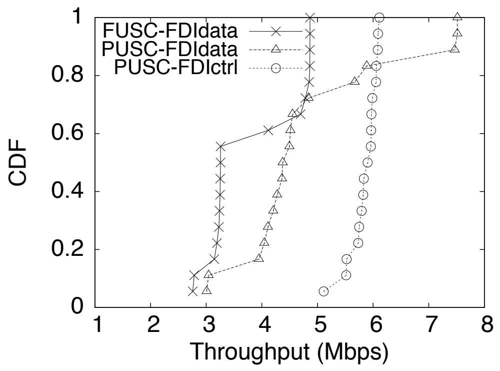
Fig. 3.10 depicts different scenarios seen in our experiments. Although both the control and data payloads are isolated in FDIctrl, collision of preambles of interfering BSs inevitably occur. We observed rare cases with FDIctrl where, the MS disconnects from its BS due to interference. We find that by placing the MS at a location very close to the interfering BS causes it to drop the connection since it cannot correctly decode the preamble from its BS (due to high power from the interfering BS). This was also common to the other scenarios (baseline and FDIdata). For FDIdata, overlap of the control payloads across cells is also a factor. We found that at locations where the control payload was corrupted, the client did not receive any packets (0 Mbps). This is because of the loss of DL-MAP which contains the parameters for correct decoding of the data bursts. Although the data bursts are isolated from interference, the corruption of DL-MAP impacts performance. Finally, with the baseline strategy, the preamble, control and data, all overlap. This



(a) No Interference



(b) Two Cell Interference



(c) Benefits of PUSC vs. FUSC

| Logical Clusters | Major Groups | Sub-channels |
|---|---|---|
| 0 - 11 | 0 | 0 - 5 |
| 12 - 19 | 1 | 6 - 9 |
| 20 - 31 | 2 | 10 - 15 |
| 32 - 39 | 3 | 16 - 19 |
| 40 - 51 | 4 | 20 - 25 |
| 52 - 59 | 5 | 26 - 29 |

(d) Sub-channel formation

Figure 3.11: Impact of Sub-carrier Contiguity on Interference.

explains the results in §3.4.2. The clients receive 0 Mbps throughput even with link adaptation due to preamble and control payload collisions; benefits are limited to a small subset of locations where the interference only affects the data payload.

## 3.5 Resource Mapping

We have so far considered isolating resources at the MAC (logical) level of sub-channels to alleviate interference. However, the sub-carrier composition of sub-channels remained the same across cells. Next we investigate how varying the sub-carrier composition at the PHY, can impact performance.

### 3.5.1 Contiguous vs. Distributed Grouping

Recall from §3.2 that current WiMAX implementations widely adopt two modes of sub-carrier grouping: contiguous (PUSC) and distributed (FUSC). Before considering interference, we experiment with PUSC and FUSC without interference to understand their baseline performance. We present client 1's throughput at various locations in Fig. 3.11(a). Note that FUSC uses 768 sub-carriers, while PUSC uses only 720 per symbol. Hence, a normalizing line is drawn at $y = \frac{768}{720}x = \frac{16}{15}x$. The clustering of points around this line indicates that the throughput of PUSC and FUSC are similar in the absence of interference.

We now introduce interference from cell 2 and measure client 1's throughput at various locations. While PUSC isolates both the data and control parts, FUSC does not allow for control part isolation [62]. With PUSC, we consider both FDIctrl and FDIdata. The results from a representative set of topologies are in Fig. 3.11(b). Three inferences are in order: **(i)** PUSC outperforms FUSC with an average gain of 35%. This can be attributed to the following effect. While the frequency offsets between cell 1 and its client 1 are taken care of using signaling in the frame, the synchronization provided externally between the cells does not address the frequency offsets between the interfering cell (cell 2) and client 1. This disturbs the orthogonality of sub-carriers between cell 1 and cell 2. Hence, although orthogonal sub-channels are used at the MAC, frequency offsets across cells allow energy from sub-carriers of cell 2 to spill over the adjacent sub-carriers of cell 1. This degrades the decoding quality at client 1. In PUSC, a contiguous set of 14 sub-carriers are grouped to form a cluster; clusters are then grouped to form a sub-channel. This contiguity of sub-carriers (within a cluster) in a sub-channel in PUSC, limits the corruption of sub-carriers (due to frequency offsets) only to those at the edges of the clusters. However, since the sub-carriers in a sub-channel are completely distributed in FUSC, most of the sub-carriers are vulnerable to the above effect. This explains the loss in performance. **(ii)** As one might expect, with PUSC, FDIctrl outperforms FDIdata. **(iii)** Interestingly however, in location 4, we find that FDIctrl does not perform the best. This is because, when the control part is isolated, it expands into the data part, thereby reducing the number of tiles available for the data payload. Hence, if the interference on the control part is not severe, the loss in throughput due to the expansion of the control part outweighs the benefits of FDIctrl.

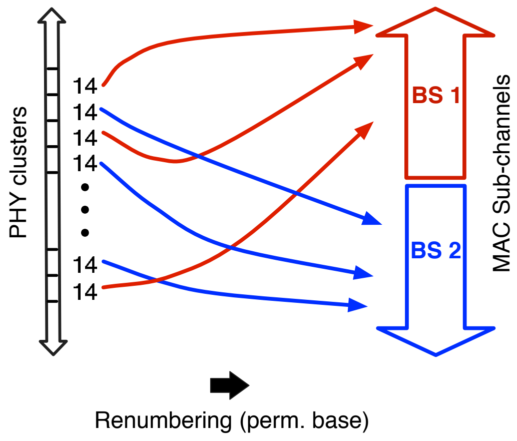
A more extensive set of measurements with each strategy is in Fig. 3.11(c). It is seen that PUSC with FDIdata provides a median gain of 50% over FUSC, while PUSC with FDIctrl can provide an additional gain of 50%. Further, for about 20% of the topologies where interference on the control part is such that it does not prevent correct decoding, FDIctrl does not pay off. In summary, *distributed sub-carrier grouping (FUSC) is more vulnerable to frequency offsets across cells. This degrades performance in the presence of interference. Further, FUSC can isolate resources only for the data part unlike PUSC and hence, is limited in addressing interference. In cases without interference, FUSC outperforms PUSC due to the higher number of sub-carriers.*

### 3.5.2 Different Permutation Sequences

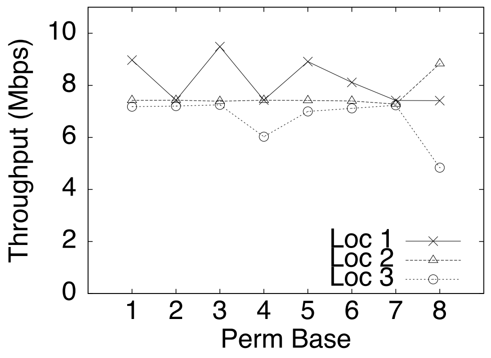
In §3.5.1, we studied the impact of contiguity in sub-carrier grouping, while keeping the tiles orthogonal between interfering cells. Given the benefits of PUSC over FUSC, we will now only consider PUSC and study permutations that control the sub-carrier composition and hence, resource orthogonality between cells at the PHY layer. In PUSC, the spectrum is first divided into 14 contiguous sub-carrier sets called *clusters*. Then, the physical clusters are shuffled as per a pre-determined renumbering sequence; these form the logical clusters, which are then gathered under major groups as in Fig. 3.11(d). Two logical clusters within a group are combined to form a sub-channel; thus, 30 sub-channels can be formed from 60 physical clusters. WiMAX uses the following formula [62] to renumber the physical clusters:

$$LN = R(PN + 13 \cdot permbase)\ mod\ N_{clusters}$$

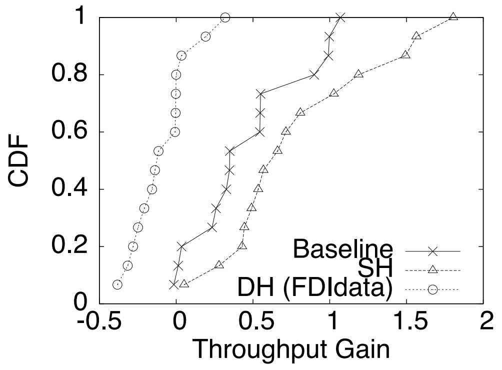
where $LN$ is the logical cluster number, $PN$ is the physical cluster number and $R$ is the renumbering sequence. Varying the permutation base ($permbase$) changes the set of physical clusters



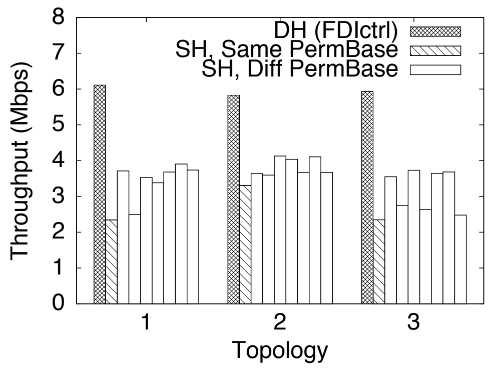
(a) Permutation Illustration



(b) No Interference



(c) Two Cell Interference



(d) Physical vs. Logical Isolation

Figure 3.12: Impact of Sub-carrier Permutation on Interference.

forming a sub-channel. Hence, two cells with orthogonal sub-channels (at the MAC) can still collide on some sub-carriers at the PHY (if their permutations cause this to happen). Fig. 3.12(a) illustrates two BSs using the same permutation base with orthogonal sub-channels at the MAC to maintain orthogonality at the sub-carrier level.

We first study the impact of permutations on a single cell (cell 1) without interference. The result in Fig. 3.12(b) shows that the frequency selectivity of the link varies with permutations, causing throughput fluctuations of 1-2 Mbps across different permutations. We refer to this inherent diversity as the *permutation gain*.

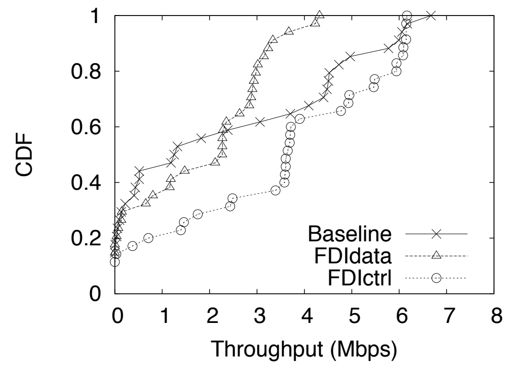
To understand the impact of permutations on interference, we consider two cells (cells 1 and 2) and three strategies: (a) baseline: both cells use all sub-channels; (b) Same Half (SH): both cells use the same (half) set of logical sub-channels; and (c) Different Half (DH): the cells use orthogonal (half) sets of logical sub-channels (FDIdata). Note that all three strategies use all sub-channels for the control part. In each strategy, cell 2 is fixed on permutation base 0, while cell 1 cycles through different permutations (including 0). The throughput of client 1 is measured at various locations. The throughput when cell 1 uses a different permutation (other than 0) than cell 2 is compared to that when both cells use the same permutation and the relative throughput gain is obtained. A CDF (over topologies) of the throughput gain of using different permutations is presented in Fig. 3.12(c). Three inferences can be made: (i) when using all resources, varying the permutations can bring a median gain of about 30%. Note that when all sub-channels are used by both cells, changing the permutation will not reduce the number of overlapping sub-carriers between the two cells. However, the permutation gain due to frequency selectivity (Fig. 3.12(b)) will still remain in the presence of interference, largely contributing to the gain observed. (ii) the median gain when using the same (half) set of sub-channels is much higher (at about 60%) compared to when using all sub-channels. Although both the cells use the same logical set of sub-channels, leaving half the sub-channels empty allows different permutations to select different sets of sub-carriers corresponding to the same set of sub-channels. This reduces the number of overlapping sub-carriers between the cells and hence interference, a benefit we refer to as *interference reduction gain*. This coupled with the permutation gain contributes to the large benefits observed with SH. (iii) With DH, the logical sub-channels used between cells are orthogonal to begin with. Hence, using the same permutation maintains the orthogonality at the sub-carrier level as well. However, when the permutation is changed between cells, this has an opposite effect (compared to SH), whereby sub-carriers now start overlapping and introducing interference. Unlike in SH, where permutation and interference reduction gains reinforce each other, they are in conflict in DH. Given that interference dominates performance, we see that using different permutations causes a median loss of about 20% with DH; permutation gain is dominant in only about 20% of the topologies.

We now compare DH with a fixed permutation that provides perfect isolation between cells (FDIctrl) against SH that uses different permutations. The results for a representative subset of topologies are in Fig. 3.12(d). As before, employing a different permutation improves performance in SH. However, since the permutation cannot be controlled to completely isolate the sub-carriers, interference still exists. Thus, its performance is inferior to that of DH with the fixed permutation. It is extremely hard to determine the permutation that yields the best performance since it varies not only with the client location, but also with the interfering cells. Fig. 3.12(b) clearly indicates that the permutation yielding the best throughput is very location-specific. An improperly chosen permutation may degrade performance, even in the absence of interference. A permutation that is the best for one location ($permbase\, 8$, Loc 2) may be the worst for another location ($permbase\, 8$, Loc 3).

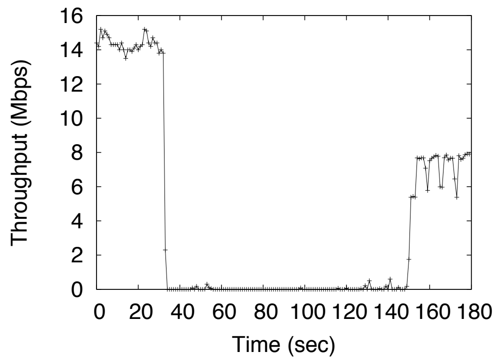
To summarize, we find that *permutations introduce diversity and can help reduce interference at the PHY when sub-channels overlap and are not completely used. However, the interference reduction benefits obtained from resource isolation at the logical sub-channel level outweigh the benefits from permutation gain. Finally, the diversity gain observed with a permutation base is location-specific and the best permutation base is hard to predict.*

## 3.6 Synchronization

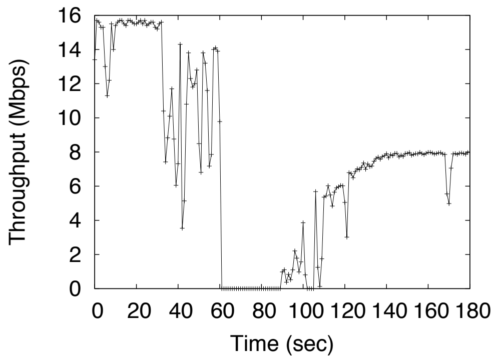
In the experiments thus far, the femtocells were synchronized via GPS modules. There are tight synchronization requirements in macrocell OFDMA networks. In upcoming femto standards [66] provisions are being made for achieving synchronization between femtocells, potentially using self-organizing (SON) servers. However, it is unclear (as of now) if tight synchronization can be achieved among femtocells. Hence, it is important to understand the impact of synchronization or lack thereof on resource isolation and resource mapping. Next, we consider cell 1 and 2 without frame level synchronization. With the preamble being transmitted at 3-5dB higher power, interference patterns could change, potentially causing further throughput degradation.



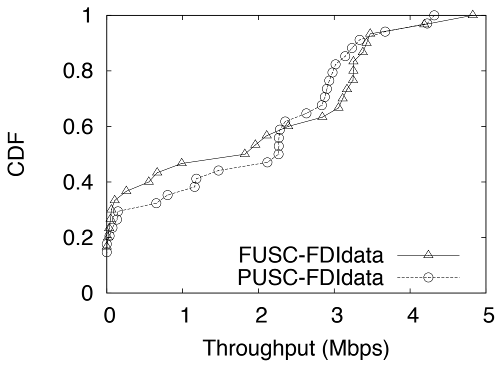
(a) Resource Isolation



(b) Pattern 1



(c) Pattern 2



(d) Sub-channelization

Figure 3.13: Impact of Lack of Synchronization on Resource Isolation and Sub-channelization.

### 3.6.1 Impact on Resource Isolation

First, we study synchronization and resource isolation. We consider three strategies: baseline, FDIdata and FDIctrl. The throughput at client 1 in the presence of interference is measured over numerous locations and its CDF is plotted in Fig. 3.13(a). It can clearly be seen that by isolating both the control and data parts of the frame between interfering cells, FDIctrl is relatively immune to lack of synchronization and continues to outperform baseline and FDIdata.While the preamble of one cell can corrupt the data of the other cell, this corruption will be restricted to one symbol with the rest of the frame being completely isolated with FDIctrl. Between baseline and FDIdata, we find that there are three distinct regions: baseline outperforms FDIdata, FDIdata outperforms baseline, and both strategies suffer.

To gain a deeper understanding, we performed a more detailed six-stage experiment with the timeline in Fig. 3.14: (1) Cell 1 alone starts operating and transferring data to client 1 on all sub-channels (All SC) from t=0 onwards, (2) cell 2 switches on and starts transmitting only the preamble and control from t=30, (3) cell 2 starts transferring data to client 2 on all sub-channels from t=60, (4) cell 1 and cell 2 switch to different sets of sub-channels from t=90 (FDIdata), (5) cell 2 stops transferring data and goes back to transmitting only the preamble and control from t=120, and (6) cell 2 switches off at t=150. Re-starting the BSs for every experimental run changes the frame alignment between cells and hence, the interference pattern. Two of the prominent patterns that repeat over numerous experiments are shown in Figs. 3.13(b) and 3.13(c). In one of the patterns in Fig. 3.13(b), we find that even when only the preamble of cell 2 is switched on, it completely inhibits the transmission of cell 1, which does not recover until cell 2 is switched off. This corresponds to the case where both baseline and FDIdata suffer (left region in Fig. 3.13(a)). In the other pattern in Fig. 3.13(c), we find that when cell 2's preamble is switched on, it does not completely disrupt cell 1's transmission but degrades its throughput from 15 Mbps to 11 Mbps. However, when cell 2 starts transferring data on all sub-channels, this completely interferes with cell 1's transmission. However, when both the cells isolate their data in stage 4, cell 1's transmission starts recovering and approaches the maximum throughput possible with half the sub-channels. This corresponds to the case, where FDIdata outperforms baseline (middle region in Fig. 3.13(a)). In the remaining 40% of topologies, the interference between the cells was not severe enough to prevent decoding even with the lowest MCS. In this case, link adaptation was sufficient to address interference (both baseline and FDIdata support the same MCS) and hence, isolating resources reduces throughput by half unnecessarily compared to operating on all sub-channels. This explains the right region in Fig. 3.13(a), where baseline outperforms FDIdata.

Thus, we find that *even without synchronization, resource isolation between interfering cells is extremely beneficial - isolation of both data and control parts continues to deliver significant benefits, while isolating the data part alone also contributes to large gains in over 30% of the scenarios*.

| Stage | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Time | 0 - 30 | 30 - 60 | 60 - 90 | 90 - 120 | 120 - 150 | 150 - 180 |
| BS1 | All SC | All SC | All SC | FDIdata | FDIdata | FDIdata |
| BS2 | - | Control | All SC | FDIdata | Control | - |

Figure 3.14: Timeline of the six stages for Fig. 3.13(b) and 3.13(c).

### 3.6.2 Impact on Resource Mapping

Next, we study the effect of lack of synchronization on resource mapping. We measure the throughput with PUSC and FUSC for cell 1 and 2. The BSs use orthogonal sets of half of the sub-channels (FDIdata) and the throughput of client 1 is measured at various locations. Fig. 3.13(d) depicts the CDF (over topologies) of throughput. We notice that PUSC outperforms FUSC in low throughput regimes, where interference is the dominant factor on performance. This is consistent with our observations in §3.5; FUSC is still vulnerable to frequency offsets between the BSs. However, for throughput samples larger than 2.5 Mbps, FUSC outperforms PUSC. We find that in these regimes where interference is not the dominant factor, client 1 alleviates the effects of interference with link adaptation. We further observe that these samples are clustered around 2.977 and 3.247 Mbps with PUSC and FUSC, respectively (with the best MCS being the same for both schemes). The clustering can be explained by the inherent benefit with FUSC due to the higher number of sub-carriers; the ratio corresponds to $\approx \frac{16}{15}$ in line with our previous observations. In summary, we find that *PUSC is less vulnerable to frequency offsets across cells and this helps in handling interference better than FUSC, even without synchronization*.

## 3.7 Design Implications

Our extensive measurements provide an understanding of the impact of interference and how to address it in femtocell networks. The key inferences are summarized below.

- Strong interference in femtocells cannot be alleviated by just link adaption and requires resource isolation.
- Isolating resources in the frequency domain can increase throughput capacity due to power pooling.
- Resource isolation, even when achieved on only the data part, can still alleviate interference.
- Having contiguity in sub-carriers (forming a sub-channel) reduces the vulnerability to frequency offsets that can cause interference across cells even when resources are isolated at the MAC level.
- Altering the sub-carrier composition of sub-channels via different permutations at the PHY provides a diversity gain but does not outweigh the benefits of interference reduction from isolation of sub-channels at the MAC.
- Isolating resources in frequency domain and retaining contiguity in sub-carriers to address interference holds promise even in the absence of synchronization.

These inferences lead us to the following set of guidelines for the design of an efficient multicell OFDMA femtocell system. When interference between femtocells does not prevent decoding with lower MCSs, it is better to operate using:

- distributed sub-carrier grouping (e.g. FUSC) to form sub-channels since this reduces the control signaling overhead, while making more room for data transmissions.
- different permutations at the PHY level to leverage the diversity gain resulting from it.

However, in the presence of strong interference, it is preferable to operate a femtocell network that

- isolates resources between interfering cells in the frequency domain to increase capacity.
- uses contiguity in sub-carriers (e.g. PUSC) to alleviate the adverse impact of frequency offsets across cells.

- trades-off permutation gain for significant interference reduction through MAC layer resource isolation.

We believe these guidelines will help network practitioners design efficient resource management solutions for femtocells. Some interesting implications that arise in the design of such solutions include:

- When interference is tolerable (via link adaptation), isolating resources will degrade performance. Hence, we would need mechanisms that differentiate between fading and interference and appropriately trigger link adaptation or resource isolation. This would in turn require interference estimation capabilities that might have to be performed using calibrating measurements or with the help of external sensors.

- When a cell has multiple MSs experiencing different levels of interference, the scheduling decisions (use of resources) would have to intelligently incorporate link adaptation and resource isolation in tandem.

- Network dynamics in the form of deployment of new cells, removal of existing cells, load variations, etc. have to be factored in when designing distributed mechanisms.

## 3.8 Discussion

Our experiments are conducted towards understanding the impact of interference *between* femtocells in indoor deployments. There have been studies that address interference mitigation between macrocells and femtocells. Designing resource management solutions with our guidelines will complement these studies and increase performance further. In our experiments, we assume a single MS associated with each BS and measure the performance with interference from other BSs. Our inferences on interference characterization will clearly hold when a BS has multiple MSs. We experiment with WiMAX BSs that use OFDMA; however, we believe that most of our inferences are applicable to other OFDMA-based technologies such as LTE and LTE-A.

## 3.9 Conclusions

In this chapter, we experimentally characterize interference in OFDMA femtocell networks. Using programmable WiMAX femtocells and commercial clients, we perform measurements with a multitude of design choices including resource isolation, resource mapping and synchronization among BSs. We derive several inferences from our measurements and provide guidelines for efficient deployment of OFDMA femtocell networks. To the best of our knowledge, this is the first study that characterizes interference in multicell OFDMA systems using measurements from an actual femtocell testbed. We believe that our guidelines provide a good starting point towards designing efficient resource management solutions for femtocells.

# Chapter 4

# A Resource Management System for Interference Mitigation in Enterprise OFDMA Femtocells

To meet the capacity demands from ever-increasing mobile data usage, mobile network operators are moving towards smaller cell structures. These small cells, called femtocells, use sophisticated air interface technologies such as Orthogonal Frequency Division Multiple Access (OFDMA). While femtocells are expected to provide numerous benefits such as energy efficiency and better throughput, the interference resulting from their dense deployments prevents such benefits from being harnessed in practice. Thus, there is an evident need for a resource management solution to mitigate the interference that occurs between collocated femtocells. In this chapter, we design and implement one of the first resource management systems, FERMI, for OFDMA-based femtocell networks. As part of its design, FERMI (i) provides resource isolation in the frequency domain (as opposed to time) to leverage *power pooling* across cells to improve capacity; (ii) uses measurement-driven triggers to intelligently distinguish clients that require just link adaptation from those that require resource isolation; (iii) incorporates mechanisms that enable the joint scheduling of both types of clients in the same frame; and (iv) employs efficient, scalable algorithms to determine a fair resource allocation across the entire network with high utilization and low overhead. We implement FERMI on a prototype four-cell WiMAX femtocell testbed and show that it yields significant gains over conventional approaches.

## 4.1 Introduction

The demand for higher data rates and increased spectral efficiencies is driving the next generation broadband access networks towards deploying smaller cell structures (called femtocells) with OFDMA [48]. Femtocells are installed indoors (e.g., enterprises, homes) and use the same spectrum and access technology as macrocells (traditional cell towers), while connecting to the core network through cable or DSL backhaul. The poor cellular signal problem indoors, experienced by many users today, can easily be overcome by femtocells. Moreover with femtocells, mobile devices (e.g., 4G-enabled smartphones) can save energy by transmitting to a nearby femtocell (rather than a distant cell tower) and enjoy high data rates. In addition, the small range of femtocells increases the cellular network capacity via increased spatial reuse. These advantages allow mobile broadband service providers to (i) improve coverage and service quality and (ii) offload traffic from macrocells to femtocells in a cost-effective manner.

To harness the aforementioned benefits in practice, one first needs to mitigate the interference that occurs in femtocell networks. Although previous studies (e.g., [67]) have proposed solutions to alleviate the interference between macrocells and femtocells, interference mitigation *between collocated femtocells* has not drawn considerable attention and thus forms the focus of this work. However, the design of a resource management solution for femtocells is complicated by the fact that the femtocells have to inter-operate with the cellular standards that they inherit from macrocells. There are several other key aspects that make the resource management problem both challenging and unique. We articulate these below.

***Femtocells versus Macrocells:*** Typical femtocell deployments are significantly more dense compared to the well-planned deployments of macrocells. Hence, while interference is localized at cell edges in macrocells, it is less predictable and more pervasive across femtocells. This renders Fractional Frequency Reuse (FFR) solutions (proposed for macrocells) inadequate in mitigating interference between femtocells.

***Femtocells versus WiFi:*** OFDMA dictates a synchronous medium access for femtocells, on a licensed spectrum. On the contrary, WiFi stations access the unlicensed spectrum in an asynchronous manner (i.e., random access via CSMA). In a typical WiFi network, interfering cells either operate on orthogonal channels or use carrier sensing to arbitrate medium access on the same channel. In an OFDMA femtocell network, there is no carrier sensing and multiple central frequencies (channels) are not available in the licensed spectrum; therefore, existing solutions for WiFi are not applicable. Interfering femtocells can either operate on orthogonal parts (called sub-channels) of the spectrum, or directly project interference on the clients of each other. Further, in OFDMA, transmissions to different clients of a single cell are multiplexed in each frame. Since every client of a cell may not need spectral isolation (for purposes of interference mitigation), blindly operating adjacent cells on orthogonal parts of the spectrum comes at the cost of underutilization of the available capacity. In other words, resource isolation in OFDMA femtocells needs to be executed by jointly considering interference mitigation and leveraging spatial reuse opportunities. Since a WiFi access point transmits data to a single client at a time (using the entire channel width assigned to it), this challenge does not arise.

**Our contributions in brief:** We design and implement one of the first practical resource management systems, FERMI, for OFDMA-based femtocell networks. FERMI decouples resource management across the network from scheduling within each femtocell and addresses the former. This allows resource allocation across femtocells to be determined by a central controller (CC) at coarse time scales. Frame scheduling within each femtocell can then be executed independently on the allocated set of resources. The four key cornerstones of FERMI's resource management solution include:

- *Frequency Domain Isolation:* It isolates resources for clients in each femtocell, in the frequency domain (as opposed to the time domain). This allows for *power pooling* to jointly mitigate interference and increase system capacity (discussed later).

- *Client Categorization:* It employs proactive, measurement-driven triggers to intelligently distinguish clients that require just link adaptation (i.e., clients that can reuse the spectrum) from those that require resource isolation with an accuracy of over 90%.
- *Zoning:* It incorporates a frame structure that supports the graceful coexistence of clients that can reuse the spectrum and the clients that require resource isolation.
- *Resource Allocation and Assignment:* It employs novel algorithms to assign orthogonal subchannels to interfering femtocells in a near-optimal fashion.

We have implemented FERMI on a four-cell WiMAX femtocell testbed. FERMI provides a complete resource management solution while being standards compatible; this enables its adoption on not only experimental platforms but also on commercial femtocell systems. To the best of our knowledge, we report the first resource management system implemented on a real OFDMA femtocell testbed. Comprehensive evaluations show that FERMI yields significant gains in system throughput over conventional approaches.

The rest of the chapter is organized as follows. We describe background and related work in §4.2. Experiments that motivate FERMI's design are in §4.3. The building blocks of FERMI are described in §4.4. In §4.5, we describe the resource allocation algorithms that are part of FERMI. We evaluate FERMI in §4.6 and we conclude in §4.7.

## 4.2 Background and Related Work

In this section, we describe relevant related work. We then provide brief background on WiMAX femtocell systems.

***Macrocellular Networks:*** While broadband standards employing OFDMA (WiMAX, LTE) are relatively recent, related research has existed for quite some time [68]. There are studies that address problems pertaining to single cell (e.g., [69]) and multi-cell systems [67, 54, 70, 71, 72, 73, 53, 6].

The above studies on multi-cell systems have looked at the interference between macrocells and femtocells, for both downlink (e.g., [6]) and uplink communications (e.g., [53]). In summary, the solutions in these studies leverage the localized interference and the planned cell layouts of macrocells, and they are restricted to theoretical studies with simplifying assumptions that prevent their adoption in practice. On the other hand, femtocell deployments in practice are not planned and thus, do not benefit from localized interference.

For interference between macrocells and femtocells (i.e., macro-femto interference), we assume one of the models in [6]. In this model, macro-femto interference is mitigated by partitioning the frame resources across macrocells and femtocells in a semi-static manner (based on macro and femto users' traffic). Thus, our focus in this work is specifically on interference *between* femtocells. While FERMI's overarching goal of femtocell interference mitigation is similar to that of the above studies, it is the first study that achieves this with a standards-compatible implementation. We evaluate FERMI on real OFDMA hardware and show that it is viable for commercial femtocell deployments.

***Spectrum Allocation:*** In addition to the studies in the cellular domain, there have been studies addressing resource allocation using graph coloring for WiFi systems (e.g., [2, 25, 3, 61]). The main objective in these studies is to allocate a minimum number of orthogonal contiguous channels to each interfering AP. Instead, our objective is to realize a weighted max-min fair allocation while utilizing as many sub-channels (fragments of the spectrum) as possible. In addition, resource allocation is just one component of our study; we implement a novel, complete resource management system with several enhancements specifically tailored to OFDMA. There have also been approaches that allocate spectrum fragments to contending stations (e.g. [58, 60]). However, these studies rely on asynchronous random access and associated sensing capabilities. We address a more challenging problem in OFDMA synchronous access systems and satisfy requirements that are specific to OFDMA femtocells.

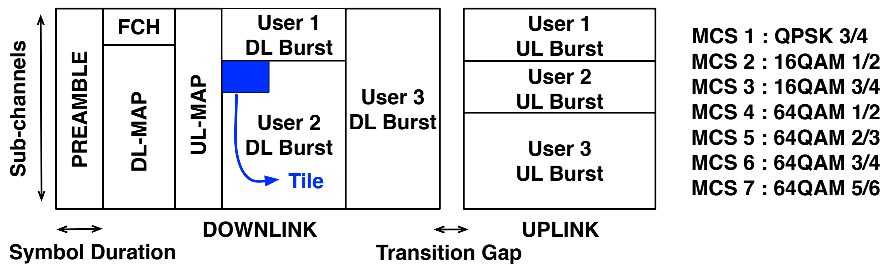


Figure 4.1: A simplified illustration of the WiMAX frame structure.

### 4.2.1 WiMAX Preliminaries

While our study applies to OFDMA femtocells in general, our measurements are conducted on a WiMAX (802.16e [62]) femtocell testbed. WiMAX divides the spectrum into multiple sub-carriers and groups several sub-carriers to form a sub-channel. Specifically, we use the distributed grouping mode (PUSC [62]) in our implementation since it is mandatorily supported. In PUSC, the individual sub-carriers forming a sub-channel are distributed throughout the spectrum. This distribution (i.e., which sub-carriers are selected as part of a sub-channel) is subject to specific permutations. Thus, two different sub-channels at the MAC level map to two distinct sets of sub-carriers at the PHY level. In general, interference from the same source may be different on different sub-channels if frequency selectivity is taken into account. However, note that PUSC picks the subcarriers composing a sub-channel randomly from the spectrum. This averages the effect of frequency selectivity and interference on a given sub-channel, thereby giving a uniform effect across sub-channels.

A WiMAX frame (see Fig. 3.1) is a two-dimensional template that carries data to multiple mobile stations (MSs) across both time (symbols) and frequency (sub-channels). The combination of a symbol and a sub-channel constitutes a *tile* (the basic unit of resource allocation at the MAC). Data to users are allocated as rectangular bursts of tiles in a frame. The BS [1] schedules the use of tiles both on the downlink and the uplink. Frames are synchronized in time both between the BS and MSs as well as across BSs (by virtue of synchronizing to the macro BS [74]). The frame consists of the preamble, control and data payload. While the preamble is used by the MSs to

[1]We use the terms femtocell, BS, cell interchangeably.

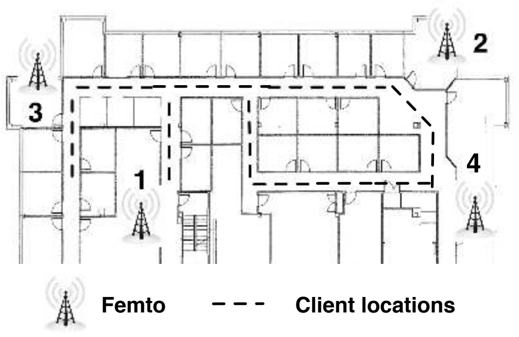


Figure 4.2: Femtocell Testbed Deployment

lock on to a particular BS, the control consists of FCH (frame control header) and MAP. MAP conveys the location of the data burst for a MS in a frame and consists of both the downlink and uplink MAPs. The DL-MAP indicates where each burst is placed in the frame, which MS it is intended for, and what modulation level (MCS as shown in Fig. 3.1) decodes it. Similarly, the UL-MAP indicates where the MS should place its data on the uplink frame. The uplink frame also has dedicated sub-channels for HARQ, which is used by the MSs to explicitly acknowledge (ACK / NACK) the reception of each burst sent by the BS.

## 4.3 Design Aspects of FERMI

To derive the right design choices for interference mitigation, we conduct extensive measurements on our testbed. The testbed consists of four PicoChip [63] WiMAX femtocells (cells 1-4) deployed in an indoor enterprise environment at NEC Laboratories America Inc. (Fig. 4.2). The clients are commercial WiMAX USB dongles (with unmodifiable, proprietary source code [64]) plugged into laptops running Windows XP. All cells use 8.75 MHz bandwidth with the same carrier frequency of 2.59 GHz. For this frequency, we obtained an experimental license from FCC to transmit WiMAX signals on the air.

There are two approaches to coping with interference in OFDMA. Switching to a lower MCS via link adaptation (rate control) could suffice if the received signal quality is above the threshold required by the lower MCS level. With strong interference (typical in dense deployments), the received SINR could be even lower than that required for the lowest MCS operation. In such cases, isolating the resources (tiles) utilized by interfering cells helps alleviate the effects, but it results in a reduced set of tiles in each cell. Clearly, the choice between link adaptation and resource isolation must be made depending on the nature of interference. In a two-dimensional WiMAX frame, the tiles can be isolated among BSs either in time (symbols) or in the frequency (sub-channels) domain as depicted in Fig. 3.6. Time domain isolation (TDI) isolates tiles by leaving empty (guard) symbols to prevent collisions; frequency domain isolation (FDI) allocates orthogonal sets of sub-channels to different BSs for their transmissions.

### 4.3.1 Accommodating Heterogeneous Clients

As discussed earlier, for clients in close proximity to their BS, link adaptation alone may be sufficient to cope with interference. Invoking resource isolation for such clients will underutilize the tiles in the frame. Given that OFDMA multiplexes data to multiple clients in a given frame (to fill the available tiles), it becomes necessary to accommodate clients with heterogeneous requirements (link adaptation vs. resource isolation) in the same frame. To achieve this, we propose to use *zoning*, where an OFDMA frame is divided into two data transmission zones [2]. The first zone operates on all sub-channels and is used to schedule clients that need just link adaptation (hereafter referred to as the *reuse zone*). The second zone utilizes only a subset of sub-channels (determined by FDI) and here, the clients that require resource isolation are scheduled (referred to as the *resource isolation zone*). Link adaptation is also performed for clients in this zone albeit only within the restricted subset of sub-channels.

We conduct an experiment with two cells to understand the benefits of zoning. Cell 2 causes interference while cell 1 transmits data to its clients. Cell 1 schedules data for two clients: one by reusing all sub-channels (the reuse client) and the other one by isolating resources (from cell 2). The reuse client is moved from the proximity of cell 1 towards cell 2; the other client is static.

[2] A zone in OFDMA is a dedicated portion of the frame in which one or more bursts can be scheduled.

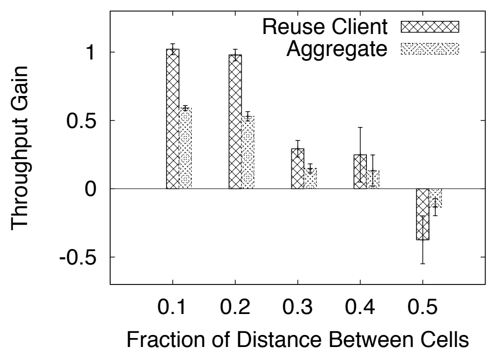


Figure 4.3: Motivation for having separate zones in the frame.

We compare the throughput that cell 1 delivers (aggregate throughput of both clients) against a scheme where there is no reuse (i.e. both clients are scheduled by isolating resources). As one might expect, as long as the reuse client does not experience appreciable interference from cell 2, reusing sub-channels provides a throughput gain over the pure resource isolation scheme. We plot this throughput gain in Fig. 4.3 as a function of the reuse client's distance from cell 1. Interestingly, significant gains (at least 20%) from reusing sub-channels can be availed even when the client is at 40% of the distance between the interfering cells. Beyond this distance, the interference from cell 2 starts degrading the throughput. We revisit zoning in detail in §4.5.

Although zoning holds promise, it only dictates how to accommodate heterogeneous clients; it does not provide a complete resource management solution. Several challenges remain in achieving this goal. Specifically, for each cell, we need to **(a)** determine the size (in symbols) of the reuse zone **(b)** determine the subset of sub-channels allocated to the resource isolation zone, and **(c)** adapt both these zones to the dynamics of the network in a scalable manner. FERMI incorporates novel algorithms to address these challenges.

## 4.4 Building Blocks of FERMI

We depict the relationship between the blocks of FERMI in Fig. 4.4. In a nutshell, the *categorization* of clients allows each BS to determine how the frame should be divided into zones, from its

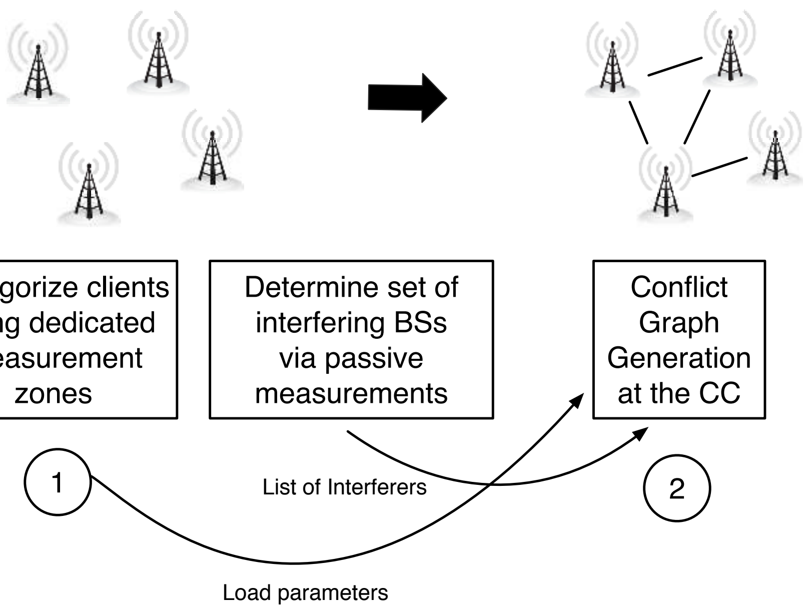

Categorize clients using dedicated measurement zones
Determine set of interfering BSs via passive measurements
Conflict Graph Generation at the CC
1
2
List of Interferers
Load parameters

Figure 4.4: The building blocks of FERMI.

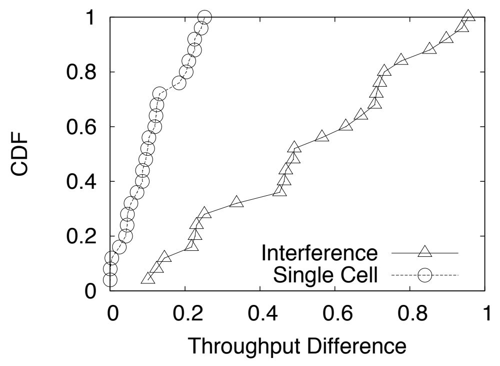


Figure 4.5: Calibration measurements for client categorization.

perspective (block 1). Each BS then determines the set of BSs that cause interference on those of its clients that require resource isolation. This information, along with cell-specific load parameters, is then fed to the central controller (CC), which then constructs an interference map (block 2). Using the interference map (i.e., conflict graph), the CC computes the network wide sub-channel allocation and zoning parameters (details in §4.5). It disseminates this information back to the BSs, which use these operational parameters until the next resource allocation update.

### 4.4.1 Client Categorization at the BS

The first building block categorizes clients into two classes; the first needs just link adaptation (class *LA*) while the second needs resource isolation together with link adaptation (class *ISO*). To understand how clients are to be categorized as either class *LA* or class *ISO*, we perform calibration experiments. We consider two cells each with a single client. We experiment over a large set of client locations to generate a plurality of scenarios. We first consider a cell in isolation (i.e., no interference). At each client location, we experiment by sequentially allocating two spectral *parts* (of equal size) of the frame to the client. Since, the fading effects on the two sets of assigned sub-channels are likely to be different, the client will receive different throughputs with the two different allocations. *We notice however, that the difference between the two allocations is at most 25 % in more than 90 % of the considered client locations* (Fig. 4.5). We now repeat the experiment, but with interference. In one of the allocations (i.e., parts), the second cell projects interference on the client; in the other, the operations are without interference (via resource isolation). *We observe that in this case, there is a throughput difference of over 25 % (in many cases, significantly higher) in more than 80 % of the topologies.*

These results suggest that the throughput (per tile) difference between an *interference-free* allocation and an allocation with interference can be used to categorize a client as class *LA* or class *ISO*. If this difference is less than a threshold (referred to as $\alpha$ later), link adaptation suffices. If it is larger than the threshold, one cannot immediately determine if the client needs resource isolation. This is because the above experiments were done by allocating equal number of tiles to the client both with and without interference. If such a client is categorized as class *ISO* and allocated a smaller set of isolated resources, its throughput may in fact only be similar to what it would achieve by being a class *LA* client. Unfortunately, it is difficult to know the cell loads a priori and hence one cannot make a clear determination of whether to categorize these clients as class *LA* or class *ISO*. Thus as a design choice, FERMI takes a conservative approach and categorizes all of such clients as class *ISO*. We find that this helps accommodate fluctuations in the load and interference patterns.

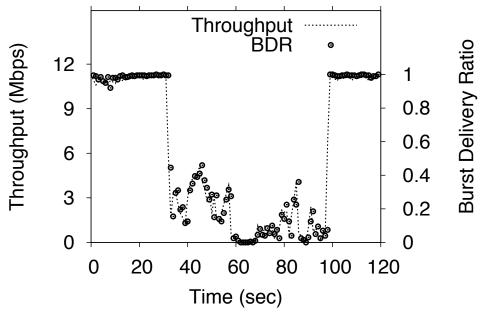


Figure 4.6: The accuracy of BDR to estimate throughput.

Although a BS does not have access to the throughput at a client, it is informed about the reception of each burst via ACKs and NACKs on the uplink. We define Burst Delivery Ratio (BDR) to be the ratio of successfully delivered bursts to the total number of transmitted bursts by the BS. The BS can *estimate* BDR by taking the ratio of the number of ACKs received to the total number of feedbacks (ACKS + NACKS) received from the clients. Since the feedback itself might practically get lost on the uplink, this is an estimate of the actual BDR. We perform experiments to understand if the BDR estimate at the BS can provide an understanding of the throughput at the client. Fig. 4.6 plots a sample result showing that indeed the BS can very accurately *track* the client throughput using the BDR estimates. We find that, as per the WiMAX standard, the feedback channels on the uplink modulate data using robust QPSK modulation. This helps in reducing the probability of a feedback being received in error by the BS and makes the BDR estimate accurate. Similar notions of uplink feedback channels are also available in other OFDMA standards such as LTE.

Having established that BDR accurately represents throughput, we next describe our categorization solution. Before providing details, we sketch how throughput in an OFDMA system can be computed. A rectangular burst consists of $x$ symbols and $y$ sub-channels which collectively form $t = x * y$ tiles. If a single burst is successfully received by the client, it delivers $t * b$ bits of information where $b$ is the bits per tile that the MCS used for the burst encodes. Here, the MCS is

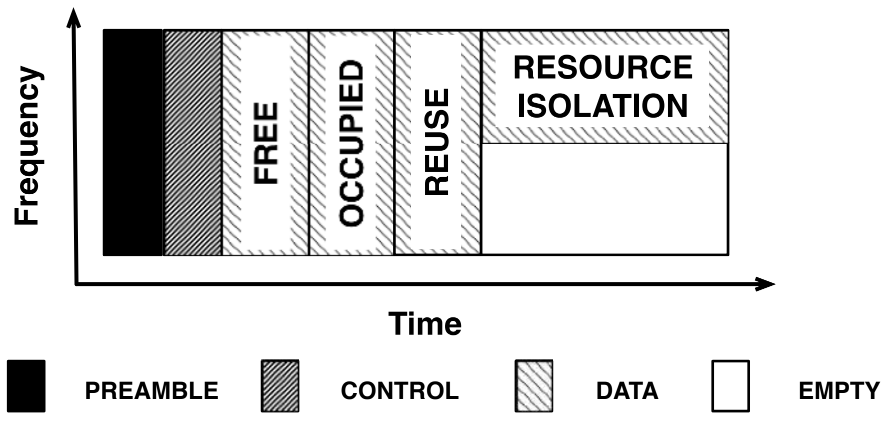
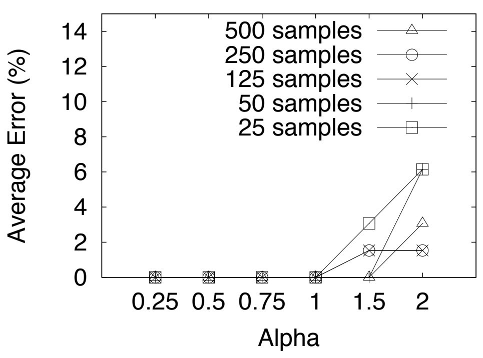


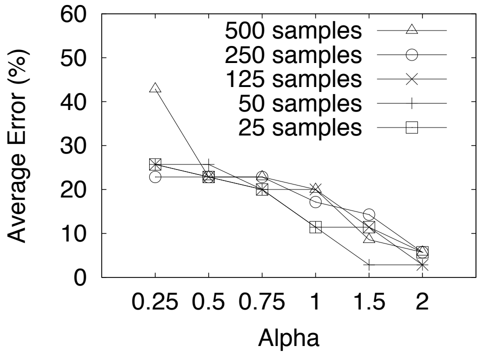
Figure 4.7: Illustration of the measurement zones in the frame.

typically chosen based on the SINR of the link. Thus, the throughput over $j$ transmitted bursts can be computed as $t * b * j * BDR$.

To achieve categorization in practice, FERMI introduces two measurement zones in the frame as depicted in Fig. 4.7, namely the *occupied* and *free* zones. Every BS operates using all sub-channels in the *occupied* zone. Scheduling a client in this zone enables the BS to calculate the BDR in the presence of interference from other cells. Scheduling a client in the *free* zone to calculate the BDR without interference is slightly more involved. Given a set of interfering BSs, all BSs but for one must leave the *free* zone empty in any frame. Allowing only one of the interfering BSs to schedule its clients in the *free* zone, will enable it to measure the BDR without interference at its clients. Hence, a random access mechanism with probability $\frac{\gamma}{n}$ is emulated to decide access to the *free zone*, where $n$ is the number of interfering BSs and $\gamma \geq 1$ is a constant parameter set by the CC. Note that clients associate with BSs at different instants and hence it is unlikely that all interfering BSs will categorize their clients at the same time. Hence, $\gamma$ is used to increase the access probability to the *free* zone. FERMI schedules regular data bursts in the measurement zones to calculate the BDR, thereby keeping the process transparent to clients and retaining standards compatibility. While the *occupied* zone can be used as an extension to the reuse zone when categorization of the clients is complete, this is not possible for the *free* zone, whose utility is towards categorization in other cells.



(a) Class *ISO* Ground Truth



(b) Class *LA* Ground Truth

Figure 4.8: The accuracy of the client categorization component.

Here, the CC keeps track of client (dis)associations, triggers the use of the *free* zone (cast as a data zone) solely for the purpose of categorization in relevant parts of the network and disables it to minimize overhead once the procedure is complete.

The accuracy of client categorization is evaluated in Figs. 4.8(a) and 4.8(b). We again consider two cells; clients 1 and 2 belong to the two cells, respectively. We generate multiple topologies by varying the location of client 1 in the presence of interfering cell 2. First, the throughput of client 1 is measured for both zones (*free* and *occupied*) to identify the ground truth; here leveraging our calibration measurements, we conclude that if the throughput difference is less than $25\%$, client 1 is at a location where it only needs link adaptation. Otherwise, it needs resource isolation. After the ground truth is established, cell 1 performs the following set of actions.

1. For $K$ samples, transmit to a client using the *occupied* zone and with the probability specified earlier, transmit to the same client using the *free* zone. During this period keep track of the BDR for each zone.

2. Calculate *occupied* throughput $T_{occ} = t * b_{occ} * K * BDR_{occ}$ where $BDR_{occ}$ is the BDR of the *occupied* zone and $b_{occ}$ is the number of bits that the sampled MCS encodes. Similarly, calculate $T_{free} = t * b_{free} * K * BDR_{free}$ (note that the number of tiles considered, $t$, is equal for the two zones).

3. If $T_{free} >= (1 + \alpha) * T_{occ}$, then the client is of class *ISO*; otherwise belongs to class *LA*. Based on the calibration measurements, we set $\alpha = 0.25$.

Here, arbitrating the access to the *free* zone is a factor that reduces the accuracy of estimation. If two BSs schedule their clients in this zone at the same time, rather than getting a BDR sample without interference, they both could get a sample that indicates interference. We use the BDR average over multiple samples to alleviate such inaccuracy.

The categorization accuracy when the ground truth is (a) resource isolation and (b) link adaptation is shown in Figs. 4.8(a) and 4.8(b), respectively. In corroboration with our measurement-based inference, we see that increasing $\alpha$ beyond 0.25 decreases the accuracy of detecting resource isolation but it increases the accuracy of detecting link adaptation. Further, while increasing the number of samples over which $\alpha$ is measured can help improve accuracy, the benefits are not significant. Hence, it pays to use fewer samples to categorize clients (towards reducing overhead). Thus, FERMI uses $\alpha = 1$ with 25 samples to obtain an accuracy greater than $90\%$.

### 4.4.2 Interference Map Generation

The CC in FERMI generates an interference (conflict) map that not only captures point-to-point but also cumulative interference experienced by the clients. Note that interference is client-dependent and since multiple clients are scheduled in tandem in each OFDMA frame, the interference patterns between BSs vary from one frame to another. This makes it impossible for any practical resource management scheme to gather schedule-dependent interference information, determine an allocation and disseminate it to the BSs for execution in every frame (sent every 5ms in WiMAX). Hence, the goal of the resource management scheme in FERMI is to allocate resources at a coarser time scale (over hundreds of frames) by collecting *aggregate* interference statistics from each BS. This decouples resource allocation from frame scheduling in each BS, thereby allowing a conflict graph approach to adequately capture interference dependencies for our purpose.

In addition to client categorization, the measurement zones in FERMI also help in deciphering interference relations. If a BS causes interference to the clients of another BS so as to require resource isolation, then an edge is added between the two BSs in the conflict graph. Note that the interference relations need to be determined only for class *ISO* clients. FERMI uses the measurements in the *occupied* zone as the basis to categorize a client as class *ISO*. Note however that *all* BSs operate in this zone and thus, the client experiences the cumulative interference from all interfering BSs. Adding an edge to each of these neighboring cells in the conflict graph would be overly conservative; some of them may only project weak levels of interference on the client. Hence, we need to determine the minimum set of interference edges that need to be added in the conflict graph to eliminate interference through resource isolation. Towards this, we use the following procedure following the initial categorization.

Consider a femtocell $A$ and a class *ISO* client $cl$ of $A$. $cl$ *passively* measures the received power from neighboring BSs (available during handover between BSs). If the power from a neighboring BS ($B$) exceeds a threshold, then $B$ is added to $cl$'s list of strong interferers. $cl$ reports this list to $A$, which then consolidates it and reports the set of conflict edges (for each strong interferer) that must be added to the conflict graph, to the CC. The CC uses this information for making the initial resource allocation decision. While this accounts for point-to-point interference, some clients may not see any individual strong interferer but the cumulative power from a subset of neighbors could be strong enough to require resource isolation. Such clients will continue to see interference after the initial resource allocation. These clients can be identified by comparing the BDR achieved on the assigned sub-channels with that seen in the *free* zone. We adopt an iterative approach to further refine the conflict graph to isolate such clients.

To illustrate, let us consider one such client. The client reports a list of neighbor BSs (not already reported) in decreasing order of received power as potential cumulative interferers to its BS. Now, FERMI needs to identify the smallest subset of these neighbors to whom adding a conflict edge will eliminate interference through resource isolation. To achieve this two filters are applied: (i) from the cumulative interference list sent by the client, the BS first removes those neighbors from the list to whom an edge was already added by one of its other clients; (ii) the pruned list is sent to the CC, where neighbors whose current resource allocation are orthogonal to that of the client's cell are further removed as they could not have caused interference. The CC then looks at the final pruned lists and adds a conflict edge to the neighbor that appears first in multiple lists. With the updated interference graph, resource allocation is determined once again and disseminated to femtocells for the next resource allocation epoch. If the BDR for the client is sufficiently improved and is now within $\alpha\%$ of what is observed in the *free* zone, the process is complete. If not, the next strongest interfering BS is added to the conflict graph (again subject to filtering based on the current resource allocation) and so on. In addressing cumulative interference, conflict edges are added only one by one in each epoch by the CC. This is because most of the interference experienced by clients is strong in dense femtocell deployments. Hence, aggressively adding conflict edges to address cumulative interference will only result in under-utilization of resources. Thus, using both passive and active measurements at clients, FERMI accounts for both strong and cumulative interference in its resource allocation decisions in each epoch.

***Why Dedicated Measurements?:*** One could argue that using only the passive received power measurements from interfering BSs may be an easier approach to categorize clients. Here, if a client receives a signal from an interfering BS that is higher than a threshold, it is categorized as class *ISO*; otherwise, it is a class *LA* client. However, for this method to work well in practice, a lot of calibration is needed to find accurate, often scenario dependent, threshold values. In addition, the received power does not necessarily give an indication of the throughput observed at the clients. To avoid these practical issues, FERMI relies on highly accurate direct measurements for client categorization, which allows it to have coarse thresholds for identification of strong interferers.

Having categorized the clients and identified the interference dependencies between femtocells, we are now ready to present the resource allocation algorithms executed by the CC.

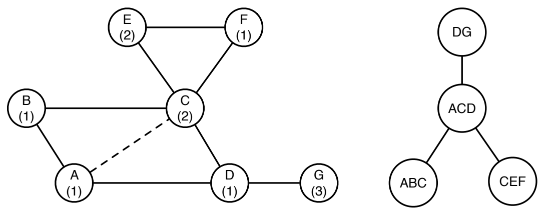
## 4.5 Algorithms in FERMI

The goal of resource management at the CC is to determine for each femtocell (i) the size of the *reuse* zone and, (ii) the specific subset of sub-channels for operations in the *resource isolation* zone, to obtain an efficient and fair allocation across femtocells. While the joint determination of parameters for both the zones is the optimal approach, this depends on throughput information that changes in each frame, thereby coupling resource allocation with per-frame scheduling decisions. Since, as discussed in §4.4, per-frame resource allocation is infeasible due to practical constraints, FERMI performs resource allocation at coarser time scales.

Each femtocell reports two parameters to the CC to facilitate resource allocation: (i) load (number of clients) in its *resource isolation* zone, and (ii) desired size (in time symbols) of its *reuse* zone. Alternative definitions for load can be adopted but the number of clients is sufficient for our purposes (as in [25]). Note that a femtocell does not have the complete picture of interference dependencies across cells; it only has a localized view. Thus, it simply provides the load in its resource isolation zone and expects the CC to allocate resources proportional to its load. Each femtocell determines the desired size of its reuse zone based on the relative load in the two zones. Since class *ISO* clients will be scheduled immediately after the reuse zone (see Fig. 4.7), if two interfering cells have different sizes for their reuse zones, then the cell with the larger reuse zone will cause interference to the class *ISO* clients of the other cell. Hence, an appropriate size for the reuse zone of each cell also needs to be determined by the CC based on the reported desired values. Next, we present the algorithm at the CC to determine the sub-channel allocation and assignment to each femtocell, followed by the selection of their reuse zone sizes.

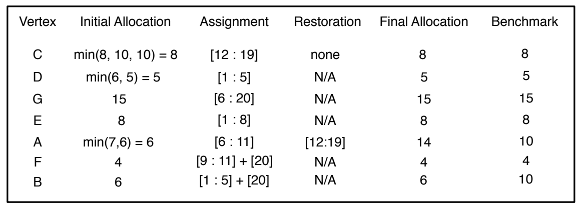
### 4.5.1 Allocation and Assignment

The goal of sub-channel allocation is to allocate and assign sub-channels to the resource isolation zone in each femtocell to maximize the utilization of sub-channels in the network subject to a weighted max-min fairness model. The reasons for the choice of the weighted max-min fairness



Triangulation

Clique Tree



| Vertex | Initial Allocation | Assignment | Restoration | Final Allocation | Benchmark |
|---|---|---|---|---|---|
| C | min(8, 10, 10) = 8 | [12 : 19] | none | 8 | 8 |
| D | min(6, 5) = 5 | [1 : 5] | N/A | 5 | 5 |
| G | 15 | [6 : 20] | N/A | 15 | 15 |
| E | 8 | [1 : 8] | N/A | 8 | 8 |
| A | min(7,6) = 6 | [6 : 11] | [12:19] | 14 | 10 |
| F | 4 | [9 : 11] + [20] | N/A | 4 | 4 |
| B | 6 | [1 : 5] + [20] | N/A | 6 | 10 |

[a : b] denotes the set of sub-channels from a to b (inclusive)

Figure 4.9: Illustration of $A^3$ algorithm for 20 sub-channels in the spectrum. The vertex loads are included in parentheses.

are two-fold: (i) weights account for variations in load across different cells; and (ii) max-min allows for an almost even split of sub-channels between cells in a contention region, which in turn maximizes the benefits from power pooling (see §4.3). Thus, given the load for the resource isolation zone from each cell along with the conflict graph, the CC's goal is to determine a weighted (load-based) max-min allocation of sub-channels to femtocells (i.e. vertices in the graph).

**Theorem 1.** *The sub-channel allocation and assignment problem in FERMI is NP-hard.*

*Proof.* Consider the simpler version of the problem, where we are interested only in the optimum objective (max-min) value and not in the specific allocation to all the cells. Hence, we are interested only in determining the largest number of sub-channels $x$ such that all cells can at least be given an allocation of $x$. This in turn can be determined as follows. For a given integer $i$, replace each vertex (femtocell) in the conflict graph $G = (V, E)$ by a clique of size $i$ and add edges between all vertices across the cliques if there existed an edge between the vertices corresponding to the cliques in the original graph $G$. Let the resulting graph be $G_i$. Now, determining the maximum $i$ such that $G_i$ is colorable with $N$ colors ($N$ being the # of sub-channels) yields the desired objective value. However, determining if $G_i$ is colorable with $N$ colors is the problem of multi-coloring $G$, where each vertex requires as assignment of $i$ colors. Thus, the max-min value to our simpler problem is $x$ if and only if $G$ can be $x$-fold colored with $N$ colors. Since multi-coloring is NP-hard and forms a special case of a simpler version of our problem, the hardness automatically carries over. $\square$

While the allocation problem may seem similar to multi-coloring at the outset, this is not the case. In fact, multi-coloring can only provide an assignment of sub-channels for a specified allocation. However, in FERMI, we are also interested in determining a weighted max-min allocation in addition to the assignment, which makes the problem much more challenging. Further, every contiguous set of sub-channels allocated to a cell is accompanied by an information element in the control part of the frame (MAP), describing parameters for its decoding at the clients. This constitutes overhead, which in turn increases with the number of discontiguous sets allocated to a cell. Therefore, our goal is to *reduce* overhead due to discontiguous allocations, while ensuring an efficient allocation of sub-channels.

**Overview of FERMI's resource allocation:** Any resource allocation algorithm attempts to allocate shared resources between entities in a contention region subject to a desired fairness. Each contention region corresponds to a maximal clique in the conflict graph. However, a given femtocell may belong to multiple contention regions and its fair share could vary from one region to another. This makes it hard to obtain a fair allocation, for which it is necessary to identify all maximal cliques in the graph. However, there are an exponential number of maximal cliques in general graphs with no polynomial-time algorithms to enumerate them. Hence, we propose an alternate, novel approach to resource allocation in $A^3$ (outlined in Algorithm 3), which runs in polynomial-time and provides near-optimal fair allocation with minimal discontiguity (overhead).

```
1: Triangulate: $A^3$ first transforms the given conflict graph $G$ into a chordal graph $G'$ by adding
   a minimal set of virtual interference edges to $G = (V, E)$.
2: Allocate and Assign: $A^3$ computes a provably weighted max-min fair allocation on the
   chordal graph $G'$.
3: Restore: $A^3$ removes the virtual edges from $G'$ and updates the allocation to the vertices
   carrying the virtual edges to account for under-utilization on the original graph $G$.
```

**Algorithm 3**: Allocation and Assignment Algorithm: $A^3$

**Chordal Graphs:** A *chordal* graph does not contain cycles of size four or more. Chordal graphs have significant applications in sparse matrix computations and have been extensively studied. Algorithms for important problems such as maximum clique enumeration can efficiently be applied on chordal graphs [75]. The key idea in $A^3$ is to leverage the power of chordal graphs in obtaining a near-optimal allocation. We now present details of the three steps in $A^3$ along with a running example in Fig. 4.9.

**Triangulation:** The process of adding edges to chordalize (triangulate) a graph is known as *fill-in*. Since adding edges to the conflict graph would result in a conservative allocation than is required, the goal is to add the *minimum* number of edges needed for triangulation. While this is a NP-hard problem in itself, $A^3$ employs a maximum cardinality search based algorithm [76] that is guaranteed to produce a *minimal* triangulation and runs in time $O(|V||E|))$, where $V$ is the set of vertices and $E$ is the set of edges in the graph. Fig. 4.9 depicts a fill-in edge between vertices $A$ and $C$. As we shall subsequently see, the restoration (third) step in $A^3$ is used to alleviate the under-utilization introduced by the triangulation.

```
1: **INPUT:** $G' = (V, E')$ and load $\ell_i, \forall v_i \in V$
2: **Allocation:**
3: Un-allocated vertices $\mathcal{U} = V$, Allocated vertices $\mathcal{A} = \emptyset$
4: Determine all the maximal cliques $\mathcal{C} = \{C_1, \ldots, C_m\}$ in $G'$ using perfect elimination ordering
5: Resource: $R_j = N$, Net load: $L_j = \sum_{i:v_i \in C_j} \ell_i, \forall C_j$
6: Determine tuples: $s_i = \max_{j:v_i \in C_j}\{L_j\}$,
   $t_i = \sum_j 1_{v_i \in C_j}, \forall v_i$
7: Determine initial allocation:
   $A_i = \min_{j:v_i \in C_j} \left\lfloor \frac{\ell_i R_j}{\sum_{k:v_k \in C_j} \ell_k} + 0.5 \right\rfloor, \forall v_i \in \mathcal{U}$
8: **while** $\mathcal{U} \neq \emptyset$ **do**
9:     Pick un-allocated vertex with maximum lexicographic rank: $v_o = \arg\max_{i:v_i \in \mathcal{U}}(s_i, t_i)$
10:    Allocate $A_o$ sub-channels to $v_o$; $\mathcal{U} \leftarrow \mathcal{U} \backslash v_o$,
       $\mathcal{A} \leftarrow \mathcal{A} \cup v_o$
11:    Update remaining resource: $R_j = R_j - A_o$,
       $\forall j : v_o \in C_j$
12:    Remove $v_o$ from cliques: $C_j \leftarrow C_j \backslash \{v_o\}, \forall j : v_o \in C_j$; Update $L_j$ $\forall j$ and $(s_i, t_i)$ $\forall v_i \in \mathcal{U}$
13:    Update allocation:
       $A_i = \min_{j:v_i \in C_j} \left\lfloor \frac{\ell_i R_j}{\sum_{k:v_k \in C_j} \ell_k} + 0.5 \right\rfloor, \forall v_i$
14: **end while**
```

**Algorithm 4**: Weighted Max-min Fair Allocation Algorithm

**Allocation:** $A^3$ uses Algorithm 4 to determine the weighted max-min allocation on the triangulated graph $G'$. Once the graph is triangulated, all its maximal cliques are listed in linear time

($O(|V|)$) by determining a perfect elimination ordering (PEO) [76]. $A^3$ determines the net load on each maximal clique (step 5) and for every un-allocated vertex (cell, $v_i$), it determines a tuple $(s_i, t_i)$, where $s_i$ indicates the highest load in the cliques that $v_i$ belongs to and $t_i$ is the number of cliques that it belongs to (step 6). $A^3$ then determines a vertex's weighted fair share in each of the maximal cliques that it belongs to and determines its minimum (rounded) share amongst all its member cliques (step 7). It picks the vertex ($v_o$) with the highest lexicographic rank and allocates the computed share of sub-channels to it (vertex $C$ is picked first with $s_c = 5$ and $t_c = 3$). $v_o$ is then removed from the list of un-allocated vertices (steps 8-10). The allocated vertex is also removed from the cliques that it is a member of, and the clique loads, resource and vertex tuples are correspondingly updated (steps 11,12). The weighted share for the remaining set of un-allocated vertices in each of the maximal cliques that $v_o$ belongs to is updated based on the remaining resources in those cliques (step 13). The process is repeated until all vertices receive allocation and runs in time $O(|V|^2)$.

**Assignment:** After the vertices get their weighted max-min allocation, the next step is to provide an actual assignment of sub-channels to satisfy the allocations. $A^3$ leverages clique trees for this purpose. A clique tree for a chordal graph $G$ is a tree whose nodes are maximal cliques in $G$. Further, it satisfies some useful properties (as we show later).

$A^3$ generates a clique tree for the chordal graph $G'$ (depicted in Fig. 4.9) in linear time by building on top of a PEO or by constructing a maximum spanning tree [75]. It picks an arbitrary node in the clique tree as its root and starts sub-channel assignment proceeding from the root to its leaves. At every level in the tree, it assigns sub-channels to un-assigned vertices in each of the nodes (maximal cliques) based on their allocation (vertex $D$ is assigned first with sub-channels [1:5]). When assigning sub-channels to a vertex, it picks a contiguous set of sub-channels that is disjoint with existing assignments to other vertices in the same clique. When contiguous assignment is not possible, $A^3$ makes the assignment to minimize fragmentation (e.g. vertex $B$ is assigned two fragments). Since a vertex may belong to multiple maximal cliques, once its assignment is made, it is retained in all subsequent levels of the tree. We establish later that the above procedure that runs in $O(|V|)$ can yield a feasible assignment of sub-channels (i.e. proper coloring of $G'$) to satisfy the allocation.

**Restoration:** Fill-in edges could result in conservative (under-utilized) allocation of resources. While the triangulation in $A^3$ attempts to reduce the addition of such edges, we still need a final step to restore potential under-utilization. $A^3$ re-visits vertices that carry fill-in edges and removes such edges one by one. When a fill-in edge is removed, the removal of a conflict may free up some sub-channels at each of the vertices carrying the edge. If so, the largest set of such sub-channels (that do not conflict with the assignment of neighbor vertices) are directly assigned to those vertices (for vertex $A$, sub-channels [12:19] are freed after the conflict removal with $C$ and can be re-assigned to $A$). This can be done in $O(|V|)$.

To summarize, given the exponential number of cliques in the original graph, $A^3$ intelligently transforms the graph into a chordal graph with only a linear number of cliques and optimally solves the allocation and assignment problem. $A^3$ keeps the potential under-utilization due to virtual edges to a minimum with its triangulation and restoration components. Thus, it provides near-optimal performance for most of the topologies with a net running time of $O(|V||E|)$. We now establish two key properties of $A^3$.

**Property 1.** *$A^3$ produces a weighted max-min allocation on the modified graph $G'$.*

*Proof.* Before the proof, we recap the definition of max-min fairness, which needs to be slightly modified given that fractional channel allocations are not possible. An allocation vector is said to be *max-min* if it is not possible to increase the allocation ($x$) of an element without decreasing that of another element with an allocation of $x + 1$ or lesser. The corresponding *weighted max-min* definition requires that an increase in the allocation ($x$) of an element with weight $w_i$ is accompanied by a decrease in the allocation of another element $j$ (with weight $w_j$) with an allocation of $\frac{w_j x}{w_i} + 1$ or lesser.

Note that a weighted max-min allocation on $G'$, where the weights correspond to the load on the vertex, is equivalent to a max-min allocation on a transformed graph $G^*$, where each vertex $v_i$

in $G'$ is replaced by a clique with $\ell_i$ nodes (sub-vertices) in $G^*$ and an edge between two vertices in $G'$ translates to an edge between all sub-vertices of the two vertices in $G^*$. A clique $C_m$ with $m$ vertices in $G'$ now translates to a clique with $\sum_{i:v_i \in C_m} \ell_i$ sub-vertices in $G^*$. Thus, to show $A^3$'s allocation mechanism yields a weighted max-min allocation on $G'$, it is sufficient to show that it yields a max-min allocation in $G^*$ where each sub-vertex has unity load (weight). We will show this by contradiction.

Assume a given allocation is not max-min. Hence, consider two sub-vertices $v_a, v_b$ in a maximal clique $C_1$ with respective allocations being $x$ and $x + 2$. For the allocation to not be max-min, we should be able to increase $v_a$'s allocation either in isolation or by decreasing $v_b$'s allocation. Since $v_a$'s allocation was restricted to begin with, it must belong to a bigger clique than $C_1$, namely $C_2$. This is because, when all the loads are unity, the allocation mechanism picks vertices belonging to bigger cliques first. Now, if $v_a$'s allocation can be increased by one sub-channel, then this must also not exceed the resource capacity of $C_2$. There are three possible cases.

(i) If $v_a$ was allocated after other sub-vertices in $C_2$, then $v_a$'s allocation would have already expanded to occupy the remaining capacity of $C_2$, which means that $v_a$'s allocation cannot be increased further.

(ii) If $v_a$ was not the last sub-vertex in $C_2$ to be allocated, then it might be that the sub-vertices that followed $v_a$ were bottlenecked in other larger cliques and could only receive an allocation $< x$, thereby allowing $v_a$'s allocation to be further increased to use up the remaining capacity in $C_2$. However, this is not possible since the sub-vertices that follow $v_a$ in $A^3$ can only belong to equivalent or smaller cliques and will hence be able to receive an allocation $\geq x$, thereby using up $C_2$'s capacity completely. Thus, $v_a$'s allocation cannot be increased in this case as well.

(iii) Similar to $v_b$, let $v_c \in C_2$ be another sub-vertex with an allocation $\geq x + 2$. In this case, $v_a$'s allocation can be increased at the cost of $v_b$'s and $v_c$'s allocations in the two cliques. However, note that if $v_c$ was allocated before $v_a$, then its allocation will be $\leq x$. Further, since $v_a$ is bottlenecked in $C_2$, if $v_c$ is allocated after $v_a$, then its allocation can at most be $x + 1$. Thus, in either case, $v_a$'s increase will have to come at the cost of another sub-vertex with an allocation $\leq x + 1$. $\square$

**Property 2.** *$A^3$ always produces a feasible assignment of sub-channels for its allocation.*

*Proof.* $A^3$ generates a clique tree for $G'$ and starts assigning sub-channels to vertices in each of the nodes (cliques) in the clique tree starting from the root. $A^3$ can run into assignment problems if it encounters an un-assigned vertex $v_i$ belonging to multiple cliques at the same level with conflicts such that it prevents a feasible assignment to $v_i$. This is where the *clique intersection property* of clique trees come into play.

The clique intersection property states that for every pair of distinct cliques $C_1, C_2$, the set of vertices in $C_1 \cap C_2$ will be contained in all the cliques on the path connecting $C_1$ and $C_2$ in the tree. Hence, if $v_i$ appears at multiple cliques at a level in the tree, it must have appeared in isolation at some higher level in the tree. Given that all vertices, when encountered first, appear in a single clique at a level, a *feasible* assignment to the vertex is always possible since the allocations always satisfy the capacity in all cliques. Further, once a vertex is assigned sub-channels, its assignment carries over to other cliques containing it in subsequent levels. Thus, by using a clique tree for assignment, $A^3$ is able to ensure a feasible assignment of sub-channels given a weighted max-min allocation. $\square$

Based on these two properties, we have the following result.

**Theorem 2.** *If $G$ is chordal, then $A^3$ produces an optimal weighted max-min allocation.*

Through our comprehensive evaluations in Section 4.6, we show that over 70% of the topologies are chordal to begin with for which $A^3$ is guaranteed to yield an optimal allocation. For the remaining topologies, $A^3$'s sub-optimality is within 10%, indicating its near-optimal allocation capability.

***Other possible comparative approaches:*** While greedy heuristics for multi-coloring do not address our allocation problem, to understand the merits of $A^3$, we propose and consider two extensions to such heuristics that also perform allocation and assignment (coloring). These simpler heuristics do not need to operate on a complete list of maximal cliques as we describe next.

The first heuristic is *progressive* (labeled *prog*); here, the allocations and assignments are made in tandem one sub-channel at a time. The vertex with the smallest weighted allocation ($\frac{\text{allocation}}{\text{load}} = \frac{A_i}{\ell_i}$) is chosen and assigned the smallest indexed sub-channel available in its neighborhood. By assigning sub-channels one at a time, this heuristic achieves reasonable fairness. However, its running time is $O(|V|^2 N)$, where its dependence on $N$ (number of sub-channels) makes it pseudo-polynomial, thereby affecting its scalability. It also results in a highly fragmented assignment of sub-channels, which in turn increases the control overhead in frames.

Another heuristic that can avoid the pseudo-polynomial complexity, is *interference-degree* based (labeled *deg*). The share to every vertex is determined based on its weight and the remaining resources (after removing allocated vertices) in its interference neighborhood and is ($\frac{\ell_i(N - \sum_{j:(v_i,v_j)\in E, v_j \in \mathcal{A}} A_j)}{\sum_{j:(v_i,v_j)\in E, v_j \in \mathcal{U}} \ell_j}$). Then the vertex with the min. share is allocated as contiguous of a set of sub-channels as possible. This heuristic runs in $O(|V|^2)$ and also keeps the overhead low. However, its fairness is significantly worse as compared to *prog*.

By adopting a greedy approach, heuristics derived from multi-coloring either achieve low complexity and overhead at the cost of fairness but not both. $A^3$ however, deciphers interference dependencies with good accuracy to provide both near-optimal fairness and reduced complexity and overhead. Further, since the allocation and assignment is conducted on the chordal graph $G'$, dynamics in the form of arrival/departure of clients/cells (i.e. addition/deletion of conflict edges) can be easily accommodated in a purely localized manner through incremental schemes [77]. This in turn allows $A^3$ to scale well to network dynamics unlike other heuristics.

**Benchmarking:** To understand how close $A^3$ is to the optimum, we need to obtain the weighted max-min allocation on the original graph $G$ (not necessarily chordal). This requires listing all the maximal cliques, which are exponential in number. However, this can be done in a brute-force manner with exponential complexity. Once all the maximal cliques are listed, the allocation procedure of $A^3$ can be directly applied to obtain a weighted max-min allocation on $G$.

### 4.5.2 Zoning

We addressed the assignment of sub-channels to the resource isolation zone of each cell. Our next step is to determine the size of the reuse zone (in symbols) for each cell based on their desired sizes. There arise three challenges in determining the reuse zone size (referred to as $s_r$). (i) If two interfering cells use two different $s_r$'s, the one with the larger $s_r$ will cause interference to the class *ISO* clients of the other cell. Hence, a common reuse zone is required among interfering cells. (ii) Since allocation and zoning are meant to operate at coarse time scales (decoupled from per-frame scheduling), the common $s_r$ among interfering cells cannot be determined based on throughput. Hence, the choice of the common $s_r$ is restricted to either the minimum or maximum of the desired zone sizes of the neighboring cells. (iii) If each cell belongs to a single contention region (clique), choosing the common $s_r$ is easy. However, since cells may belong to multiple cliques, this will result in a common $s_r$ (minimum or maximum) propagate to the entire network. Cells with a desired zone size less than the common $s_r$ may not have sufficient data for their class *LA* clients to fill up to the $s_r$, while cells with a larger desired zone size will have to perform isolation (without reusing sub-channels). Either case results in under-utilization, which is exacerbated when a single common $s_r$ is allowed to propagate to the network.

FERMI addresses the above challenge as follows (illustration in Fig. 4.10). For each cell, the CC determines the minimum of the advertised (desired) $s_r$'s of all the cell's neighbors and uses that as its operational $s_r$ (e.g. 10 symbols for BS1, 5 symbols for BS2). The cell schedules its class *LA* clients in the reuse zone till the operational $s_r$ (using all sub-channels). It continues to schedule class *LA* clients in the second zone between its operational $s_r$ and its desired $s_r$. However, these are scheduled only in the band allocated to the cell by $A^3$ (the scheduling of BS2 between the $5^{\text{th}}$ and the $15^{\text{th}}$ symbols). The class *ISO* clients are scheduled in the resource isolation zone (after the desired $s_r$) using the band allocated by $A^3$.

Introducing a *transition* zone (marked on Fig. 4.10) that schedules class *LA* clients between the operational and desired $s_r$s (using the band given by $A^3$), provides a graceful transition between the

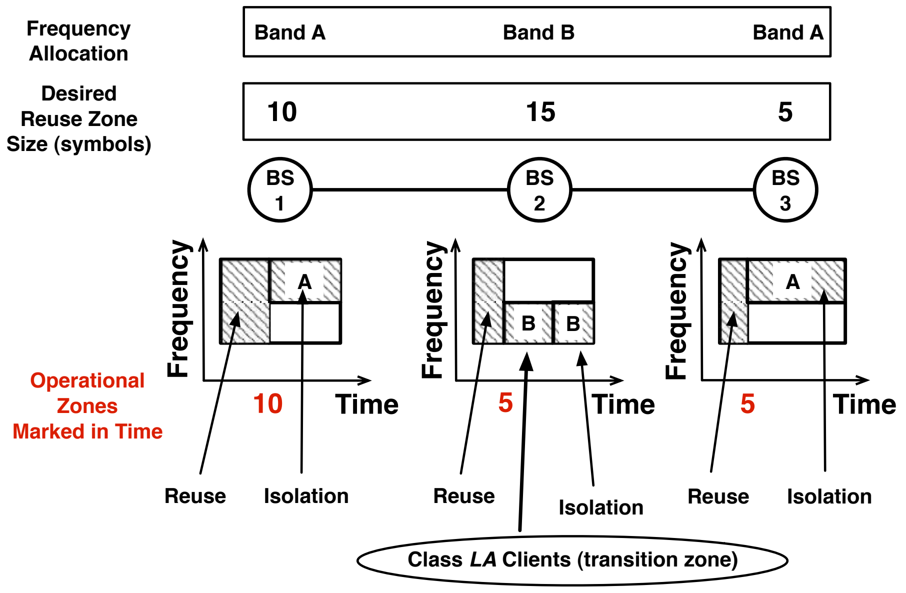


Figure 4.10: Illustration of zoning mechanism of FERMI.

reuse and resource isolation zones. Since the chance for under-utilization is more when the operational $s_r$ exceeds the desired $s_r$, FERMI uses the minimum of the desired $s_r$s in the neighborhood as the operational $s_r$ for a cell. Further, since each cell computes its operational $s_r$ only based on the desired $s_r$s of its neighbors and not their operational $s_r$s, propagation of a single common $s_r$ in the network (and the resulting under-utilization) is avoided. As an example, this would correspond to every BS having the same $s_r$ (i.e. global min.) of 5 symbols in Fig. 4.10. Using the minimum of the desired $s_r$s of neighbors (i.e. local min.) avoids this propagation for BS1 and allows it to have a $s_r$ of 10 symbols. Hence, different regions of the network can have different $s_r$ values, which increases the potential for sub-channel reuse. Further, cells that belong to multiple contention regions with different operational $s_r$s in the different cliques (e.g. BS2 in Fig. 4.10) will not suffer from interference to their class *ISO* clients, since the operational $s_r$ of all their cliques will be less than their desired $s_r$, while they schedule only class *LA* clients in the region between their operational and desired $s_r$. As we shall show in our evaluations, FERMI's zoning arrangement provides significant gains since the $s_r$ values in different cliques can be decoupled (i.e. a single globally minimum desired $s_r$ does not propagate).

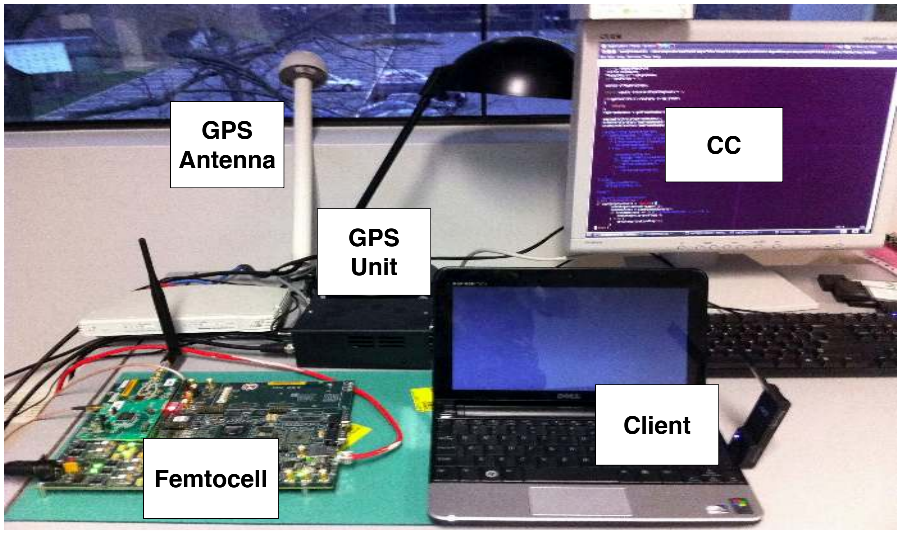


Figure 4.11: Our WiMAX equipment.

## 4.6 System Evaluation

To evaluate FERMI, we both conduct experiments on our testbed as well as use simulations. Simulations help evaluate the scalability and the relative performance of each algorithm with parameters that are not easy to adapt in practice.

### 4.6.1 Prototype Evaluations

**Implementation Details:** Fig. 4.11 shows a picture of our testbed equipment. Given that we do not have a macro BS at our disposal, we use external GPS modules to achieve synchronization among femtocells. The GPS modules are placed next to windows with cables providing a 1 pulse per second (pps) signal to each femtocell (antenna and cable depicted). The clients are USB dongles connected to laptops.

FERMI is implemented on the PicoChip platform which provides a *base reference design* implementation of the WiMAX standard. The reference design does not involve sophisticated scheduling routines and provides just a *working link* between the BS and the MS. Since the clients are off-the-shelf WiMAX MSs (with no possibility of modification), it is a challenge to realize a working

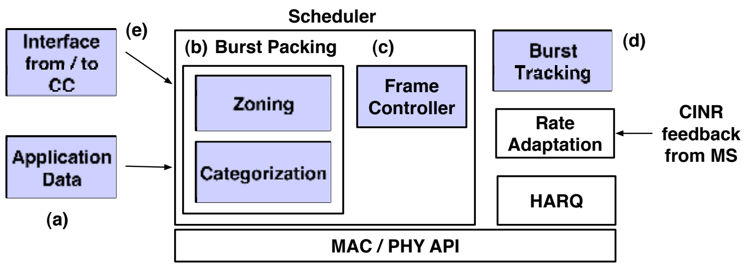


Figure 4.12: Implementation details of FERMI.

implementation of various components such as categorization and zoning. Some of these challenges were to keep our implementation within the *boundaries* of the rigid WiMAX frame structure and to integrate commercial clients with our experimental testbed. We significantly extend (shown as shaded components in Fig. 4.12) the reference design to implement FERMI. Specifically, our implementation operates as follows.

**(a)** When data from higher layers is passed onto the MAC, we first route the data based on what MS it is intended for and whether that MS is already categorized (as in §4.4) or not. **(b)** If the MS is already categorized, its data is packed in the relevant zone of the frame that the MS needs (reuse vs. resource isolation). If not, its data is packed in the measurement (recall *free* and *occupied*) zones introduced for categorization. The burst packing component implements a rectangular alignment of the data of both MSs that have been categorized before as well as MSs that are being categorized. **(c)** After packing, the data is passed onto the frame controller which prepares the control payload before the frame is transmitted on the air. **(d)** The burst tracking component keeps an information tuple for the measurement zones for the MSs that are being categorized. It tracks the ACK status of each measurement burst. After enough BDR samples are collected, it decides on the client category and informs the burst packing component about the decision. **(e)** The interface with the CC leverages kernel sockets to communicate the load and conflict information to the CC via Ethernet and receives operational parameters for zoning and allocation (used by the burst packing component).

**Experimental Evaluations:** Next, we evaluate the performance of each algorithm using our testbed. We create five topologies as shown in Fig. 4.13. The dotted edge between BS1 and

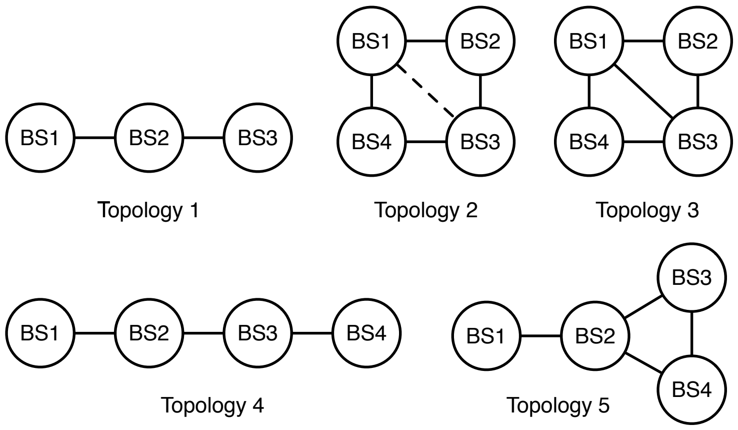


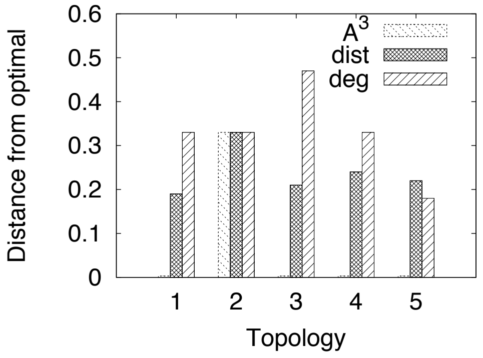
Figure 4.13: Topologies used for prototype evaluation.

BS3 in Topology 2 is the $fill\text{-}in$ edge introduced by $A^3$ (other topologies are already chordal). In generating these topologies, we leverage our WiMAX testbed (see Fig. 4.2) by changing the client locations for each BS. We measure the fairness of each algorithm relative to the optimal allocation (benchmark) as the normalized distance to the benchmark $d = \sqrt{\sum_{i \in V}(t_i - s_i)^2} \ / \ \sqrt{\sum_{i \in V}(s_i)^2}$ [78] where $t_i$ and $s_i$ denote the number of sub-channels assigned to vertex $i$ by an algorithm and the benchmark, respectively.

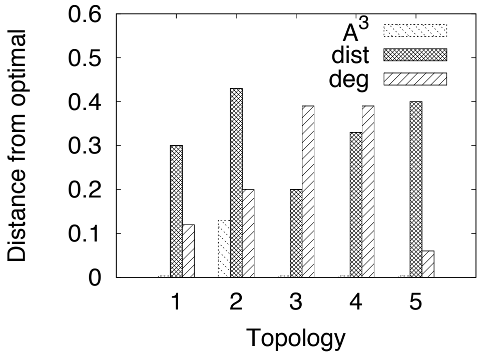
***Throughput and Fairness:*** In our testbed, the cells have $N = 30$ sub-channels available in the spectrum. Each BS has two clients (one class *LA*, one class *ISO*). When there is no zoning, we schedule both clients on the same set of sub-channels allocated to the BS. For scenarios with zoning, the specific zoning strategy determines the size of the reuse zone and the resource isolation zone.We perform experiments for each topology with the allocation determined by each algorithm (assuming equal load at each BS). Here, we introduce another heuristic (labeled *dist*) that decides the share of a vertex based on its weight and the resources in the neighborhood (without removing the allocated vertices). The share of a vertex $i$ becomes $\frac{\ell_i N}{\sum_{j:(v_i,v_j)\in E} \ell_j}$. It mimics a distributed degree-based allocation and helps us understand the importance of having a centralized approach.

| | **Topology 1** | | | | **Topology 2** | | | | **Topology 3** | | | | **Topology 4** | | | | **Topology 5** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Algorithm** | $A^3$ | **dist** | **deg** | **BM** | $A^3$ | **dist** | **deg** | **BM** | $A^3$ | **dist** | **deg** | **BM** | $A^3$ | **dist** | **deg** | **BM** | $A^3$ | **dist** | **deg** | **BM** |
| **BS1 (1)** | 15 | 15 | 20 | 15 | 20 | 10 | 20 | 15 | 10 | 7 | 7 | 10 | 15 | 15 | 20 | 15 | 20 | 15 | 23 | 20 |
| **BS2 (2)** | 15 | 10 | 10 | 15 | 10 | 10 | 10 | 15 | 10 | 10 | 16 | 10 | 15 | 10 | 10 | 15 | 10 | 7 | 7 | 10 |
| **BS3 (3)** | 15 | 15 | 20 | 15 | 20 | 10 | 20 | 15 | 10 | 7 | 7 | 10 | 15 | 10 | 10 | 15 | 10 | 10 | 11 | 10 |
| **BS4 (2)** | - | - | - | - | 10 | 10 | 10 | 15 | 10 | 10 | 16 | 10 | 15 | 15 | 20 | 15 | 10 | 10 | 12 | 10 |
| **Utilization** | 45 | 40 | 50 | 45 | 60 | 40 | 60 | 60 | 40 | 34 | 46 | 40 | 60 | 50 | 60 | 60 | 50 | 42 | 53 | 50 |
| **Throughput(Mbps)** | 20.8 | 18.3 | 21.8 | - | 29 | 19.7 | 27.8 | - | 19.6 | 15.8 | 20.7 | - | 26.7 | 22.7 | 27 | - | 23.9 | 19.9 | 25 | - |

Table 4.1: Throughput and utilization of each algorithm along with individual allocations (for equal load) for the BS.



(a) Fairness (equal load)



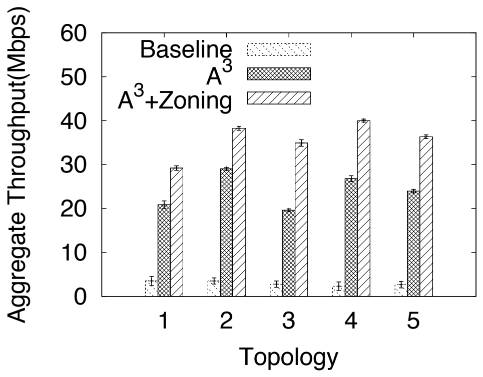
(b) Fairness (variable load)

Figure 4.14: Fairness benefits of $A^3$.

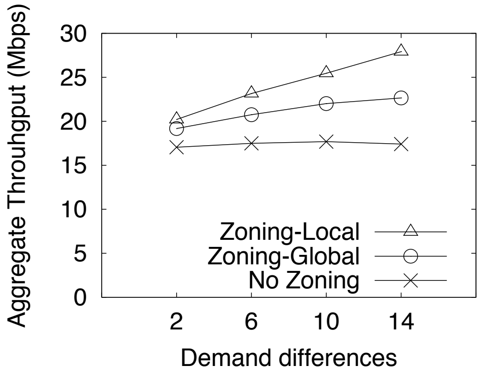
Table 4.1 summarizes the number of sub-channels allocated to each BS along with utilization and aggregate throughput measurements from the experiments. We observe that *dist* has the lowest utilization and therefore the lowest aggregate throughput. This is because it over-accounts for interference by just considering the vertex degrees in allocation. *deg* inherently penalizes vertices with high degree and allocates more resources to the others; it slightly outperforms $A^3$ in utilization and throughput (albeit at the cost of fairness). Fig. 4.14(a) and 4.14(b) plot the fairness for equal load and variable load (the loads are listed next to each BS in parentheses in Table 4.1), respectively. It is seen that $A^3$ consistently outperforms the other algorithms except topology 2 (equal load case) where it requires a fill-in edge. However, the restoration step of $A^3$ can account for under-utilization (due to the fill-in edge) helping it achieve the same utilization as the benchmark. In all other topologies, $A^3$ achieves the exact allocation as the benchmark (BM in Table 4.1) since they are naturally chordal.

***Zoning Benefits:*** We now present throughput measurements for $A^3$ - with and without zoning in Fig. 4.15(a). The baseline strategy is where each cell operates on all tiles with link adaptation. We observe that even without zoning, $A^3$ has significant gains over the baseline. The gains are further pronounced when zoning is employed, giving $A^3$ a throughput increase of 50% on average.

Next, we quantify the benefits of decoupling reuse zone demands in the network (local min.) against having a single reuse demand propagated to each contention region (global min.). For this



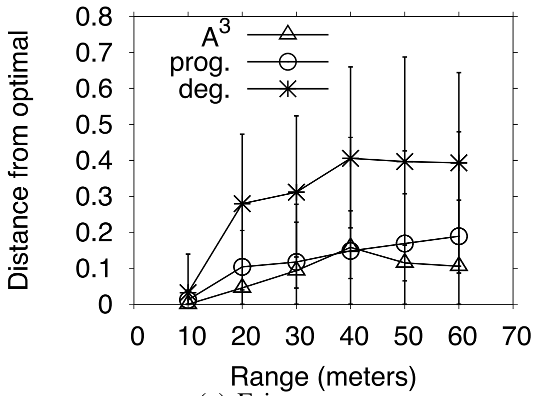
(a) Throughput



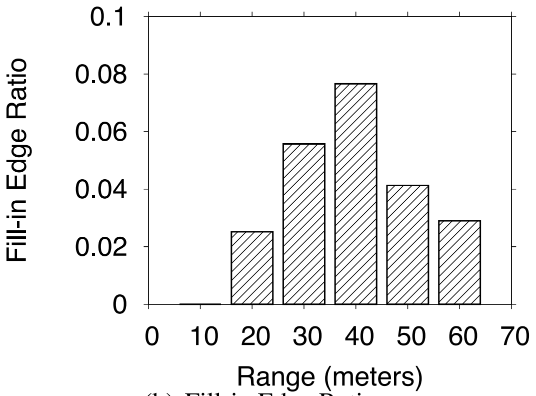
(b) Global vs Local Min.

Figure 4.15: Zoning Benefits of $A^3$.

experiment, we use topology 1 in Figure 4.13. We set equal reuse zone size demands for BS1 and BS2 (varied in each measurement) and a fixed demand of 4 symbols for BS3. Demand difference is defined as the difference between the common demands of BS1 and BS2 and the demand of BS3 (4 symbols). As we vary the demand difference from 2 to 14, we measure the aggregate throughput and present it in Fig. 4.15(b). It is seen that both global min. and local min. zoning result in increasing throughput as the demand difference increases. For the global min., although the operational size is the same (4 symbols), the high demands of BS1 and BS2 allow them to schedule their class *LA* clients over a larger set of resources (recall the transition zone in §4.5). Note that since class *LA* clients are likely to support a higher MCS than class *ISO* clients, having a large demand contributes to throughput gains (as compared to scheduling class *ISO* clients in the transition zone). For local min. zoning, the operational size for BS1 is significantly higher as compared to the global min. resulting in an increasing throughput gain over the global min. strategy. This shows FERMI's benefits from decoupling desired reuse zone sizes between different contention regions in the network.



(a) Fairness



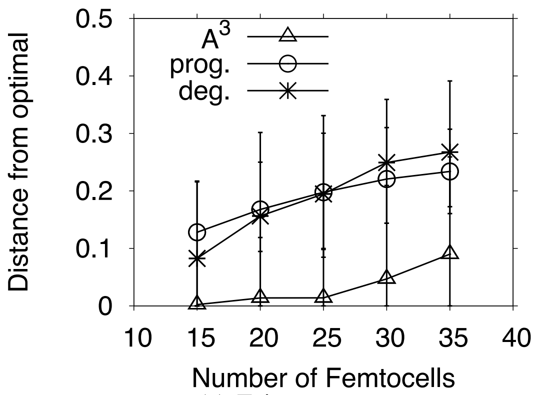
(b) Fill-in Edge Ratio

Figure 4.16: Effect of femtocell range on $A^3$'s fairness.

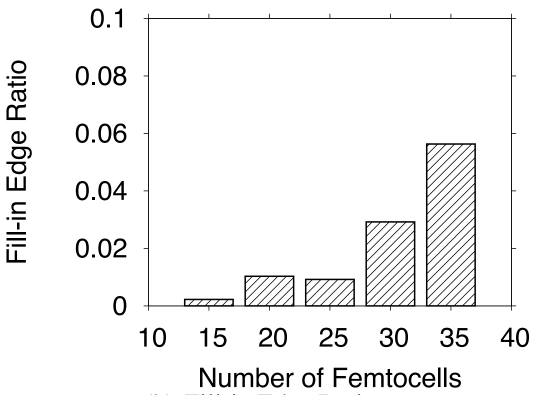
### 4.6.2 Evaluations with Simulations

**System Model and Metrics:** We implement a simulator to evaluate FERMI in comparison to its alternatives. The simulator incorporates a channel model proposed by the IEEE 802.16 Broadband Wireless Access Working Group for femtocell simulation [79]. This model captures wireless effects such as log-distance path loss, shadow fading and penetration loss, typical of indoor deployments. The SNR from the model is mapped to a MCS using a rate table from our testbed to compute throughput. The simulation area is a 7x7 grid where the distance between each grid point is 12 meters. In addition, the width and height of this area is 100 meters. We simulate a deployment by randomly choosing grid locations for each cell. We then randomly generate a client location for each cell and determine the conflict graph. We measure the overhead of each algorithm as the number of contiguous sub-channel chunks allocated per cell. In addition to overhead, we define the *fill-in edge ratio* to be the ratio of number of fill-in edges to the edges that are already present in the conflict graph. If the conflict graph is chordal, the fill-in edge ratio is 0. Next, we present our simulation results. Each data point is an average over results from 100 randomly generated topologies.

***Effect of Cell Range:*** Fig. 4.16(a) plots the effect of cell range on fairness. We observe that both heuristics (*prog* and *deg*) consistently deviate more from the benchmark as range increases. With increasing range, the number of sub-channels that a cell is assigned decreases (resources



(a) Fairness



(b) Fill-in Edge Ratio

Figure 4.17: Effect of number of femtocells on $A^3$'s fairness.

are shared among more cells). Recalling the fairness formula, a given difference in allocations (between the benchmark and the heuristics) becomes more pronounced with a smaller number of sub-channels assigned by the benchmark ($s_i$). Interestingly, $A^3$ exhibits an improvement in fairness after a particular range (while maintaining less than 0.15 distance from the benchmark). We find that the fill-in edge ratio is the main factor that affects $A^3$'s fairness (plotted in Fig. 4.16(b)). For small ranges, the graph contains some isolated vertices (very few cycles) and $A^3$ does not introduce fill-in edges. As range increases, cycles start to form and $A^3$ adds fill-in edges to make the graph chordal. However for further ranges, increased connectivity turns in favor of $A^3$ since the cycles happen rarely and the fill-in edge ratio decreases again.

***Effect of Number of Cells:*** We now fix a number of sub-channels (5) and a range (20 m.) and vary the number of cells. From Fig. 4.17(a), we see that $A^3$ consistently outperforms the other heuristics in terms of fairness and is within 0.1 distance of the benchmark due to the rare need for fill-in edges. The distance increases with the number of cells due to an increased likelihood of cycles; the trend again follows that of the fill-in edge ratio (plotted in Fig. 4.17(b)). For *deg* and *prog*, the distance also increases with the number of cells because of a reduced number of sub-channels per cell ($s_i$), similar to the effect of cell range.

***Effect of Number of Sub-channels:*** We now simulate 30 cells with a fixed range of 10 m. and vary the number of sub-channels in the spectrum. Fig. 4.18(a) shows that $A^3$ exhibits a constant

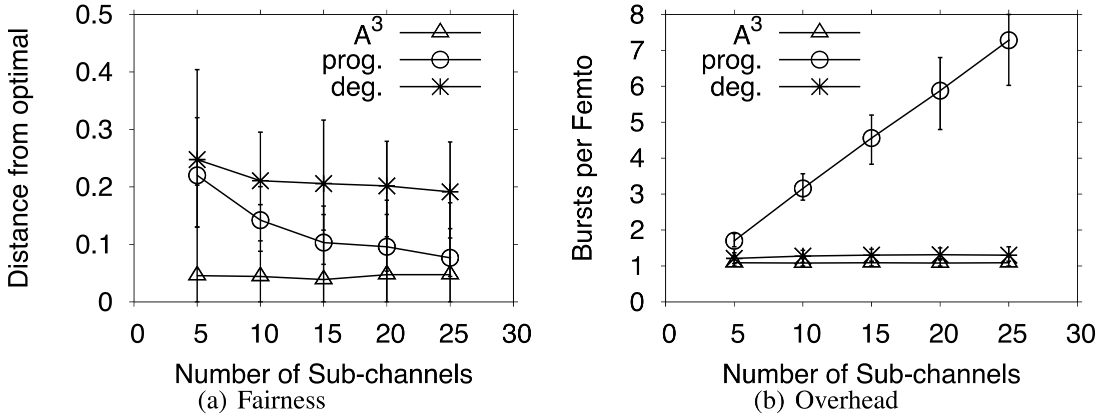


Figure 4.18: Effect of Number of Sub-channels on $A^3$'s fairness and overhead.

distance from the optimal. Since $A^3$'s performance is mainly influenced by the fill-in edge ratio, the number of sub-channels does not have a significant effect on $A^3$'s fairness. In addition, the distance for *prog* and *deg* decreases with an increased number of sub-channels due to the increase in number of sub-channels per cell ($s_i$). This makes the differences in allocations (between the heuristics and the benchmark) less pronounced as compared to when there are fewer sub-channels. Fig. 4.18(b) shows that the overheads for $A^3$ and *deg* are very close to 1 and do not change with the number of sub-channels. This shows that both strategies can assign a single contiguous set of sub-channels to the cells. However, *prog* tends to have an increasing overhead trend. Since *prog* assigns a fragmented set of sub-channels, the overhead increases with an increasing number of sub-channels that can be assigned to a cell.

***Effect of Zoning:*** In simulations with zoning, each cell has two clients: one that requires resource isolation (class *ISO*) and one that requires just link adaptation (class *LA*). We simulate 40 cells with range 10 m, and a frame structure having 30 sub-channels and 30 symbols. Among the cell population, we have three different types of reuse zone demands: i) high-demand cells that randomly demand a reuse zone size between 15 and 20 symbols ii) moderate demand cells that generate a demand value between 10 and 15 symbols and iii) low-demand cells with a generated demand between 5 and 10 symbols. We experiment by varying the fraction of the high-demand cells. Fig. 4.19 shows the total throughput achieved for each zoning strategy. It is seen that as

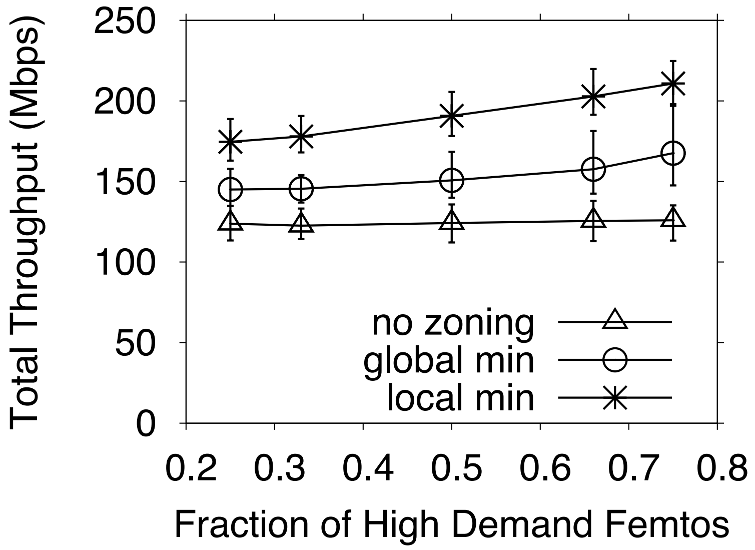


Figure 4.19: Effect of zoning on throughput

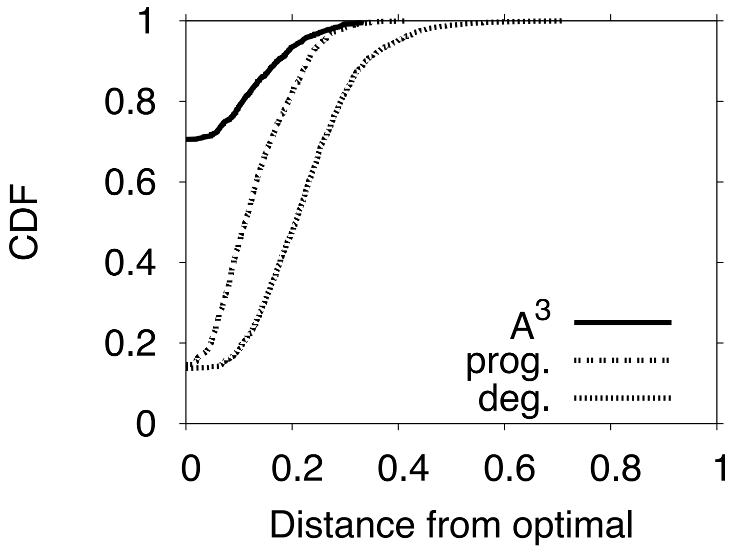


Figure 4.20: Overall fairness performance of $A^3$.

the fraction of high-demand cells increases, the throughputs for both zoning strategies increase. However, the gain of local min. over global min. is more with a higher fraction of high-demand cells. This is a natural artifact of vertices converging to a higher local demand as opposed to the global minimum value which is typically the same on average (generated by the low-demand vertices). The results reinforce FERMI's benefits of decoupling the reuse zone demands in different contention regions of the network (i.e. preventing a single demand from propagating).

***Overall Fairness:*** Finally, we present the CDF of the distance from the optimal as a cumulative set of all previously described simulations for the three algorithms considered, in Fig. 4.20. We mainly use the results with range 10 m. and 20 m., as these represent a more realistic deployment (given the entire area is 100x100 meters). The results provide an understanding of how fair a given algorithm is in practical deployments with a large set of variables (# femtos, # sub-channels, zoning etc.). It is seen that $A^3$ reaches the exact same allocation as the benchmark in about 70% of the topologies, which is far superior to the performance of the other heuristics. *deg* has the worst performance and reaches the benchmark allocation in only 10% of the topologies. *prog* does better than *deg* but still significantly underperforms $A^3$.

## 4.7 Conclusions

In this chapter, we design and implement FERMI, one of the first resource management systems for OFDMA femtocell networks. Resource management in femtocells offers a set of unique practical challenges (posed by the requirement for standards compatibility) that - to the best of our knowledge - had not been addressed to date. FERMI provides a complete resource management solution with several unique features. It uses measurement-driven triggers to classify clients into two categories, those that need resource isolation and those that do not. It incorporates a frame zoning structure that supports the graceful coexistence of clients from both categories. For purposes of interference mitigation, it allocates orthogonal sub-channels of the OFDMA spectrum with high utilization and low overhead. We implement FERMI on our WiMAX femtocell testbed and show via both experiments and simulations that its performance is superior to other conventional methods.

# Chapter 5

# Design and Implementation of an Integrated Beamformer and Uplink Scheduler for OFDMA Femtocells

Beamforming is a signal processing technique with numerous benefits. Unlike with omni-directional communications, it focuses the energy of the transmitted and/or the received signal in a particular direction. Although beamforming has been extensively studied on conventional systems such as WiFi, little is known about its *practical* impact on OFDMA femtocell deployments. Since OFDMA schedules multiple clients (users) in the same frame (in contrast to WiFi), designing intelligent scheduling mechanisms and at the same time harnessing gains from beamforming, is a challenging task.

Unlike downlink, we show that the integration of beamforming with uplink scheduling projects an interesting trade-off between beamforming gain on the one hand, and the power pooling gain resulting from joint multi-user scheduling on the other hand. This in turn makes the uplink scheduling problem even hard to approximate. To address this, we propose algorithms that are simple to implement, yet provably efficient with a worst case guarantee of half. We implement our solutions on a real WiMAX femtocell platform integrated with an eight-element phased array beamforming antenna. Evaluations from both prototype and trace-driven simulations show that our solution delivers throughput gains of over 40% compared to an omni-directional scheme.

## 5.1 Introduction

To meet the demands for increased capacity driven by the exponential growth in mobile data traffic [80], broadband network deployments are moving towards smaller cells – called femtocells – that use Orthogonal Frequency Domain Multiple Access (OFDMA). Femtocells inherit OFDMA and their synchronous access nature from macrocells; this allows for easier deployment and seamless operations with mobile clients.

The ability to focus energy in specific directions (called beamforming) using antenna arrays can serve as a valuable tool in enhancing the performance of OFDMA femtocells. While beamforming is extensively studied for WiFi systems (e.g., [9, 81]), the fundamental differences between WiFi and OFDMA – scheduling of multiple users in the same frame and the synchronous access in OFDMA – render such studies inapplicable to femtocells. The future releases of OFDMA systems (e.g., WiMAX 802.16m [66]) support beamforming as part of the standard. They allow for multiple beams at the base-band level (one beam per user) to be used on the data payload within the same frame (*user-level beamforming*). However, potential beamforming gains are limited as they support a limited number of antennas at femtocells (e.g., four in WiMAX and LTE-Rel.8) and the specific beamforming mechanisms vary from one standard to another.

A more flexible alternative to user-level beamforming is to employ *frame-level beamforming*, which applies a single beam (common to all users) to the entire frame (not only the data payload but also the control part) at the RF level. Since user-specific beams are not leveraged, such an approach may incur sub-optimal performance for the data payload since a beam pattern that "fits" all users has to be found. However, we identify that frame-level beamforming still has critical advantages: **(a)** it is realizable as a plug-and-play external RF beamformer and hence, is agnostic of the access technology (WiMAX, LTE, etc.); and **(b)** beamforming gain is not limited by the number of antennas supported by the standards. Even more interestingly, we show that it also offers a unique, complementary benefit for interference mitigation, in a multicell context (discussed in detail later in §5.3). However in realizing the true potential of frame-level beamforming, an essential first step is to integrate it with the scheduler at each femtocell.

The integration of frame-level beamforming with downlink scheduling can be accomplished by extending existing solutions [82, 83] - by running the scheduler for each beam and picking the one yielding the best objective. However, the corresponding uplink problem encounters two aspects unique to the uplink that make the problem challenging:

**Power pooling vs. beamforming:** When multiple users are jointly scheduled in a frame (i.e., the resources are shared), each user pools his transmit power on a smaller set of sub-channels (instead of using all sub-channels). This results in a higher user rate per sub-channel and hence higher frame throughput (*power pooling gain*). While beamforming improves a user's rate, using a single beam (common to users) for a frame may not yield the optimal performance for every user scheduled in the frame. Hence, the higher the number of users scheduled in a frame, the higher is the power pooling gain but lower is the beamforming gain, and vice versa.

**Batch scheduling:** For a femto base station (BS) to compute the uplink allocation, the scheduling requests have to come from the clients (as is the case in macrocells). Clients contend for sending such requests, which they send in response to changes in their buffer occupancies. This incurs both contention delay and overhead (multiple requests sent for a transmission buffer worth of data). A better option is to allow the BS to poll clients for their buffer occupancies at periodic intervals (e.g., polling service in WiMAX [62]). This avoids contention and reduces overhead. However, scheduling is now done for a group of frames jointly (batch scheduling) based on client buffer occupancies. More generically, even without polling, batch scheduling allows a user's data to be spread across multiple frames, thereby accentuating power pooling gains and hence, aggregate throughput (details in §5.4.1).

In this study, we consider the batch scheduling problem on the uplink for non-real-time traffic. While it is optimally solvable for omni-directional communications, we show that it is hard to even approximate when integrated with frame-level beamforming, where it also requires the determination of a beam for each frame. In addressing this problem, we make the following contributions.

- We propose simple but efficient algorithms for both continuous and discrete rate functions with a worst case guarantee of $\frac{1}{2}$.
- We implement our solutions on a real WiMAX femtocell platform that is integrated with an eight-element phased array antenna for beamforming.
- We conduct comprehensive over-the-air evaluations with both a prototype implementation and trace-driven simulations; results indicate an average gain of 40% over an omni-directional scheme, in practical settings.

The rest of the chapter is organized as follows. §5.2 presents a brief WiMAX overview and related work. In §5.3 and 5.4, we describe the motivation and the design of our system. §5.5 describes our algorithms. §5.6 contains implementation details and §5.7 shows evaluation results. We conclude in §5.8.

## 5.2 Background

**WiMAX:** While our solutions apply to OFDMA femtocells in general, our implementation is on WiMAX. Hence, we provide brief background on relevant WiMAX components (details in [62]). OFDMA divides the spectrum into multiple frequencies (sub-carriers) and several sub-carriers are grouped to form a sub-channel. The sub-carriers can be grouped in a contiguous, partially contiguous or fully distributed manner. Partially contiguous grouping, being the mandatory model, is alone implemented on most WiMAX devices and is hence, considered in our work. Here, sub-carriers forming a sub-channel are permuted to allow for a common transmission rate for a user on all sub-channels. This requires only a single channel feedback per user. Permuting also helps average out inter-cell interference.

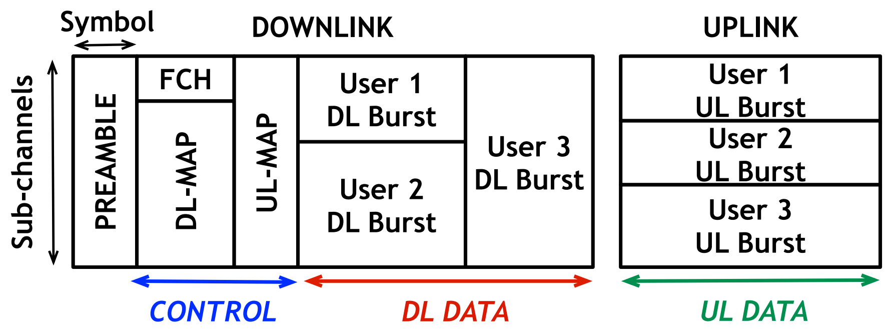


Figure 5.1: An illustration of the WiMAX Frame

A WiMAX frame is a two-dimensional template that is scheduled at the MAC with data to/from multiple mobile stations (MSs), across both time (symbols) and frequency (sub-channels). Data to/from users are allocated as rectangular bursts of resources in a frame (see Fig. 5.1). For e.g., the uplink allocation in Fig. 5.1 can be visualized to be at the sub-channel granularity. The frame consists of the preamble, control and data payload. While the preamble is used by the MS to lock on to the BS, the control consists of FCH (frame control header) and MAP. The BS, using MAP, indicates the schedule of transmissions both on the downlink and the uplink. The DL-MAP specifies the location of each burst, which MS it is intended for, and what modulation decodes it. Similarly the UL-MAP tells the MS where to place its data in the uplink and how to modulate it.

**Beamforming:** The ability to transmit (receive) signal energy in (from) specific directions is called beamforming. This is achieved by weighing the signals transmitted (received) from an antenna array in both magnitude and phase. The weight vector applied determines the specific beam pattern generated. Such weight vectors can be pre-determined and stored a priori (switched beamforming) or can be computed on the fly based on instantaneous channel feedback from the clients (adaptive beamforming).

To achieve low complexity, standards use code-book based beamforming, whereby weight vectors (pre-determined by the code-book) are employed and the one yielding the best SNR at the client is chosen. Since beams are enabled only for the data part, a beam vector suited for each client scheduled in a frame can be chosen, to encode data in the digital signal space (user-level beamforming). In contrast, applying a single beam physically at the RF level (frame-level beamforming), will provide beamforming for the entire frame, including the control part (advantageous in a multicell context as explained in §5.3). While its beamforming gain for the data part may be sub-optimal compared to user-level beamforming, frame-level beamforming is agnostic of the standards and thus, is not limited by them. Indeed, this has allowed us to integrate an eight-element phased array antenna with a WiMAX BS, although the current OFDMA BSs do not support beamforming and the next generation ones (e.g., 802.16m) support only four antennas. We consider frame-level beamforming for uplink, where the beam patterns emphasize reception in desired directions and could either correspond to physical directions covering the $360°$ azimuth (e.g., [81]) or be based on Gaussian code-books (e.g., [84]).

**Related Work:** Several works have studied the design of throughput [82, 83, 85, 86] and QoS [87, 88] schedulers for OFDMA. However, their focus has been on per-frame scheduling. While batch scheduling is not very different from per-frame scheduling on the downlink (in terms of optimization), the ability to leverage power pooling on the uplink makes the problem challenging and has not been addressed. While some studies (e.g., [89]) have looked at the design of polling intervals specifically for the uplink of WiMAX, they have not addressed the batch scheduling problem.

User-level beamforming has been jointly optimized with per-frame scheduling for OFDMA systems [90, 91]. However, the notion of frame-level beamforming and its various benefits have not been explored; its integration with uplink batch scheduling and the challenges it encounters have also not been considered. We contribute by designing and implementing an efficient uplink scheduler that integrates frame-level beamforming with batch scheduling for OFDMA.

## 5.3 Motivation

We motivate the importance of frame-level beamforming and batch scheduling for the uplink using measurements from an experimental testbed. We defer however, a detailed description of the testbed to §5.6.

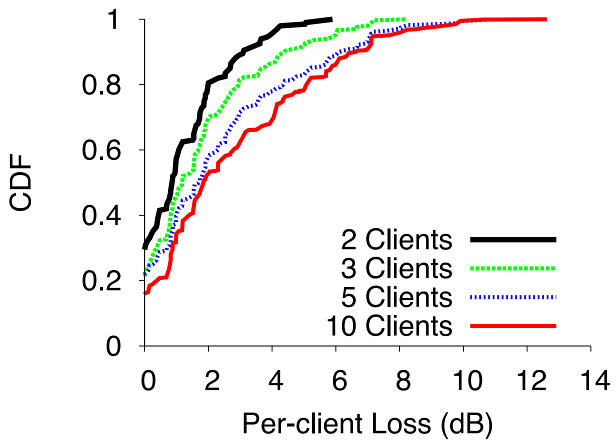


Figure 5.2: Sub-optimality of frame-level beamforming.

**Small Loss Compared to User-level Beamforming:** One drawback of frame-level beamforming is the constraint of using a single beam for all the users scheduled in a frame, thereby potentially limiting the per-user beamforming gain. In strong line-of-sight environments, where very few beams work well for each client, the sub-optimality of such a common beam could be significant. To understand this sub-optimality indoors, we placed a client at multiple (thirty) locations in the building and recorded the uplink SNR (as measured at the BS) with various beams for each of these locations. Then, different subsets of client locations were grouped together to emulate different sets of clients being scheduled in the same frame. For each of these subsets, a single beam was chosen (frame-level beamforming) that yielded the smallest aggregate loss compared to the case where the optimal beam for each client was chosen (user-level beamforming).

The CDF of the loss per client (in dB) is shown in Fig. 5.2. It is seen that even when five clients (reasonably high for femtocells) are multiplexed in the same frame, the loss is less than 2 dB for over 60% of the scenarios, indicating that the sub-optimality of using a common beam for a frame is not significant. This can be attributed to the multipath nature of the indoor environment, which allows for multiple beams to perform reasonably well for a given location, thereby increasing the possibility of finding a mutually *good* beam for multiple clients. However, as more and more clients are multiplexed together, the sub-optimality will tend to increase.

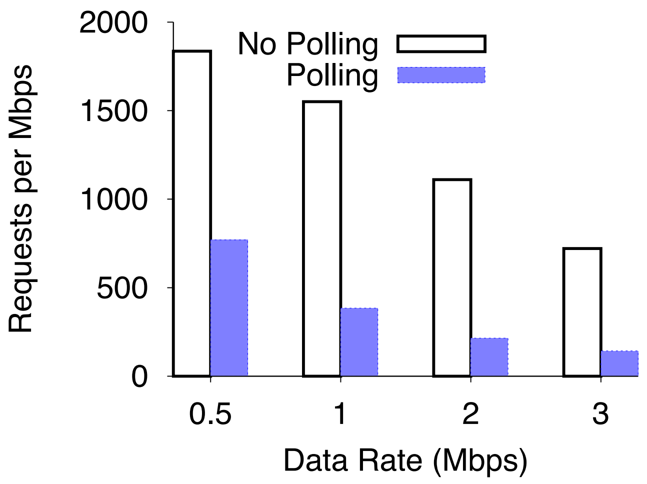
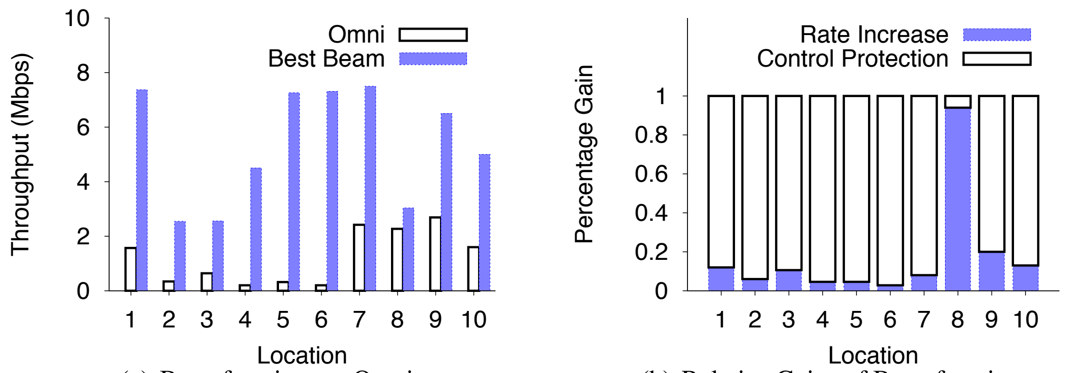


Figure 5.3: Relative overhead with batch scheduling.

**Reduced Overhead with Batch Scheduling:** Client requests for resources on the uplink typically experience contention, delay and overhead, with redundant requests being potentially sent out prior to transmission. To overcome these issues, typical OFDMA systems allow the BS to poll clients for data periodically (period adjusted based on traffic variations). We profile the relative overhead between client-initiated and BS-polled requests, by measuring the number of requests generated for a fixed amount of data for a single client, in Fig. 5.3. We see that BS polling reduces the overhead by as much as five folds. Further, the overhead does not grow with input rate unlike with client-initiated requests. With BS polling, buffer occupancy information of clients is available only once every polling interval. During this interval, frames can be scheduled one at a time or jointly as a batch, with the latter yielding better performance owing to joint optimization.

**Improved Decoding of Control Part:** One not-so-obvious benefit of frame-level beamforming is its ability to improve the decoding of the control part, through improved beamforming gain. As described earlier in §5.2, the control part carries vital information that helps clients decode and transmit their data payload. If the control part in a frame cannot be decoded by the clients (e.g., due to interference from other cells), the frame fails to deliver even a single byte of data. To understand the impact of interference on the control part, we experiment with two BSs that have one client associated with each. To make sure that the interference impact is only on the control part, we



(a) Beamforming vs. Omni (b) Relative Gains of Beamforming

Figure 5.4: Benefits of frame-level beamforming in a multicell scenario.

configure the two BSs to use orthogonal sets of sub-channels for their data payloads (i.e., data payload is protected from interference). While one BS is omni-directional, the other is equipped with a beamforming antenna. The client associated with the beamforming BS is placed at various locations to generate varying interference scenarios. In each of the locations, we measure the downlink throughput delivered at the client using each beam.

The throughputs delivered by the best beam and the omni beam are presented in Fig. 5.4(a). We see that there is a lot of room for improvement with beamforming ($\approx 7X$ on average) even when the data parts are immune to interference. There are two factors behind the observed throughput: (i) the rate (modulation) increase due to beamforming gain on the data part, and (ii) improved resilience of the control part and hence, reduced frame losses. Fig. 5.4(b) presents the relative contributions of these two components. We see that the improved decoding of the control part owing to frame-level beamforming is the more important component and provides the bulk of the gain in most scenarios. This clearly indicates that the performance of existing resource isolation solutions such as [8] can be enhanced even further, if the control part decoding is improved with the help of frame-level beamforming[1]. To realize the benefits of frame-level beamforming[2] and batch scheduling, we next present our design of an integrated beamformer and uplink scheduler – *iBUS*.

[1]While a scheme that combines an external common beam on the control part with user-level beamforming on the data part may be ideal, the varying size of the control part prevents its realization due to lack of standards support.

[2]Hereafter referred to as simply beamforming.

## 5.4 Design of iBUS

### 5.4.1 Overview

In batch scheduling, given a polling interval ($F$ frames) and the buffer occupancies of the clients, the problem is to effectively pack (schedule) the out-standing client data jointly over $F$ frames, with potentially different schedules across frames. When batch scheduling is integrated with beamforming, in addition to packing, one also needs to determine a beam for each frame in the batch. Picking a beam for a frame and a subset of clients (for data packing) are inter-twined since the beam choice for a frame will affect the rate supported by clients in that frame and hence, the amount of data that can be packed. Similarly picking a subset of clients will influence the beam choice since the sub-optimality of beamforming will vary with the subset of clients.

Two aspects make the integrated scheduling problem challenging: **(a) Power Pooling:** In OFDMA, when multiple clients are multiplexed in the same uplink frame, they pool their powers on a smaller set of sub-channels, thereby potentially supporting higher rates per sub-channel. Batch scheduling across multiple frames enhances power pooling further - spreading a client's data across multiple frames will require fewer sub-channels per frame, resulting in higher power pooling gain per frame. **(b) Beamforming:** Spreading data across multiple frames for each client will maximize the power pooling gain. However, this results in more clients being scheduled in the same frame, making it harder to find a *good* common beam for each frame; this increases the sub-optimality of beamforming. The scheduler's objective is to strike the right trade-off between these two components.

*Remarks:* We note that unlike uplink, in the downlink, there is no notion of power pooling (due to fixed BS total transmit power) and no signaling overhead that may necessitate batch scheduling. Hence, downlink is limited to per-frame scheduling, solutions for which exist even when a user's rate varies across sub-channels [82]. Further, its integration with beamforming can be realized by running existing schedulers for various beams and picking that schedule and beam, which maximizes the objective. However for the uplink, the combination of batch scheduling and power pooling with beamforming makes the problem challenging even when a user's rate does not vary across sub-channels.

### 5.4.2 Formulation and Hardness

The integrated uplink scheduling problem can be formally stated as the following non-linear integer program.

$$\text{ISP:} \quad \text{Maximize} \sum_i U_i(R_i)$$

$$\begin{aligned} R_i = \sum_{j,k} x_{i,j,k} \left\{ \sum_{\ell} y_{j,\ell} \cdot k \cdot r_{i,k,\ell} \right\} &\leq B_i,\ \forall i \in \mathcal{K} \\ \sum_{\ell} y_{j,\ell} &\leq 1,\ \forall j \in \mathcal{F} \\ \sum_{i,k} x_{i,j,k} \cdot k &\leq N,\ \forall j \\ \sum_{k} x_{i,j,k} &\leq 1,\ \forall i, j \end{aligned}$$

where $x_{i,j,k}$ and $y_{j,\ell}$ are binary indicator (output) variables indicating the assignment of $k$ contiguous sub-channels in frame $j$ to user $i$ and the assignment of beam $\ell$ to frame $j$ respectively. $\mathcal{K}$, $\mathcal{F}$, $\mathcal{L}$ indicate the set of users, frames in the batch and the available beams, respectively. The objective is to maximize the aggregate client utility, where the utility function can be any concave function (e.g., logarithmic function captures proportional fairness [85]) of the data scheduled and can vary from one client to another. The first constraint limits the net allocation to a user to be limited to its buffer occupancy $B_i$ (in bits). Further, it indicates the inter-dependence of power pooling (# sub-channels assigned, $k$) and beam chosen ($\ell$) on the user's rate per sub-channel ($r_{i,k,\ell}$ in bits). The remaining constraints are all conservation constraints indicating the (i) assignment of a single beam per frame, (ii) net allocation of sub-channels being limited to $N$ in a frame, and (iii) assignment of one contiguous set of sub-channels to a user in a frame, respectively.

We next show that it is hard to even obtain a PTAS (polynomial time approximation scheme) for the ISP problem. Specifically, we have the following result.

**Theorem 3.** *For some $\epsilon > 0$, there is no $(1-\epsilon)-$ approximation algorithm for ISP unless P=NP.*

*Proof.* Scheduling of multiple sub-channels in a frame across users with finite buffers and varying rates across sub-channels (SCF) has been studied in [82] and shown to not admit a PTAS, using a reduction from 3-bounded 3-matching problem. We next show that SCF is a special case of ISP.

Consider a simpler version of ISP, where the beam choices for the frames do not have to be determined but are given a priori ($\ell_j$ for frame $j$). Further, consider only one sub-channel per frame ($N = 1$) available for allocation in each of the $F$ frames. Since a user's ($i$) rate will vary from one beam to another and across frames ($r_{i,1,\ell_j}, \forall j$), the resulting ISP problem is now equivalent to an SCF problem with $F$ sub-channels (mapping from frames in ISP) in a frame and users with finite buffers and varying rates across the sub-channels, where the rate of user $i$ on sub-channel $c$ is $r_{i,c,\ell_c}$, $c \in [1, F]$. Hence, the desired result. $\square$

### 5.4.3 Components of iBUS

To address the above problem, we design iBUS, which consists of the following key components.

**Measurement of Beam SNRs:** If there are $|\mathcal{L}|$ beam patterns, $|\mathcal{L}|$ frames are used for measurement (one for each pattern). On the uplink part of each frame, the BS schedules all the active users such that their data spans all the sub-channels (number of time symbols allocated varies across users). Such a schedule can be visualized as user bursts being arranged vertically rather than horizontally as in Fig. 5.1 (see uplink). The resulting SNR ($\rho$) for each user with each beam is measured directly at the BS and corresponds to that when power is split on all sub-channels ($\rho(i, N, \ell), \forall i, \ell$).

**Rate Estimation using Power Pooling:** Rate tables are determined separately to identify the best modulation and coding rate (MCS) to employ for a given SNR that yields a specific loss rate (e.g., $< 0.1$). However, since the rate would vary with the number of sub-channels allocated (due to power pooling), a direct SNR measurement for each possible set of sub-channels for each

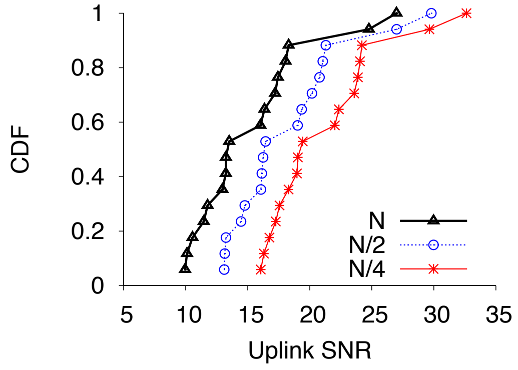


Figure 5.5: SNR Pooling Gain

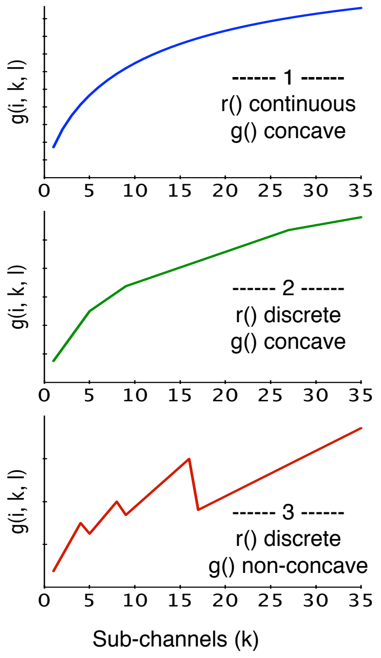


Figure 5.6: Pooling Rate Curves $g()$

beam ($\rho(i,k,\ell)$) would constitute significant overhead. Hence, based on the SNR measurement from all sub-channels ($\rho(i,N,\ell)$), we extrapolate the SNR for other subsets of sub-channels, taking power pooling into account. For e.g., if half of the sub-channels are allocated to a user, then its resulting SNR $\rho(i,\frac{N}{2},\ell) = \rho(i,N,\ell) + 3$ (in dB). Since the power gets distributed over $\frac{1}{2}$ of sub-channels, the per sub-channel power is doubled, corresponding to $+3$ in dB scale. In general, $\rho(i,\frac{N}{\alpha},\ell) = \rho(i,N,\ell) + 10\log_{10}(\alpha)$, $\alpha \in [1, \frac{N}{N-1}, \ldots, \frac{N}{1}]$. Fig. 5.5 validates the modeling of power pooling by measuring the SNR in practice for a client, when data is transmitted on different sets of sub-channels.

From $\rho(i,k,\ell)$, one can obtain $r_{i,k,\ell}$ using the rate table. Essentially, every user ($i$) has a power-pooling rate curve ($g(i,k,\ell)$) as a function of number of sub-channels ($k$) and the beam chosen ($\ell$). Depending on the nature of SNR-rate mapping, these curves may or may not be concave. If continuous Shannon rates are employed, then we have $g(i,k,\ell) = k \cdot \log_2(1+\frac{N}{k}\rho(i,N,\ell))$, which is a concave function (see Fig. 5.6). If discrete rate tables are used then $g(i,k,\ell) = k \cdot r_{i,k,\ell}$ results in a piece-wise linear (between SNR thresholds) function, where the mapping $r_{i,k,\ell} \leftarrow \frac{N}{k}\rho(i,N,\ell)$ is determined by the rate table. Depending on the rate-SNR thresholds, $g(i, k, \ell)$ may still be concave or not (Fig.5.6 shows $g()$ for two such examples). As we shall see in §5.5, depending on the nature of $g()$ (concave or arbitrary), different algorithms will be required.

**Buffer Estimation using Polling:** The BS polls all its clients towards the end of the current polling interval to estimate their buffer occupancies for scheduling during the next polling interval. Note that if the total buffer occupancy of all clients is less than the total frame resources in the polling interval, then there will be under-utilization. Further, the schedules computed for the polling interval will be relevant only if the SNRs are relatively static during that interval. Hence, the polling interval is decided based on both the coherence time of SNR values (reasonably high for indoor static clients) and the typical transmission buffer size of clients (details in §5.6).

**Schedule Determination:** Once the BS has all the information needed to determine the batch schedule, it executes its algorithms. The batch schedule consists of two parts. The first part is a schedule for $F - |\mathcal{L}|$ frames ($|\mathcal{L}| << F$), where the ISP problem is solved over $F - |\mathcal{L}|$ frames. The latter part is a schedule for the remaining $|\mathcal{L}|$ frames, and serves as the measurement interval to obtain the SNR information for the next polling interval. Hence, in the latter part, the beam choices for the $|\mathcal{L}|$ frames are fixed (one beam each) during measurement, and allocation is done only across time symbols by employing the same algorithm used for solving ISP, albeit with some fixed variables. While the main purpose of the second part is measurement, one can also optimize the packing of client data that remains after the first $F - |\mathcal{L}|$ frames, given the beam choices.

Solving the ISP problem, i.e., determining data packing along with the beam choices over a given set of frames, forms the core of our solution and is described next.

## 5.5 Algorithms in iBUS

Given the hardness of the scheduling problem ISP, we focus on algorithms that have a provable worst case guarantee and are simple to implement. Our algorithms can be classified based on the nature of the power pooling rate curve.

```
1:  INPUT: Buffer occupancy $B_i, \forall i \in \mathcal{K}$; rates $g(i, k, \ell), \forall i, k \in [1, N], \ell \in \mathcal{L}$
2:  OUTPUT: Beam choices $\ell_j, \forall j \in \mathcal{F}$; per-frame user allocation $A_{ij}, \forall i, j$
3:  for $f \in [1 : |\mathcal{F}|]$ do
4:    $O_{max} = 0$
5:    for $\ell_f \in [1 : |\mathcal{L}|]$ do
6:      $D_i \leftarrow B_i, \forall i$;;
        ;
        ;
        $a_{ij} = A_{ij}, \forall i, j \in [1, f-1]$ and $a_{ij} = 0\ \forall i, j = f$
7:      for $k = 1 : N$ do
8:        $i^* = \arg\max_{i \in \mathcal{K}} \Big\{ U_i \Big( \sum_{j=1}^{f} g(i, a_{ij}, \ell_j) + \min\big(g(i, a_{if}+1, \ell_f)$
            $-g(i, a_{if}, \ell_f), D_i\big)\Big) - U_i\big(\sum_{j=1}^{f} g(i, a_{ij}, \ell_j)\big)\Big\}$
9:        if $i^* \neq \emptyset$ then
10:       $D_{i^*} \leftarrow D_{i^*} - \min\big(g(i^*, a_{i^*f}+1, \ell_f)$
            $-g(i^*, a_{i^*f}, \ell_f), D_{i^*}\big)$
11:       $a_{i^*f} \leftarrow a_{i^*f} + 1$
12:       else break endif
13:     end for
14:     $O_f = \sum_{i \in \mathcal{K}} U_i \Big( \sum_{j=1}^{f} g(i, a_{ij}, \ell_j) \Big)$
          $-U_i \Big( \sum_{j=1}^{f-1} g(i, a_{ij}, \ell_j) \Big)$
15:     if $O_f > O_{max}$ then
16:     $O_{max} = O_f$; $\ell_{max} = \ell_f$; $A_{if} = a_{if}, \forall i$
17:     endif
18:   end for
19:   $\ell_f = \ell_{max}$; ; ;
      $B_i \leftarrow B_i - \min(g(i, A_{if}, \ell_f), B_i), \forall i$
20: end for
```

**Algorithm 5**: Integrated Scheduler: iBUS1

### 5.5.1 Concave Rate Curves

We present two greedy algorithms called iBUS1 and iBUS2, outlined in Algs. 5 and 6, respectively. In iBUS1, resource allocation and beam selection are performed one frame at a time in the batch, sequentially. For a given frame (iteration), based on the remaining data available for the users, resource allocation is performed for every choice of the beam, and the beam yielding the best aggregate marginal utility for the frame is chosen (steps 14-16). For resource allocation within each frame given a beam (steps 6-13), iBUS1 assigns each sub-channel to the user who yields the highest marginal utility taking the users' buffer status into account (step 8). For concave utility functions, such an allocation is indeed optimal. Further, allocation based on the marginal utility also helps multiplex multiple users in the same frame thereby maximizing the benefits of power pooling. Once the best beam and its corresponding resource allocation are determined (steps 14-17) for the current frame, the user buffers are updated (step 19) and the procedure is repeated for the remaining frames in the batch, sequentially. iBUS1 has a time complexity of $O(|\mathcal{F}||\mathcal{L}||\mathcal{K}|N)$.

The key difference between iBUS1 and iBUS2 is that while iBUS1 performs resource allocation within each frame in isolation, iBUS2 performs joint resource allocation across all frames considered in an iteration. Hence, in addition to enabling power pooling within each frame, it allows a user's data to be spread across multiple frames thereby pooling the user's power across frames as well. This in turn results in a more efficient data packing and hence, higher aggregate throughput. Specifically, when considering a beam for a given frame ($f$) during a considered iteration, resource allocation is performed for all frames till the current frame jointly (steps 6-13) to determine the resulting utility for the beam. The beam yielding the highest utility is then chosen for that frame (steps 15-19). Hence, each iteration of resource allocation now involves determining not only the user to whom a sub-channel must be allocated but also in which frame ($\in [1, f]$, step 8). Note that once the beam is chosen for a given frame, user buffers do not have to be updated for the next frame since resource allocation will then be re-performed. Thus, for every newly considered frame, user buffers are re-initialized (step 6). Essentially, resource allocation for all frames till the one under

```
1:  INPUT: Buffer occupancy $B_i, \forall i \in \mathcal{K}$; rates $g(i, k, \ell), \forall i, k \in [1, N], \ell \in \mathcal{L}$
2:  OUTPUT: Beam choices $\ell_j, \forall j \in \mathcal{F}$; per-frame user allocation $A_{ij}, \forall i, j$
3:  **for** $f \in [1 : |\mathcal{F}|]$ **do**
4:     $O_{max} = 0$
5:     **for** $\ell_f \in [1 : |\mathcal{L}]$ **do**
6:        $D_i \leftarrow B_i, \forall i;;$
          ;
          ;
          $A_{ij} = 0 \, \forall i, j = [1, f]; k_j = N, \forall j \in [1, f]$
7:        **while** $k_j \neq 0, \exists j \in [1, f]$ **do**
8:           $(i^*, j^*) = \arg\max_{i \in \mathcal{K}, j \in [1,f]} \Big\{ U_i \Big( \sum_{m=1}^{f} g(i, A_{im}, \ell_m)$
                   $+ \min\{g(i, A_{ij} + 1, \ell_j) - g(i, A_{ij}, \ell_j), D_i\})$
                   $- U_i(\sum_{m=1}^{f} g(i, A_{im}, \ell_m)) \Big\}$
9:           **if**$(i^*, j^*) \neq \emptyset$ **then**
10:          $D_{i^*} \leftarrow D_{i^*} - \min\left(g(i^*, A_{i^*j^*} + 1, \ell_{j^*})\right.$
                 $\left. - g(i^*, A_{i^*j^*}, \ell_{j^*}), D_{i^*}\right)$
11:          $A_{i^*j^*} \leftarrow A_{i^*j^*} + 1;;$
             ;
             ;
             $k_{j^*} \leftarrow k_{j^*} - 1$
12:          **else** break **endif**
13:       **end while**
14:    **end for**
15:    $O_f = \sum_{i \in \mathcal{K}} U_i \Big( \sum_{j=1}^{f} g(i, A_{ij}, \ell_j) \Big)$
16:    **if** $O_f > O_{max}$ **then**
17:    $O_{max} = O_f; \ell_{max} = \ell_f$
18:    **endif**
19:    $\ell_f = \ell_{max}$
20: **end for**
```

**Algorithm 6**: Integrated Scheduler: iBUS2

consideration is done mainly for the purpose of determining the beam yielding the highest utility for the considered frame. The actual resource allocation is the one that is computed during the final frame of iteration, where the allocation for all the frames in the batch is jointly determined along with the beam for that frame. While iBUS2 enables power pooling both within and across frames, the joint resource allocation across frames results in an additional time complexity of $O(|\mathcal{F}|)$ and hence a net time complexity of $O(|\mathcal{F}|^2|\mathcal{L}||\mathcal{K}|N)$.

While iBUS2 results in a better performance than iBUS1, we now show that *even* iBUS1's worst case performance can be bounded. We provide some definitions on matroid and sub-modularity that are relevant for the proof.

*Partition Matroid:* Consider a ground set $\Psi$ and let $S$ be a set of subsets of $\Psi$. $S$ is a matroid if, (i) $\emptyset \in S$, (ii) If $P \in S$ and $Q \subseteq P$, then $Q \in S$, and (iii) If $P, Q \in S$ and $|P| > |Q|$, there exists an element $x \in P \backslash Q$, such that $Q \cup \{x\} \in S$. A partition matroid is a special case of a matroid, wherein there exists a partition of $\Psi$ into components, $\phi_1, \phi_2, \ldots$ such that $P \in S$ if and only if $|P \cap \phi_i| \leq 1,\ \forall i$.

*Sub-modular function:* A function $f(\cdot)$ on $S$ is said to be sub-modular and non-decreasing if $\forall x, P, Q$ such that $P \cup \{x\} \in S$ and $Q \subseteq P$ then,

$$\begin{aligned} f(P \cup \{x\}) - f(P) &\leq f(Q \cup \{x\}) - f(Q) \\ f(P \cup \{x\}) - f(P) &\geq 0, \quad \text{and } f(\emptyset) = 0 \end{aligned}$$

**Theorem 4.** *iBUS1's worst case performance is within $\frac{1}{2}$ of the optimum.*

*Proof.* The sub-optimality of maximizing a sub-modular function over a partition matroid using a greedy algorithm of the form $x = \arg\max_{x \in \phi_i} f(P \cup \{x\}) - f(P)$ in every iteration was shown to be bounded by $\frac{1}{2}$ in [92]. We will now show that iBUS1 is such an algorithm, with our scheduling objective corresponding to a sub-modular function to obtain the desired result.

Let the ground set be composed of the following triplets.

$$\Psi = \{(j, \ell_j, \mathbf{a}_j) : \quad j \in [1 : |\mathcal{F}|], \ell_j \in [1 : |\mathcal{L}|], \\ \mathbf{a}_j = \cup_{i \in \mathcal{K}} a_{ij} \text{ s.t. } \textstyle\sum_i a_{ij} \leq N\}$$

Now $\Psi$ can be partitioned into $\phi_j = \{(j, \ell_j, \mathbf{a}_j) : \ell_j \in [1 : |\mathcal{L}|], \mathbf{a}_j = \cup_{i \in \mathcal{K}} a_{ij} \text{ s.t. } \sum_i a_{ij} \leq N\}$, $\forall j$. Let $S$ be defined on $\Psi$ as a set of subsets of $\Psi$ such that for all subsets $P \in S$, we have (i) if $Q \subseteq P$, then $Q \in S$; (ii) if element $x \in P \backslash Q$, then $Q \cup \{x\} \in S$; and (iii) $|P \cap \phi_j| \leq 1, \ \forall j$. This means that $S$ is a partition matroid. Further, any $P \in S$ will provide a feasible schedule with at most one feasible allocation and beam choice for each frame, thereby allowing the partition matroid to capture our scheduling problem. Our scheduling objective is given as,

$$\begin{aligned} f(P) &= \sum_{i \in \mathcal{K}} \mu_i(P) \\ \text{where, } \mu_i(P) &= U(\min\{\sum_{j:(j,\ell_j,\mathbf{a}_j) \in P} g(i, a_{ij}, \ell_j), B_i\}) \end{aligned}$$

It can be seen that if $Q \subseteq P$, then $\mu_i(Q) \leq \mu_i(P)$. Hence, for an element $(j, \ell_j, \mathbf{a}_j)$ such that $P \cup \{(j, \ell_j, \mathbf{a}_j)\}$ forms a valid schedule, it follows that $f(P \cup \{(j, \ell_j, \mathbf{a}_j)\}) - f(P) \leq f(Q \cup \{(j, \ell_j, \mathbf{a}_j)\}) - f(Q)$. This results both from the potential buffer limitation in subsequent frames as well as the concave nature of the utility function. This establishes that the function $f(P)$ is indeed sub-modular. Further, our scheduling problem aims to maximize this non-decreasing sub-modular function over a partition matroid. Hence, if the optimal allocation corresponding to every beam were given by some oracle for each frame, then picking the beam yielding the highest marginal utility for a frame in iBUS1 (steps 14-17) would correspond to determining

$$(j, \ell_j^*, \mathbf{a}_j^*) = \arg \max_{(j,\ell_j,\mathbf{a}_j) \in \phi_j} \{f(P \cup \{(j, \ell_j, \mathbf{a}_j))\}) - f(P)\}$$

Thus, the sub-optimality of $\frac{1}{2}$ would then follow from the result in [93].

Note that, there are exponential allocations possible that satisfy $\sum_i a_{ij} \leq N$ for every beam choice. Hence, bypassing the oracle assumption would require us to find the optimal allocation for a given beam, which is a problem in itself. However, since the utility functions are concave, this would correspond to a concave optimization problem, which iBUS1 solves optimally by employing a steepest gradient approach based on maximum marginal utility (steps 7-13). Hence, iBUS1 is able to bound its sub-optimality by $\frac{1}{2}$. $\square$

```
1: INPUT: Buffer occupancy $B_i, \forall i \in \mathcal{K}$; rates $g(i,k,\ell), \forall i, k \in [1,N], \ell \in \mathcal{L}$
2: OUTPUT: Beam choices $\ell_j, \forall j \in \mathcal{F}$; per-frame user allocation $A_{ij}, \forall i, j$
3: for $f \in [1 : |\mathcal{F}|]$ do
4:    $O_{max} = 0$
5:    for $\ell_f \in [1 : |\mathcal{L}]$ do
6:       $D_i \leftarrow B_i, \forall i;;$
         ;
         ;
         $a_{ij} = A_{ij}, \forall i, j \in [1, f-1]$ and $a_{ij} = 0 \; \forall i, j = f$
7:       Run an FPTAS for multiple-choice knapsack problem $FA(f, \ell_f)$; Obtain $a_{if}, \forall i$
8:       $O_f = \sum_{i \in \mathcal{K}} U_i \left( \sum_{j=1}^{f} g(i, a_{ij}, \ell_j) \right)$
            $-U_i \left( \sum_{j=1}^{f-1} g(i, a_{ij}, \ell_j) \right)$
9:       if $O_f > O_{max}$ then
10:      $O_{max} = O_f$; $\ell_{max} = \ell_f$; $A_{if} = a_{if}, \forall i$
11:      endif
12:   end for
13:   $\ell_f = \ell_{max}$; ; ;
      $B_i \leftarrow B_i - \min(g(i, A_{if}, \ell_f), B_i), \forall i$
14: end for
```
**Algorithm 7**: Integrated Scheduler: iBUS3

### 5.5.2 Arbitrary Rate Curves

For arbitrary power pooling rate curves, we present the following algorithm iBUS3, which is a modified version of iBUS1. Specifically, iBUS3 is similar to iBUS1 in picking the beam yielding the highest marginal utility for each frame. However, its frame allocation for a given beam choice is significantly different. This is because if the power pooling rate curves are not concave, then allocations based on marginal utilities will not work. Hence, allocations to users in each frame must be made as a subset of sub-channels instead of individual sub-channels. Thus, we model the allocation problem for a given frame and beam choice $(f, \ell_f)$ with arbitrary rate curves as the following multiple choice knapsack problem (MCKP), which is a NP-hard problem in itself.

$$\begin{aligned} FA(f,\ell_f): \qquad \text{Maximize} & \sum_i \sum_{k\in\mathcal{N}_i} u_{ik}(f,\ell_f)x_{ik}(f) \\ \text{s.t.} & \sum_i \sum_{k\in\mathcal{N}_i} k\cdot x_{ik}(f) \le N \\ & \sum_{k\in\mathcal{N}_i} x_{ik}(f) = 1;\ \forall i \end{aligned} \tag{5.1}$$

where $x_{ik}(f) \in \{0,1\}$ and $\mathcal{N}_i \in \{0,1,\ldots,N\}$.

Essentially, there are $N+1$ allocations ($\mathcal{N}_i$) possible for each user and the MCKP problem is to pick exactly one allocation for each user such that the total frame allocation does not exceed $N$. The user's profit or utility for each allocation is computed based on the current and previous frames' allocations as follows:

$$\begin{aligned} u_{ik}(f,\ell_f) &= \sum_{i\in\mathcal{K}} U_i\left(\sum_{j=1}^{f-1} g(i,a_{ij},\ell_j) + \min\{g(i,x_{ik}(f),\ell_f),B_i\}\right) \\ \text{where,} \qquad & a_{ij} = k:\ \text{s.t. } x_{ikj}=1,\ \forall i,j\in[1,f-1] \end{aligned}$$

Note that the user's buffer occupancy is automatically accommodated in its profit. Now a FT-PAS based on dynamic programming from [94] can be employed to efficiently solve $FA(f,\ell_f)$ to within $(1-\epsilon)$ of the optimal at a time complexity of $O(\frac{|\mathcal{K}|N}{\epsilon})$. This results in a net time complexity of $O(\frac{|\mathcal{F}||\mathcal{L}||\mathcal{K}|N}{\epsilon})$ for iBUS3.

In establishing a bound on the worst case performance of iBUS3, we employ the following lemma proved in [95].

**Lemma 1.** *If the sub-modular function is only $\alpha-$ approximable, then the approximation guarantee of greedy maximization changes to $\frac{\alpha}{p+\alpha}$, where the maximization is subject to a $p-$ independence system.*

**Theorem 5.** *iBUS3 provides an approximation guarantee of $\frac{1}{2} - \epsilon$.*

*Proof.* Note that matroids are 1-independence systems (see [96] for an exposition). Given that the FPTAS yields a $(1-\epsilon)$ approximation in computing the frame allocation and hence the sub-modular function, we have $\alpha = 1 - \epsilon$ and hence the resulting performance guarantee of iBUS3 reduces to $\frac{1}{2} - \epsilon$. □

*Remarks:* While we have shown that our algorithms have an approximation guarantee of half for both concave and arbitrary power pooling rate curves, one might wonder how much further can this guarantee be improved without increasing the complexity significantly. Unfortunately, the answer is not optimistic. We had shown that ISP does not admit a PTAS and hence a $(1 - \epsilon)$ approximation. Further, recall from §5.4.2 that single frame scheduling with varying sub-channel rates and finite user buffers (SCF) is a special case of our ISP problem. In [82], it was shown that one can improve the guarantee for SCF to 0.63 albeit through a complex LP relaxation and rounding procedure, where an exponential number of subsets is involved in the LP formulation. In light of this, we believe that our greedy algorithms strike a good balance between both performance and complexity. Further, §5.7 reveals their close-to-optimal performance in practice, on average, compared to their worst case guarantees of half.

## 5.6 Implementation

Fig. 5.7 depicts our system design. The testbed consists of a WiMAX femtocell platform and commercial clients (USB dongles attached to laptops). Our experiments are conducted over-the-air

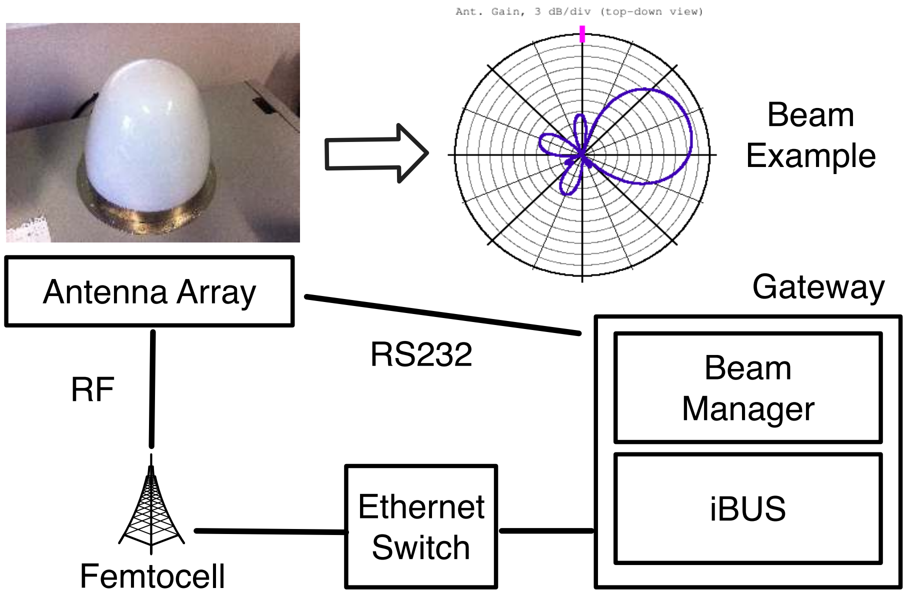


Figure 5.7: iBUS system architecture.

(10 MHz bandwidth) with an experimental license from FCC. We have an eight-element phased array antenna designed by Fidelity Comtech, attached via a RF cable to the BS. The array generates 16 beam patterns of $45°$ each, spaced $22.5°$ apart to cover the entire azimuth of $360°$. We also have a Linux PC for the gateway functionality required for WiMAX. In WiMAX, the gateway manages the service flows needed to transmit/receive data to/from the clients.

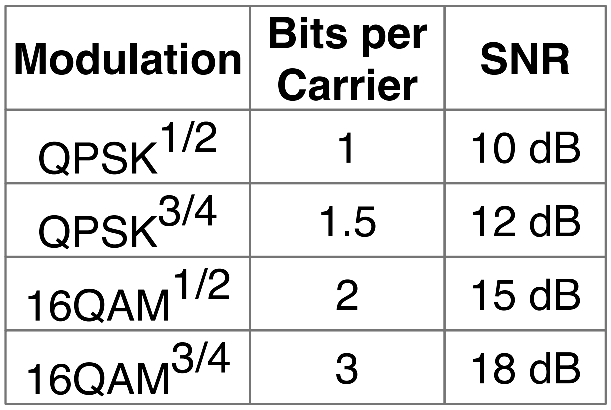
The BS and the gateway communicate using sockets via an Ethernet switch. When the BS decodes user data on the uplink, it passes it to the application at the gateway (*iperf*) via the switch. In addition, the switch is used to pass the client buffer information and the measured beam SNRs to the gateway. The gateway both implements our algorithms in Java, and at the same time controls the array by a serial port (RS232) application that we have developed in C.

When the gateway receives the buffer and SNR information, it executes the iBUS algorithms and sends the computed batch schedule to the BS, one frame at a time. A beam corresponding to a frame in the batch is applied to the array just before the corresponding schedule is sent to the BS. However, note that there is inevitably a delay before a particular beam is applied by the antenna following the gateway command. We measured this delay to be $\approx$ 6ms on average (8ms max.). Since each WiMAX frame is 5ms, applying one beam for a single frame is difficult. To circumvent this, each schedule corresponding to a frame in the batch is *extended* over 20 frames at the BS, by taking into account the resources from the 20 frames jointly. In addition, each beam $l_i$ is applied 10ms (two frames) prior to schedule $i$. With this a priori beam application, when the BS starts executing schedule $i$, the hardware is ready to transmit with beam $l_i$. One can alternatively implement the algorithms and the beam management component at the BS, for tighter beam and schedule synchronization. However, this would add complexity and overhead to the BS, making the deployment less practical. Our design realizes a light-weight, standards-compatible OFDMA beamforming system that does not result in significant processing overhead at the BS. In addition, since the gateway is agnostic of the underlying technology, we believe that it can flexibly be extended to other OFDMA systems such as LTE, which use a similar notion of sub-channels.

In our implementation, the BS polls the clients for buffer occupancies every 100 frames, thereby forming the batch scheduling interval. Since there are 16 beam patterns, 16 frames are used to measure the beam SNRs of each client. Selecting a large polling interval will not sufficiently capture the channel diversity (e.g., the SNRs can change in the interim), making the computed schedule sub-optimal. Conversely, a short polling interval will not sufficiently reduce the contention overhead for buffer requests. We have verified in our experiments that the SNRs with each beam are relatively stable during a 100 frame (500ms) interval, and that the client buffers are large enough to hold pending data.

## 5.7 Performance Evaluation

In this section, we evaluate iBUS using both our prototype implementation and trace-driven simulations. To isolate the gains achieved with power pooling (i.e., packing) and beam selection, we propose two simpler versions of iBUS1 that do not require beam adaptation across frames: **(a) iBUS-mean** schedules multiple clients in the same frame like iBUS1 but uses a fixed beam for every frame. The fixed beam is chosen to minimize the mean loss from the best beams of each



| Modulation | Bits per Carrier | SNR |
|---|---|---|
| QPSK$^{1/2}$ | 1 | 10 dB |
| QPSK$^{3/4}$ | 1.5 | 12 dB |
| 16QAM$^{1/2}$ | 2 | 15 dB |
| 16QAM$^{3/4}$ | 3 | 18 dB |

Figure 5.8: MCS Thresholds used by our femtocells.

client. Each client $i$ has SNR ${\rho^i}_{max}$ with some beam yielding the max. SNR from its perspective. If client $i$ has SNR ${\rho^i}_l$ for beam $l$, then the loss is ${\rho^i}_{max} - {\rho^i}_l$. iBUS-mean picks beam $l \in \mathcal{L}$ that minimizes the mean loss over all clients, **(b) iBUS-weighted** uses a fixed beam as in **iBUS-mean** but the loss is weighted based on the buffer occupancy of each client, to account for asymmetric application rates. In this case, the loss for a client is calculated as $({\rho^i}_{max} - {\rho^i}_l) * B_i$. Again, the beam $l \in \mathcal{L}$ that minimizes this weighted mean loss over all clients is fixed for all frames in the batch. We compare our algorithms to an omni-directional scheme (labelled **omni**) that leverages power pooling by employing the same packing procedure as iBUS1 (but each frame is transmitted omni-directionally).

### 5.7.1 Prototype Evaluation

In WiMAX, the modulation to be used by the clients for uplink communications is determined by the BS. Since the BS directly measures the received SNR from clients, it uses a threshold-based table to determine the modulation for each client. Fig. 5.8 enlists the thresholds used in our platform.

With such discrete rate tables, all SNRs in between two consecutive thresholds map to the MCS with the lower threshold (similar to WiFi). For example in Fig. 5.8, the BS instructs the clients to use $QPSK^{\frac{3}{4}}$ for all SNRs between 12dB and 15dB. Depending on the thresholds (de-

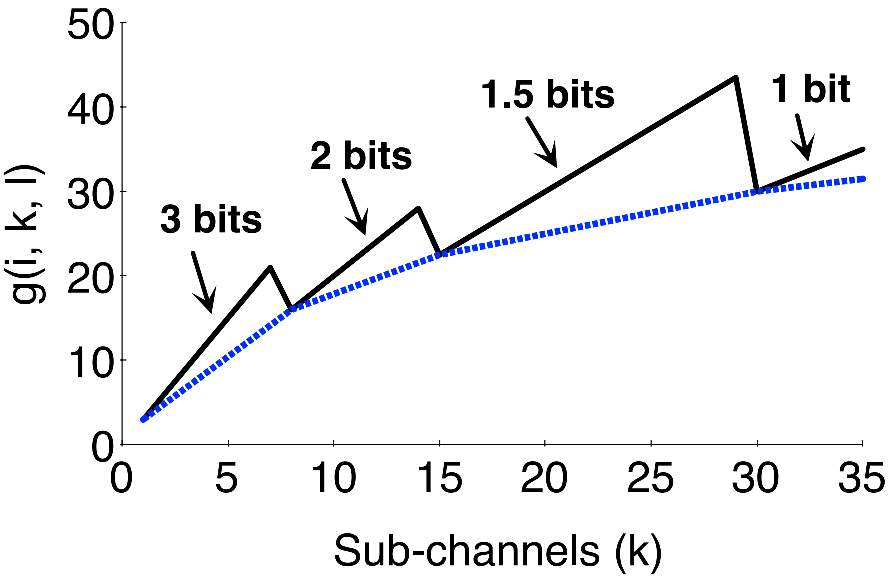


Figure 5.9: $g()Transformation$

termination of which is not specified by the standard and is left to the vendor), this may result in non-concave client power-pooling curves ($g()$). Indeed with our femtocells, we observed that such non-concavity occurs in practice. While iBUS3 can address non-concavity, we choose to implement iBUS1, iBUS-mean and iBUS-weighted on our prototype because of their low complexity and ease of implementation.

To make iBUS1 compatible with non-concave $g()$, we transform a given $g()$ into a corresponding concave function. Fig. 5.9 depicts an example transformation where $g()$ is computed as the product of number of sub-channels and the number of bits that a given MCS encodes. To do the transformation, we first determine the number of sub-channels where each MCS transition occurs. This can easily be done by computing the SNR for a given number of sub-channels using the power pooling formula. Then, the transition points are connected by lines with an appropriate slope. Note that the resulting concave function assumes that the error rate at a given MCS level increases in conjunction with the SNR drop as the power is distributed over more sub-channels. Even if the table maps two SNRs to the same MCS, the lower SNR will likely result in a higher bit (or symbol) error rate, reducing the throughput in practice. This observation is also leveraged by prior

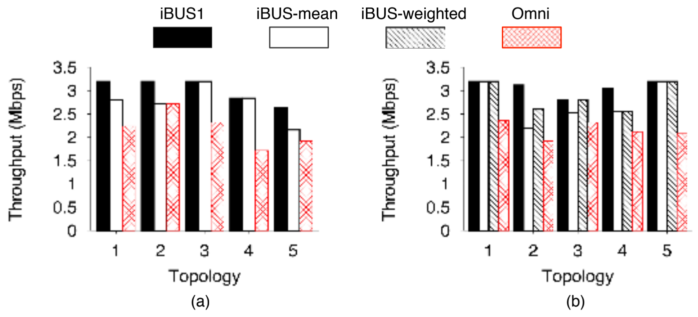


Figure 5.10: Performance results with iBUS in real experiments. Equal rates (left), different rates (right).

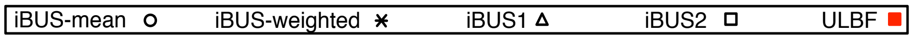
studies (e.g., [97]). In our experiments, we observed that the throughput predicted by the concave transformation matched the observed client throughput in practice reasonably well. On average, the predicted throughput was 94% of the actual throughput.

We run each scheme on several topologies, created by placing three clients in different locations. The utility function of a user is equal to the total data scheduled for that user in the batch (i.e., $U_i(R_i) = R_i$). The clients initiate UDP flows to the BS, generated by iperf. Fig. 5.10(a) shows the aggregate throughput for five sample topologies of clients with equal application rates (iBUS-weighted yields the same fixed beam as iBUS-mean and hence, is not considered). We observe that iBUS1 outperforms omni by 42%, on average. In addition, the beam adaptation component in iBUS1 helps outperform iBUS-mean by 15% - 25%. However, cases exist where iBUS-mean performs as well as iBUS1, indicating that the proper choice of a *fixed* beam may provide significant gains, depending on the topology. Fig. 5.10(b) shows the aggregate throughput for clients with different application rates. We observe that iBUS1 outperforms omni by 45% on average. For iBUS-weighted, we see that it can improve performance over iBUS-mean in some cases (e.g., topology 2 and 3). In addition, there are again cases where iBUS-mean and iBUS-weighted perform as well as iBUS1. To summarize, while iBUS1 consistently outperforms the omni-directional scheme by ≈45%, an appropriate choice of a fixed beam can also yield significant gains in practice.

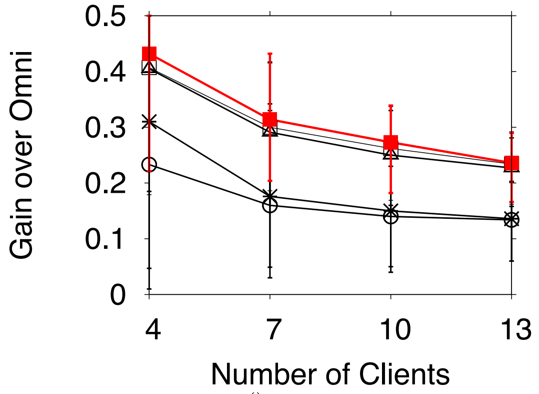
### 5.7.2 Trace-driven Simulations

To evaluate with a large client population, we resort to SNR traces measured at 30 different client locations in our testbed. We consider 100 topologies, generated by picking random subsets of the client locations from our traces. To evaluate iBUS1 and iBUS2 with MCS tables in practice (i.e., discrete $r()$), we employ the same $g()$ transformation as before. We also consider a continuous $r()$ function (Shannon). We introduce an alternative scheme labelled **ULBF**, that mimics user-level beamforming and serves as a loose upper bound. In ULBF, packing is executed by spreading a client's data over multiple frames as in iBUS2. However, rather than using a common beam for a frame, each client can operate using its best beam (i.e., beam with the max. SNR.) even within a frame.

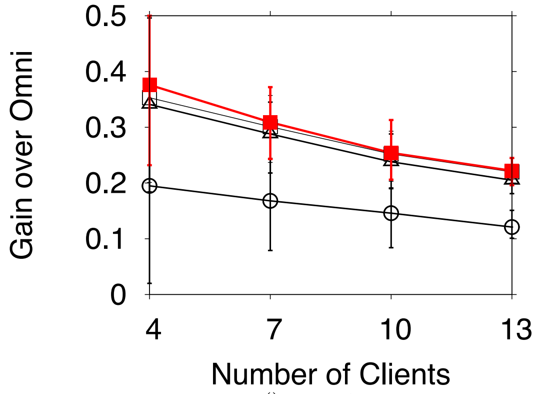
Fig. 5.11 shows the the mean gains achieved over omni for a varying number of clients. With discrete rate tables (Fig. 5.11(b) and 5.11(c)), we observe that iBUS1 delivers around 40% gain over the omni scheme for four clients (inline with our prototype evaluation). However, iBUS2 and ULBF do not significantly improve the performance further. Even though there is increased power pooling and hence higher SNR, the effective rate is not substantially higher than iBUS1 due to the nature of discrete MCS tables. In addition for all the schemes, the gains tend to drop with increasing numbers of clients. With a larger client population, finding a common beam comes at the cost of significant SNR loss from their best beam for some clients. On the other hand, the omni benefits from client diversity (increased chance of a client having high SNR with the omni beam) and is able to deliver increased throughput. However, since the femtocell client populations are typically small ($<8$), appreciable gains can still be attained. Thus, we conclude that iBUS1 is a low-complexity alternative to user-level beamforming in practical deployments. Since the increased power pooling and beamforming gains of user-level beamforming cannot fully be harnessed with discrete MCS tables, iBUS1 serves as a flexible alternative (i.e., number of antennas is not limited) with similar performance. Fig. 5.11(d) and 5.11(e) show the mean gains over the omni scheme for the continuous rate (Shannon) function. This time, we observe that iBUS2 outperforms iBUS1 by



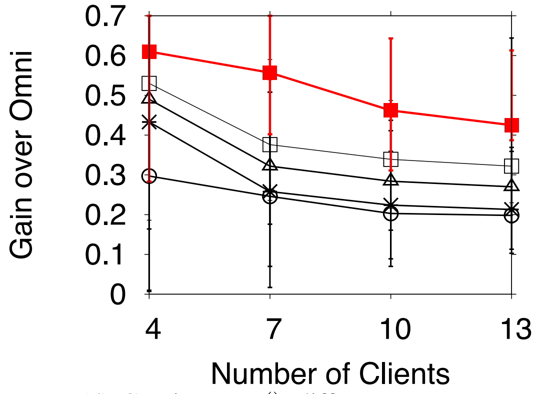
(a) Legend for the graphs below



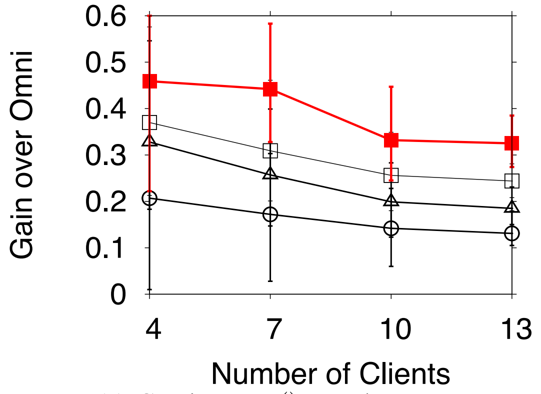
(b) Discrete $r()$, different rates



(c) Discrete $r()$, equal rates



(d) Continuous $r()$, different rates

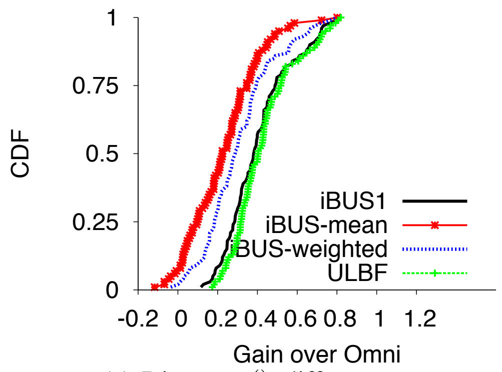


(e) Continuous $r()$, equal rates

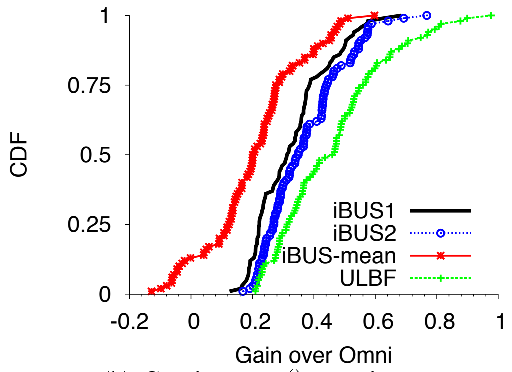
Figure 5.11: Performance of iBUS obtained with trace-driven simulations.

10% - 20% due to its better power pooling capability across frames. Further, the gap with respect to ULBF is around 30% - 40%. This behavior is expected since the continuous rate function will benefit more from better power pooling and proper beam selection, compared to a discrete rate table.

In the above experiments, we have demonstrated the mean gains over the omni scheme. To better visualize the distribution, we plot the CDF of the gain for four clients. Fig. 5.12(a) shows that the maximum gain with iBUS1 can be as high as 60% - 80% for 20% of the scenarios. Further, an interesting observation that is revealed is that in 10% - 15% of the scenarios, iBUS-mean performs worse than omni. Recall that iBUS-mean picks a common beam that minimizes the mean loss from the best beam of each client. In such scenarios, some clients may have to operate with a beam,



(a) Discrete $r()$, different rates



(b) Continuous $r()$, equal rates

Figure 5.12: CDF of the gain distribution with four clients. iBUS2 is not shown in (a) for clarity (has similar performance as iBUS1).

yielding a SNR that is several dB less than their optimal beam and even worse than the omni-directional beam. However, adapting the beam in iBUS1 addresses this issue to deliver improved performance. Similar observations hold for both types of rate functions (see Fig. 5.12(b)).

## 5.8 Conclusions

We investigate the joint problem of uplink batch scheduling and frame-level beamforming (shown to not admit a PTAS) in OFDMA femtocells. We propose a set of algorithms with a worst case approximation guarantee of $\frac{1}{2}$. Our algorithms address the unique tradeoff between scheduling multiple users in a frame (benefiting from power pooling) and harnessing the gains from directional beams (by applying a common beam for multiple users). We implement the algorithms on a real WiMAX femtocell platform. Our prototype evaluation and trace-driven simulations demonstrate that our algorithms provide benefits of over 40% over an omni-directional scheme. To our best knowledge, this is the first study that integrates and evaluates frame-level beamforming on an actual OFDMA femtocell platform.

# Chapter 6

# Computing While Charging: Building a Distributed Computing Infrastructure Using Smartphones

Every night, a large number of idle smartphones are plugged into a power source for recharging the battery. Given the increasing computing capabilities of smartphones, these idle phones constitute a sizeable computing infrastructure. Therefore, for a large enterprise which supplies its employees with smartphones, we argue that a computing infrastructure that leverages idle smartphones being charged overnight is an energy-efficient and cost-effective alternative to running tasks on traditional server infrastructure. While parallel execution and scheduling models exist for servers (e.g., MapReduce), smartphones present a unique set of technical challenges due to the heterogeneity in CPU clock speed, variability in network bandwidth, and lower availability compared to servers.

In this chapter, we address many of these challenges to develop CWC—a distributed computing infrastructure using smartphones. Specifically, our contributions are: (i) we profile the charging behaviors of real phone owners to show the viability of our approach, (ii) we enable programmers to execute parallelizable tasks on smartphones with little effort, (iii) we develop a simple task migration model to resume interrupted task executions, and (iv) we implement and evaluate a prototype of CWC (with 18 Android smartphones) that employs an underlying novel scheduling algorithm to minimize the makespan of a set of tasks. Our extensive evaluations demonstrate that the performance of our approach makes our vision viable. Further, we explicitly evaluate the performance of CWC's scheduling component to demonstrate its efficacy compared to other possible approaches.

## 6.1 Introduction

Today, a number of organizations supply their employees with smartphones for various reasons [98]; a survey from 2011 [12] reports that 66% of surveyed organizations do so and many of these organizations have 75–100% of their employees using smartphones. For example, Novartis [99] (with 100,000 employees in 140 countries) handed out smartphones for its employees to manage emails, calendars, as well as information about health issues; Lowe's [100] did so for its employees to have real time access to key product information and to allow managers to handle administrative tasks.

We argue that in such settings, an enterprise can harness the aggregate computing power of such smartphones, to construct a distributed computing infrastructure. Such an infrastructure could reduce both the capital and energy costs incurred by the enterprise. First, this could reduce the number of servers to be purchased for computing purposes. For example, Novartis awarded a contract of $2 million to IBM to build a data center for their computational tasks [101]. If they could exploit the smartphones handed out to their employees to run some portion of their workload, it is conceivable that the cost of their computing infrastructure could have been reduced. Due to recent advancements in embedded processor design, now a smartphone can replace a normal desktop or a server running a dual core processor for computation. According to Nvidia, their Quad Core CPU, Tegra 3, outperforms an Intel Core 2 Duo processor in number crunching [102]; for other workloads, one can expect the performance of the two CPUs to be comparable.

Our second motivation for the smartphone-based computing infrastructure is that the enterprise could benefit from significant energy savings by shutting down its servers by offloading tasks to smartphones. The power consumed by a commercial PC CPU such as the Intel Core 2 Duo is 26.8W [103] at peak load. In contrast, a smartphone CPU can be over 20x more power efficient, e.g., the Tegra 3 has a power consumption of 1.2W [103, 104]. Since their computing abilities are similar, it is conceivable that one can harness 20 times more computational power while consuming the same energy by replacing a single server node with a plurality of smartphones. In fact, to harness the energy efficiency of embedded processors, cloud service providers are already pushing towards ARM-based data centers [105] to reduce energy costs while offering the same or even improved performance.

The construction and management of such a distributed computing infrastructure using smartphones however, has a number of associated technical challenges. We seek to articulate these challenges and build an efficient framework towards making such a platform viable. In particular, the biggest obstacles to harnessing smartphones for computing are the phone's battery-life and bandwidth. If a smartphone is used for computing during periods of use by its owner, we run the risk of draining its battery and rendering the phone unusable. Further, today data usage on 3G carriers are typically capped, and thus, shipping large volumes of data using 3G is likely to be impractical. Thus, our vision is to use these smartphones for computing when they are being charged, especially at night. During these periods, the likelihood of active use of the phone by its owner will be low. Moreover, the phone will be static and, will likely have access to WiFi in the owner's home (today, 80% of the homes in the US have WiFi connectivity [106]); this will both reduce fluctuations in network bandwidth, and allow the transfer of data to/from the smartphones at no cost.

We name our framework CWC, which stands for computing while charging. To realize CWC, we envision the use of a single server, connected to the Internet, for scheduling jobs on the smartphones and collecting the outputs from the computations. The scheduling algorithms executed on the server are lightweight, and thus, a rudimentary low cost PC will suffice. Smartphones are only utilized for computation when being charged. If an owner disconnects the phone from the power outlet, the task is suspended, and migrated to a different phone that is connected to a power outlet. Towards building CWC, our contributions are as follows:

- **Profiling charging behaviors:** While an enterprise can possibly mandate that its employees charge their smartphones when they are not being actively used, we examine the typical charging behaviors of smartphone owners. Using an Android application that we develop, we gather charging statistics on the phones of 15 volunteers. Our results demonstrate that, on average, a typical user charges his phone for up to 8 hours every night.

- **Scheduling tasks on smartphones:** As a fundamental component of CWC, we design a scheduler that minimizes the makespan of completing the jobs at hand, taking into account both the CPU and bandwidth available for each a smartphone. Since the optimal allocation of jobs across phones is NP-hard, we design a greedy algorithm for the allocation, and show via experiments that it outperforms other simple conceivable heuristics.

- **Migration of tasks across phones:** CWC executes tasks on smartphones only when they are being charged. Tasks are suspended if phones are unplugged during execution. We design and implement an approach to efficiently migrate such tasks to other phones that are plugged in.

- **Automation of task executions:** The typical means of running an application on smartphones is to have the phone's owner download, install, and run the application. However, we cannot rely on such human intervention to leverage smartphones for a computing infrastructure. We demonstrate how task executions on phones can be realized in a completely automated manner. Note that, while we recognize the potential privacy implications of running automated tasks on smartphones, we simply assume here that an enterprise would not run malicious tasks on its employees' smartphones. Improving the isolation of tasks implemented in typical smartphone operating systems is beyond the scope of our work.

- **Preserving user experience:** Blind execution of tasks for extended durations on a smartphone being charged, can prolong the time taken for the phone to fully charge. We show experimentally that intensive use of a phone's CPU can delay a full charge by 35%. We design and implement a CPU throttling mechanism, which ensures that task executions do not impact the time to charge a phone.

- **Implementation and experimentation:** Finally, we implement a prototype of CWC and conduct extensive experiments on a testbed of 18 Android phones. Specifically, we showcase the efficacy of the scheduling and task migration algorithms within CWC.

## 6.2 Related Work

To the best of our knowledge, no prior study shares our vision of tapping into the computing power of smartphones. However, some efforts resemble certain aspects of CWC.

**Smartphone testbeds and distributed computing platforms.** Publicly-available smartphone testbeds have been proposed [107, 108] to enable smartphone OS and mobile applications research. CrowdLab [109] and Seattle [110] provide resources on volunteer devices. There also exist systems where users voluntarily contribute idle time on their PCs to computational tasks (e.g., [111]). In contrast, our vision is not for the smartphone infrastructure to be used for research and testing, but to enable energy and cost savings for real enterprises. Moreover, the issues that we address have not been considered by these efforts.

Flavors of MapReduce for smartphones have been implemented (e.g., [112]). However, such efforts do not address the issues of detecting idle phone usage, partitioning tasks across phones with diverse capabilities. They do envision using phones to offer a distributed computing service.

**Participatory sensing.** Recent studies such as [113], advocate the collective use of the sensing, storage, and processing capabilities of smartphones. With participatory sensing [114], users collect and analyzing various types of sensor readings from smartphones. Unlike these efforts, a distinguishing aspect of CWC is that the data to be processed does not originate from the phones. In addition, CWC allows the execution of a variety of tasks unlike above, where typically a fixed task (sensing) is supported. Finally, CWC seeks to leverages compute resources on smartphones, rather than tapping human brainpower via users' phones [115].

**Measurements of smartphones.** There have been measurement studies [116, 117] to characterize typical network traffic and energy consumption on smartphones. In contrast, our focus is on developing a scalable platform for gathering measurements from the phones in CWC. Several prior studies [118, 119] have observed that phones are idle and are being charged for significant periods of time every day. We are however the first to recognize that these idle periods can be harnessed to build a distributed computing infrastructure.

**Provisioning tasks on cloud services.** Prior efforts have also tried to identify when the use of cloud services is appropriate (e.g., [120]), discuss the challenges involved in using them (e.g. [121]), or present solutions for provisioning applications (e.g., [122]). However, these efforts focus on traditional server-based cloud services. Prior efforts on managing resources in the cloud (e.g., [123]) do not tackle challenges associated with provisioning tasks on heterogenous resources nor deal with the variability of wireless links.

## 6.3 Feasibility Study

In this section, we examine (a) the challenges associated with building a smartphone-based computing infrastructure and (b) the potential savings in capital and energy costs offered by such an infrastructure.

### 6.3.1 Challenges

**Adequacy of computing power:** The smartphone infrastructure is only attractive if it can effectively accomplish the computing tasks undertaken on today's servers. Due to rapid advances in embedded processor technologies, numerous smartphones with Quad Core CPUs are emerging [124]. Some of these CPUs offer clock speeds of up to 2.5 Ghz per core (Qualcomm snapdragon quad-core APQ8064) and their computational capabilities are *beyond that* of typically used server machines. To compare the performance of a smartphone CPU with that of typical desktop and server CPUs, we refer to the well known CoreMark benchmarks [125]. Figure 6.1 (borrowed from [104, 125]) shows

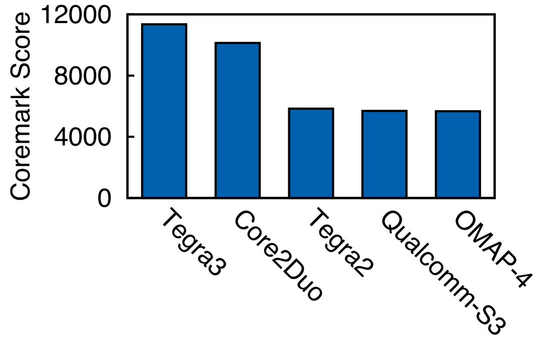


Figure 6.1: CoreMark score of different CPUs.

the performance of major smartphone CPUs against a widely used desktop and server CPU—the Intel Core 2 Duo; the higher the CoreMark score, the better. We see that while the Nvidia Tegra-3 outperforms the Intel Core 2 Duo, the Core 2 Duo outperforms the other processors by more than 50%. This shows that state-of-the-art smartphones like Samsung Galaxy S-3 (running on Tegra-3 CPU) can only replace a single-core server or desktop machine. In our smartphone testbed, most of the smartphones are running on Tegra-2, Snapdragon S-3, and Ti OMAP-4 CPUs; in spite of this, we can execute a typical server job with two or three (of these older) smartphones.

**Availability of idle task execution periods:** Beyond the dramatic improvements in their compute capabilities, smartphones are attractive for a computing service because their resources are unused for long periods of time. Most users leave their smartphones idle overnight while the batteries are being recharged. When being recharged, a smartphone typically runs only light-weight background jobs (e.g., downloading e-mail) that require minimal computation and intermittent network access. Scheduling jobs on a phone during such periods is unlikely to impact smartphone owners.

To identify and utilize idle periods, we have implemented an Android application to *profile* the charging behavior of users (Apple iOS also supports the core functionality required). The application tracks three states on every phone: (a) *plugged*: when the user is charging the phone, (b) *unplugged*: when the phone is detached from the charger, and (c) *shutdown*: when the phone is powered off. When there is a change in state (i.e., *unplugged* to *plugged*), the application logs

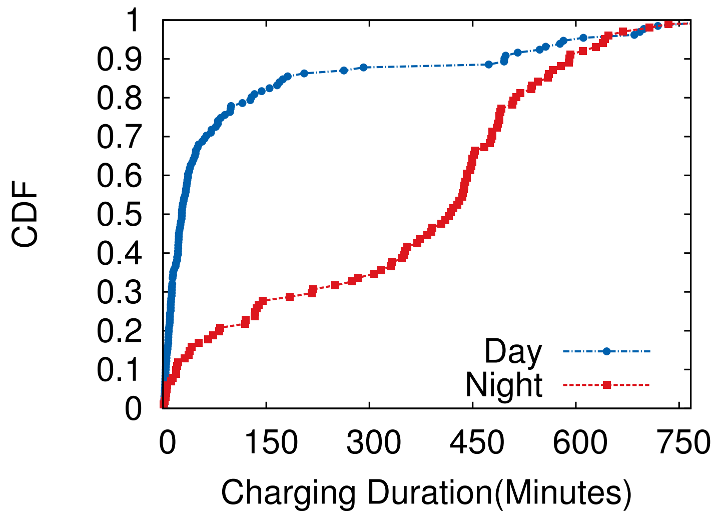
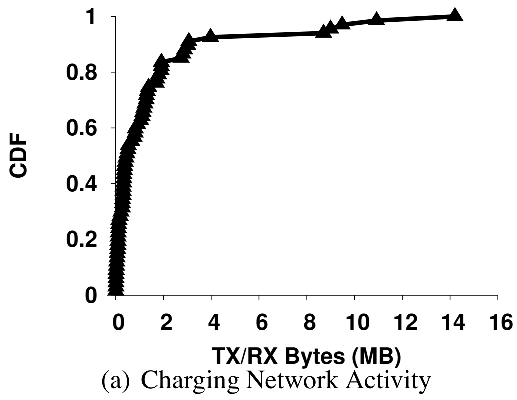


Figure 6.2: Charging duration for both day and night.

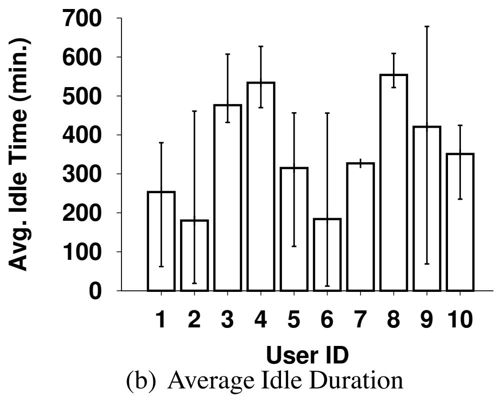
the change to a server along with a timestamp (of the user's local timezone). In addition, it logs the total bytes transmitted and received over all wireless interfaces (cellular and WiFi) when in the *plugged* state; this statistic is reset every time the phone newly enters the *plugged* state. The server parses the log files and computes for every charging interval of a particular user: (a) the duration of the interval, and (b) the number of bytes (transmitted and received) during that interval.

We conduct a study of realistic user behavior by having 15 volunteers (real users) install our application on their phones and gathering statistics. In Fig. 6.2, we plot the distribution of charging interval lengths, with every interval assigned to day or night; if the *plugged* state occurs between 10 p.m. and 5 a.m. of a user's local time, that interval is considered to be in the night, else in the day. We observe that the median charging interval is around 30 minutes and 7 hours long, at day and night respectively. In addition, there are fewer charging intervals in the night. This suggests that users charge their phones for an uninterrupted stretch of several hours at night. During the day, charging is interrupted frequently, resulting in a large number of short intervals.

We now focus only on the charging intervals at night. Fig. 6.3(a) plots the CDF of data transfers over all night charging intervals. Although the user is unlikely to be actively using the phone, there is background data in the form of periodic e-mail checks, push notifications from news and social



(a) Charging Network Activity



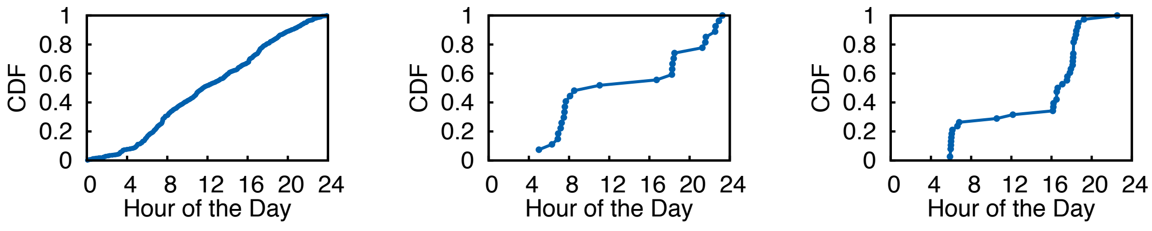
(b) Average Idle Duration

Figure 6.3: The study of charging durations (with less than 2MB network activity) across 10 volunteers.

media, etc.. However, we find that the total network activity is less than $\approx$ 2 MB for 80% of all night charging intervals.

Using the network activity data, we identify night charging intervals that can be considered idle to be the ones in which the data transfer is less than 2 MB. Fig. 6.3(b) shows that the users, on average, have at least 3 hours of idle charging at night. However, the characteristics highly depend on the individuals. For example, users with the highest idle durations (users 3, 4, and 8) have lower variability in their behavior; this suggests that they regularly charge their phones for 8 to 9 hours at night. In addition, users very rarely turn their phones off while charging (only 3% of the logs are in the *shutdown* state). The consistent low load at night (as also reported by [126]) suggests that idle usage patterns occur in large-scale settings as well. Given this, we speculate that this will provide an overlap of long idle charging times across users, yielding several operational hours for computing, without disturbing users' routine activities.

Next, we also examine the *plugged* and *unplugged* activity of each user. Our goal is to identify periods where, the phones are most likely to be unplugged. In our setting, we consider *unplugging* as a failure since we do not execute tasks when a phone is *unplugged*. Figure 6.4(a) plots the CDF of *unplugged* activity (failure) for all users. It is seen that the likelihood of failure between 12 AM to 8 AM is less than 30%. In CWC, we simply migrate such failed tasks to phones that are still



(a) Unplugging behavior of users (b) Sample User 1 (c) Sample User 2

Figure 6.4: Availability of smartphones for CWC task scheduling.

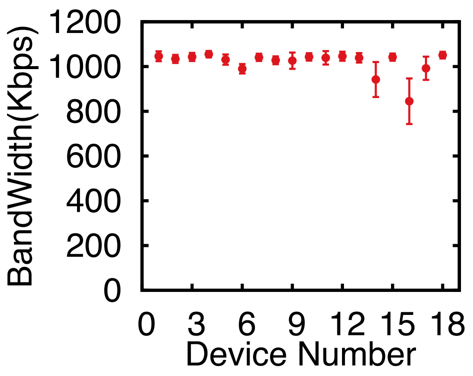


Figure 6.5: WiFi Network Stability.

plugged in (discussed later).

Profiling an individual user behavior's can allow the prediction of device specific failures (this can help since tasks can be migrated to phones that are less likely to fail at the time of consideration). Figures 6.4(b) and 6.4(c) show the unplugging behaviors of two representative users from our study; as evident, the two behaviors differ. However, interestingly, the likelihoods of failure in both cases are very low between 12 A.M. and 6 A.M. It shoots up between 6 A.M. and 9 A.M. when people begin to use their phones. During the day, the likelihood of *unplugged* activity is high; it decreases when phones are charged again at night.

In summary, our study suggests that the charging behaviors of users are typically consistent at night, and offers an opportunity for harnessing the computing power of idle smartphones during these times.

**Stability of the wireless network:** To fully utilize the idle periods to execute jobs, a stable network connection is necessary. Since we only schedule jobs when a phone is on charge (typically at night), it is safe to assume that the channel qualities do not fluctuate much. The location of a device may, however, affect the bandwidth (due to fading); to account for temporally varying fading effects, a periodic (short) bandwidth measurement test is required prior to scheduling jobs on the phones. To examine the stability of these measurements over WiFi links, we perform experiments at three different locations (within a 2 mile radius), when the phones are put on charge. Figure 6.5 depicts the results of such a bandwidth test for WiFi links where we run a *iperf* session from the phones to the server for 600 seconds.

We see that the variation in bandwidth for WiFi links is very low; this means that we can use infrequent (periodic) bandwidth measurements. Since we expect that communications between the smartphones and the supporting server will typically be via WiFi at users' homes, we conclude that bandwidth stability is not likely to be an issue. Cellular links can also be utilized as appropriate, but will require more frequent bandwidth measurements [127].

### 6.3.2 Benefits

**Savings in infrastructure costs:** Since the idle compute resources on already deployed smartphones are used, the cost borne by corporations to bootstrap the platform will be minimal in comparison to that in setting up a similar service on a server-based infrastructure. Companies have either to invest in buying hardware (e.g., servers, switches) or in outsourcing their tasks to third party cloud services. In addition, establishing computing infrastructure requires careful planning with regards to factors such as space, federal and state regulations, and the provisioning of power and cooling support. In contrast, the use of a smartphone infrastructure obviates such considerations. To leverage existing smartphones as the elements of a utility computing service, an enterprise will need no more than a central, lightweight server to identify idle resources and allocate them to computational tasks.

**Savings in energy costs:** A primary concern of cloud service providers is the power consumption in their data centers. A typical data center server can consume 26.8 Watts (Intel Core 2 Duo) to 248 Watts (Intel Nehalem) [128] of power, depending on the configuration. More importantly, this does not account for the power required for cooling. In a data center setting, to calculate the total power consumption, we use an Average Power Usage Effectiveness (PUE) ratio [129] of 2.5; for every Watt consumed by a server, $2.5$ watts are in addition consumed for cooling and power distribution overheads. Extrapolating this, we can project the energy cost of a server with a Intel Core 2 Duo to be:

$$\frac{67}{1000}\text{KWH} \times 24 \text{ hrs} \times 365 \text{ days} \times \$0.127 = \$74.5/\text{year}$$

(using the average commercial price of 12.7c/KWH in the US in April 2011). Note that a more powerful server (like the Intel Nehalem) may cost up to $\$689$ /year.

In comparison to a datacenter server, the power consumption of a smartphone is as low as 1.2 Watts at peak load. The cost of operating a smartphone (with a similar model) can be estimated to be:

$$\frac{1.2}{1000}\text{KWH} \times 24 \text{ hrs} \times 365 \text{ days} \times \$0.127 = \$1.33/year$$

Note that the PUE ratio does not apply in case of Smartphones since they do not require any cooling. The above analysis suggests that energy costs of operating the smartphone computing infrastructure is significantly lower (by an order of magnitude) than using typical datacenter servers.

## 6.4 Design and Architecture

In this section, we describe the design of CWC. We first describe the parallel task (job) execution model for CWC [1], and then seek answers to the following. **(a)** How can we predict task execution

[1]We use the terms task and job interchangeably.

times?, **(b)** How can we implement automated task execution on smartphones without requiring direct user interaction?, and **(c)** How can we preserve user experience while the tasks are being executed on the phones?

**Task model:** In CWC, a task is a program that performs a computation on an input file, such as counting the number of occurrences of a word in a text file. Similar to the model in MapReduce, a central server partitions a large input file into smaller pieces, transmits the input partitions (together with the executable that processes the input) to the smartphones in CWC. Upon receiving the executable and the corresponding input, the phones execute the task in parallel and return their results to the central server when they finish executing the task. The central server performs a logical aggregation of the returned results, depending on the task. For the word count example, the server can simply sum the number of occurrences reported by each phone (obtained by processing their respective input partitions) to compute the number of occurrences in the original input file. We call such tasks *breakable tasks* to reflect that in this class, a task does not exhibit dependencies across partitions of its input and hence can be broken into an arbitrary number of concurrent pieces.

While the above model is suitable for parallel tasks in general, some tasks *cannot* be broken into smaller pieces on which computations can be performed followed by merging the results, to produce a logical outcome. We call such tasks *atomic tasks*; such a task (and its input) can only be executed on a single phone due to the dependencies in its input. An example of an atomic task is photo filtering (e.g., blurring a photo). A blur is typically obtained by computing a new pixel value based on the neighboring pixels. Since the blurred pixel value depends on its neighboring pixels, a blur on a photo cannot be obtained by breaking the photo into smaller pieces, blurring the pixels in each individual piece and merging the results. Although an atomic task cannot be parallelized, there are still concurrency benefits when many such tasks are executed in batches. For example, if one needs to filter 1000 photos, each individual photo can be transferred to a phone and thus, multiple photos can be filtered in parallel. CWC accounts for both breakable tasks and batch atomic tasks in its scheduler (details in Section 6.5). Next, we describe how CWC predicts task execution times.

### 6.4.1 Predicting Task Execution Times

When a task is scheduled on a phone, there are two important factors that affect the completion time of that task. First, it takes time to copy the executable (i.e., binary) and the input file partition to a phone. This depends on the achievable data rate on the link between the phone and the central server that copies the data. Second, the same task takes different times to complete on different phones (depending on the computational capabilities of the phone). While "computational capabilities" is broad and can include characteristics such as the speed of reading a file from the disk (e.g., the SD card on a phone) or the size and speed of the cache, we only focus on the CPU clock speed of a phone; a phone with a fast CPU (in GHz) should execute a given task in less time as compared to a phone with a slow CPU.

Next, we introduce some basic notation that we use in the subsequent discussions. $b_i$ is the time that it takes to copy 1 KB of data from the central server to phone $i$. $c_{ij}$ is the time it takes to execute task $j$ on 1 KB of input data using phone $i$. $E_j$ is the size (in KB) of the executable for task $j$ and $L_j$ is the size (in KB) of input data that task $j$ needs to process. Given this notation, the completion time of task $j$, when it is scheduled to run on phone $i$, is equal to $E_j * b_i + L_j * (b_i + c_{ij})$. The first term accounts for the time that it takes to copy the executable to the phone and the second term accounts for the copying of the input data and executing the task on it. If phone $i$ is assigned a piece of job $j$'s input file, we denote this by $l_{ij}$ and one can simply replace $L_j$ with $l_{ij}$ in the above formula to account for executing input partitions.

The estimation of the $b_i$ values are via direct measurements in CWC (bandwidth tests described earlier). The estimation of $c_{ij}$ for each phone task pair has to be low-cost since many such combinations may exist. To estimate $c_{ij}$ values, we resort to a scaling technique where we first execute each task $j$ on 1 KB of its input using the slowest phone with a $S$ MHz CPU speed. If the slowest phone takes $T_s$ milliseconds to locally execute task $j$ on a 1 KB input (excluding the associated executable and data copying costs), a phone with '$A$' MHz CPU speed is expected to complete the same task in $T_s * \frac{S}{A}$ milliseconds. This technique avoids the cost of profiling each phone-task pair and as we

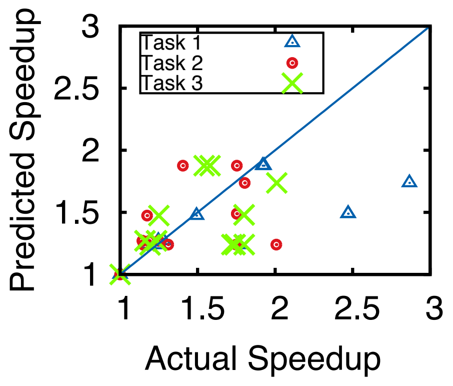


Figure 6.6: Predicted speedup vs. the measured speedup.

show in Figure 6.6, is a fairly accurate representation of actual task completion times. In plotting Figure 6.6, we first run a task on the slowest phone in our testbed (HTC G2 with 806 MHz CPU). We then run the same task on all the other phones. Comparing the actual runtimes of each phone $i$ (denoted $t_i$) to the run time of the slowest phone (denoted $t_s$), we have the measured speedup, $\frac{t_s}{t_i}$. We then compute the expected speedup based on CPU clock speeds. If a phone has a X MHz CPU, then the expected speedup with respect to the HTC G2 is equal to $\frac{X}{806}$. We do this comparison for three different tasks (described later in detail in Section 6.6). Figure 6.6 shows that the CPU scaling model captures the actual speedup for most of the points (the points are clustered around the $y = x$ line), with a few exceptions where the actual speedup is higher than what is predicted by the model (the rightmost points on the x-axis).

The above model is used by CWC's task scheduler (described in Section 6.5), which runs on the central server and periodically assigns partitions of tasks to a set of phones based on the predicted run time. The phones return their results along with the time it actually took to locally execute their last assigned task. The scheduler then updates its prediction for each phone (and task) based on the reported execution times and uses it for predicting the run time in the following scheduling period. With this, CWC accounts for the few cases that the initial prediction fails to capture.

### 6.4.2 Automating Task Execution

One of the key requirements of CWC is that a task be executed without requiring continuous user input. The typical means of "running a task" on smartphones today is running an application (i.e., "app"). When a user wants to execute a new task on her phone, she needs to download and install the app. This process typically requires direct human input for various reasons (e.g., Android users are presented a list of app permissions and have to manually validate the installation). Such a mechanism is clearly not apt for CWC, since the tasks are to be dynamically scheduled on smartphones.

To run tasks on the phones, we leverage a cross-platform mechanism that uses the Java Reflection API for the Android OS. With reflection, a Java executable (i.e., a `.class` file) can dynamically load other executables, instantiate their objects and execute their methods, at runtime. This allows CWC to ship different task executables and input files to a particular phone in an automated fashion. In addition, the reflection functionality can be implemented as an Android service [2] thus, bypassing the need for human input. Note that dynamic class loading is not specific to Android; such capabilities are also available with other smartphone OSs (e.g., iOS permits this via shared libraries).

With reflection implemented on the smartphone side, CWC does not require any additional infrastructure at the central server. In fact, developers can continue to use their traditional Java programs and have them scheduled for parallel execution by CWC. Since Android can execute Java code, we just require developers to implement their tasks in Java (no knowledge of Android API required). In Fig. 6.7, we depict the flow chart of a typical CWC task. The `.java` source files are compiled into `.class` files at the central server, which are then packaged as `.jar` files using the Android tool chain (i.e., the *dx* command). The `.jar` file (containing the executable for the Android VM) together with the input data is copied to the phone. The phone extracts the `.jar` file and uses reflection to load and run the task, producing an output of results. Figure 6.8 shows the Java implementation of a typical CWC task. When the `.java` file is compiled and packaged in a `.jar` file, the task is executed on the phone using the code snippet in Figure 6.9. Thus, each

[2] Android services do not display graphical elements to users and can run in the background.

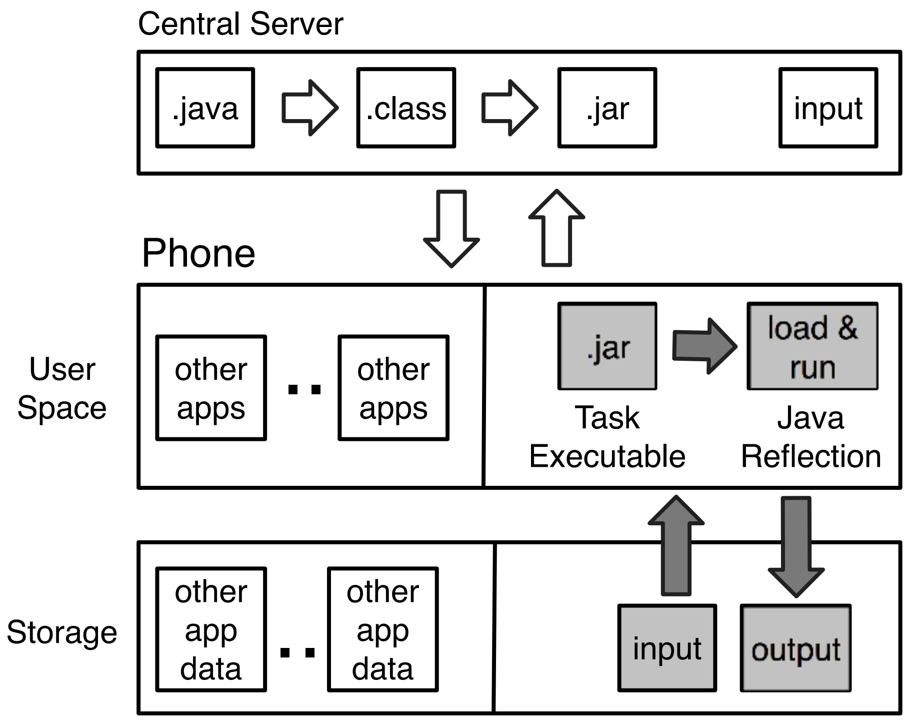


Figure 6.7: Flow chart of a CWC task (shown in shaded components).

```
public class Task {
    public static void main (String[] args) {
        executeTask(args[0]);
    }
    public static void executeTask (String filename) {
    //read and process input
    }
}
```

Figure 6.8: Task.java to be compiled at the central server.

phone concurrently processes the input partition (*input.txt*) assigned to it and this is transparent to the developers who implement their tasks (with the template in Figure 6.8) with a single desktop machine in mind.

### 6.4.3 Preserving User Expectations

While predicting task execution times and automating them are important, we must note that phones are personal devices. Thus, first it is important to ensure that when a user chooses to use her phone,

```
String path = getFilesDir().getAbsolutePath();
String jarFile = path + "/task.jar";
DexClassLoader classLoader = new DexClassLoader(jarFile,
    path, null, getClass().getClassLoader());
Class[] types = {new String[]{}.getClass(),};
Class<?> myClass = classLoader.loadClass("Task");
Method m = myClass.getMethod("main", types);
Object[] passed = {new String[]{path + "/input.txt"}};
Object x = m.invoke(classLoader, passed);
```

Figure 6.9: Reflection functionality on the smartphone.

CWC stops the execution of the last assigned task to that phone so as not to adversely impact the end-user experience (e.g., task execution on the CPU can affect the responsiveness of the user interface). The tasks that are thus stopped, are then migrated to other phones that are still plugged in (as discussed).

Second, running tasks on phones that are are plugged in, should have a minimal impact on the charging times of the phones' batteries. We observe that a heavy utilization of a phone's CPU draws power and therefore, in some cases, prolongs the time taken to fully charge a phone's battery. Specifically, we conduct experiments where we first fully charge many types of phones (i.e., from 0% residual battery to 100%) in two settings; the first setting is without any job running on the phone, and the second setting corresponds to a case wherein a CPU intensive task is continuously run. As an example, we discuss results in the case of HTC Sensation phones, where we repeatedly run a CPU intensive task of counting the number of prime numbers in a large input file continuously during the charging period. We observe that while it takes around 100 minutes to complete full charging in the first setting, the time increases to 135 minutes in the second setting. Note that this increase could be phone-specific; our observation is that the more powerful the phone, the higher the penalty in terms of the increase in the charging period. Further note that, if the tasks are only scheduled after the phone is fully charged, there is no penalty (the phone remains fully charged); this is because the energy from the power outlet is directly applied to CPU computations. However, this would delay task processing and is thus avoided in CWC; moreover, users may not leave their phones plugged in until they are fully charged.

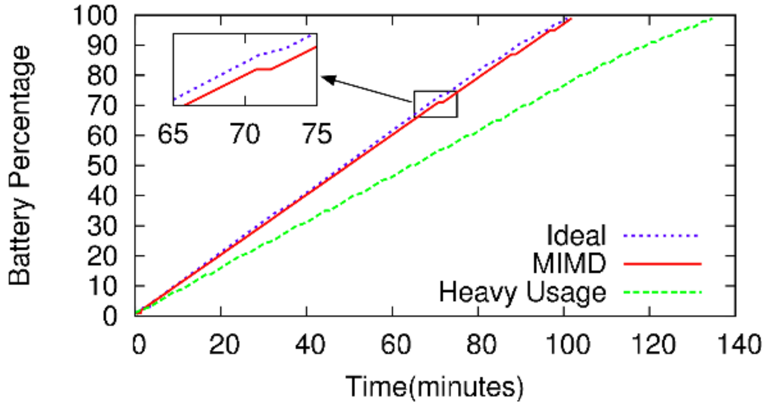


Figure 6.10: Charging times with different discharging schemes.

Our goal is minimize the aforementioned adverse affect on a device's charging profile. If the CPU utilization can be controlled, we could achieve the above goal. Prior approaches advocate dynamically varying the voltage and/or the frequency of the CPU [130]. However, the modification of the voltage and frequency values require root privileges on the phone, and is therefore not applicable in our setting (using root privileges on a device voids its warranty). Thus, our approach is to periodically pause the tasks being executed on the phones, and leave the CPU idle during such paused intervals. Next, we discuss when and for how long, we pause the execution of a task.

To begin with, our experiments demonstrate that the residual battery percentage (reported by the operating system) exhibits a predictable linear change with respect to time (as seen from Figure 6.10) in the case where no jobs run on the phone (referred to as the phone's charging profile). The rate of this linear change is specific to the device and the power source, but the relation remains linear across all the devices. When a task runs on the CPU of the phone, it draws power and thus, the charging profile deviates from this linear profile. We seek to minimize this deviation.

If there was no other task (e.g., background jobs not scheduled by CWC) running on the phone, we could determine the deviation due to CWC. Unfortunately, the process is complicated by the existence of such other tasks (possibly at different times) each of which with unpredictable CPU requirements and therefore power consumption patterns. Further, there could be fluctuations in the input power drawn from the supply (e.g. charging using a USB vs. a wall charger). Given this, our approach is to continuously monitor the rate at which the battery is charged, and either increase or decrease CPU utilization accordingly; the amount by which we increase the CPU utilization is called the scaling factor.

Specifically, we first measure the time it takes for the residual battery charge ($\delta$) to increase by 1% of its preceding value, without any jobs running on the phone. This value of $\delta$ is referred to as the target charging parameter. Then, we run the task for a time period of $\delta/2$ and put the process to sleep for the next $\delta/2$ seconds. We repeat this process until the overall residual battery charge increases by 1%. Let the time taken for this be $\beta(\geq \delta)$; this is referred to as the actual charging parameter. If $\beta = \delta$, there may be energy available to further ramp up the CPU utilization (the power from the outlet might be higher than what is required for charging). In this case, we decrease the sleep time during each $\delta$ period by a factor of $0.75$, thereby inherently increasing CPU utilization; a new $\beta$ is then computed based on the new settings. If $\beta > \delta$, the power drawn by the CPU is affecting the battery charging profile. Thus, we increase the sleep time by a factor of 2. Again, a new $\beta$ value is computed. The process is repeated continously. Note that the above strategy is akin a multiplicative increase/multiplicative decrease (MIMD) of the period for which the CPU is kept idle. Finally, since the phone's charging profile could change with time (as an example due to other tasks), we recompute the value of $\delta$ each time the residual battery charge changes by 5%.

We plot the results with our adaptive MIMD based CPU scheduling in Fig. 6.10 for the HTC Sensation phones. The results from an ideal charging profile (no tasks) as well as a case where the CPU is heavily utilized without our approach are also shown. We see that our approach allows the phone to charge in a time that is almost the same as in the ideal case; there MIMD behavior of our approach is highlighted in the zoomed insert in the figure. Without our approach, the charging time increases by 35 %. Note here that, the use of the adaptive approach results in a reduction in computation (increase in computation time) of about 24.5 % compared to the heavily utilized scenario (due to the sleep cycles).

## 6.5 Task Scheduling

In this section, we detail how tasks are scheduled in CWC. We are given a set $J$ of jobs and a set $P$ of smartphones. As discussed earlier, each job $j \in \mathcal{J}$ and phone $i \in \mathcal{P}$. The time it takes $i$ to process $x$ KB of $j$'s input is given by

$$E_j * b_i + x * (b_i + c_{ij}) \tag{6.1}$$

where, $E_j$ is the size (in KB) of job $j$'s executable, $b_i$ is the time (in milliseconds) that it takes phone $i$ to receive 1 KB of data from the server, and $c_{ij}$ is the time that it takes for phone $i$ to execute the job $j$ on 1 KB of input data. Our objective is to schedule the tasks across the phones such that the time it takes for the last phone to complete, $T$, (the *makespan*) is minimized. In the schedule, each job $j$'s input can be split into pieces and each piece can be assigned to a phone. $l_{ij}$ denotes the size (in KB) of job $j$'s input partition assigned to phone $i$. $l_{ij} = 0$ simply indicates that phone $i$ is not assigned any input partition of job $j$. $u_{ij}$ is an indicator variable that denotes whether or not a partition of job $j$'s input is scheduled to run on phone $i$. The scheduling problem (SCH) is then captured by the following quadratic integer program.

SCH: Minimize $T$

$$\text{s.t.} \qquad \sum_j u_{ij} * (E_j * b_i + l_{ij} * (b_i + c_{ij})) \leq T,\ \forall i \in \mathcal{P}$$

$$\sum_i l_{ij} \quad = \quad L_j,\ \forall j \in \mathcal{J}$$

$$u_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{P}, \forall j \in \mathcal{J}$$

$$\sum_i u_{ij} = 1 \ \ \forall \text{ atomic } j \in \mathcal{J}$$

where we minimize the makespan, $T$. The first constraint requires that all phones finish executing their assigned tasks before $T$. The second constraint ensures that for every job, all of its input is processed. The last constraint ensures that atomic jobs are allocated to a single phone. SCH reflects the general case for the minimum makespan scheduling (MMS) problem, which is known to be NP-hard. MMS is defined as: "*Given a set of jobs and a set of identical machines, assign the jobs to the machines such that the makespan is minimized*" [131]. A more general version of MMS is scheduling using unrelated machines (U-MMS), where each machine has different capabilities and thus, can execute tasks in different times. In both of these problems, only atomic jobs are considered. In other words, the goal is to assign each job to exactly one of the machines such that the makespan is minimized. SCH is a general case of U-MMS. We consider both atomic and breakable tasks and the machine capabilities are different. Since the special case of SCH (U-MMS) is NP-hard, the hardness carries over to SCH as well.

**Our Solution:** We address the SCH problem by solving the complementary bin packing problem (CBP), similar to the approach in [132]. In CBP, the objective is to pack items using at most $\|\mathcal{P}\|$ bins (with capacity $C$) such that the maximum height across bins is minimized. Here, the items correspond to the tasks and the bins correspond to the phones. The correlation between CBP and SCH can be drawn as follows. Let us assume that there is an optimal solution to CBP where the maximum height across the bins is $M$. If one rotates each bin $90°$ to the right, each bin visually appears as a phone in makespan scheduling. Items packed on top of each other in a bin correspond to input partitions assigned to a phone one after the other. Clearly, $M$ corresponds to the maximum completion time across the set of phones in the rotated visualization. Thus, packing all items (tasks) using at most $\|\mathcal{P}\|$ bins (phones) and minimizing the maximum height across bins will minimize the makespan [3].

[3]The reader can refer to [132] for details.

```
$L$ : sorted list in decreasing order of execution time
$C$ : bin capacity
repeat
    find the first item in $L$;
    that can fit in any opened bin
    if such an item exists then
        pack the item in the bin with min. height
        if the item was packed as a whole then
            remove it from $L$
        else
            insert its remaining input in $L$
            re-sort $L$
        end if
    else
        if there are un-opened bins then
            open the best bin for the largest item in $L$
            pack the item in the opened bin
            if the item was packed as a whole then
                remove it from $L$
            else
                insert its remaining input in $L$
                re-sort $L$
            end if
        else
            cannot open any more bins
            cannot finish packing with $C$
        end if
    end if
until all jobs are packed
```

**Algorithm 8**: Greedy Packing Algorithm

The pseudocode of our greedy algorithm to solve CBP is given in Algorithm 8. The idea is to first sort the tasks in decreasing order of local execution time. The first item in the sorted list is one where $R_j * c_{sj}$ is the largest where $s$ is the slowest CPU phone in the system and $R_j$ is the remaining input size (in KB) of item (job) $j$ that is yet to be assigned to some phone. Initially $R_j = L_j$.

In each iteration, we search for the first item in the list that can be packed in any of the previously opened bins (an open bin represents a phone that has previously been assigned some input partition). Note that determining whether an item can be packed in a bin depends on whether the current height of the bin plus the execution cost of that item in the particular bin is less than the bin capacity. If we can find such an item, we pack it in the bin with the minimum height at that time (i.e., the phone with the least total execution time). When packing such an item (line 6), we pack its largest input partition that can fit. If the item can fit without partitioning it, we prefer packing it as a whole.

The idea behind our design is the following. If a task is broken down to $N$ pieces, then the central server would have to aggregate $N$ partial results, which would be an extra overhead at the central server when the phones return their partial results. Thus, if two packings would produce the same minimum bin height in the end, we prefer one with fewer partitions. If packing an item as a whole is not possible (simply because doing so would result in the bin height going over the capacity), we pack that item's largest partition that can fit without violating the bin capacity (again with the purpose of keeping the number of partitions low). If no item can fit in the opened bins (line 13), we check if we can open a new bin. If not, the algorithm cannot find a feasible packing for the given capacity. If we can open a new bin, we open the bin that would accept the largest item in $L$, with the minimum increase in its height (line 15). Clearly, such a bin is the one that minimizes Equation 6.1 for the largest item in $L$. After opening the bin, we again try to pack the item as a whole (line 17). If not, we pack the item's largest partition subject to the bin capacity.

The above algorithm is repeated multiple times for different selections of bin capacities. Here, we adopt an approach similar to binary search. We first determine an upper bound ($UB$) on the bin height. Clearly, the maximum bin height occurs when all items are assigned to the bin that maximizes Equation 6.1 (i.e, the worst bin). For the lower bound ($LB$), we initially pick a loose bound where all items are packed in a single magical bin that has the aggregate processing capability and the aggregate bandwidth of all bins; there are no other bins. This magical bin represents the ideal case where the inputs are partitioned without the executable cost. After determining these initial bounds, we execute Algorithm 8 with $C = \frac{(LB+UB)}{2}$. If the algorithm succeeds packing all items with bin capacity $C$, we update $UB$ to be equal to $C$. If the algorithm cannot find a feasible packing with the initial $C$, we update $LB$ to be equal to $C$. The algorithm is then repeatedly executed with $C = \frac{(LB+UB)}{2}$, until the algorithm succeeds with the minimum $C$. Here, the binary search simply reduces the search space for the minimum bin capacity, with which the algorithm packs all the items.

When CWC determines the schedule as described above, it starts copying the relevant executables and the input partitions to each phone. This is done on a per-partition basis; the next assigned task to the phone is copied only after the phone completes executing its last assigned task. When the phones inform the central server about a task completion, they report the partial results together with the time it takes to locally execute the assigned task. As described in Section 6.4, CWC uses such execution reports to update its prediction on completion times of tasks. If the same task is assigned to the same phone in the future (albeit with a different input partition), CWC uses the updated prediction for scheduling.

**Handling Failures:** In CWC's task execution cycle, some phones may naturally fail while executing a given task. In our setting, the term failure can correspond to a variety of cases. For example, when a phone is plugged off the charger, we treat it as a failed node since continuing to execute a CPU-intensive task on it would drain the battery (a critical concern for CWC). In CWC, such failures are communicated back to the central server whenever possible (i.e., when the phone still has a network connection), and the execution can be resumed from the point where it failed (details of task state migration are in Section 6.6). We call these class of failures where the phone maintains a connection with the server "online failures". Other scenarios may include harder failures, in which the phone loses its connection to the server (e.g., wireless driver suddenly crashes or the network connection is dropped), and thus, cannot report its failed state back to the central server (the description of detecting such failures at the central server is again deferred to Section 6.6). We call this class of failures "offline failures".

Assume that at time instant $A$, we compute a schedule $X$. In $X$, each phone $i$ has a set of tasks $X_i$ that it will execute as time progresses. CWC starts copying the executable and input partitions in $X_i$ to $i$ one task at a time and waits for $i$ to either report a completion or a failure. If no report is received for the last copied task, say $last_i$ (due to an offline failure), CWC marks $i$ as failed and inserts $last_i$ and all the remaining tasks in $X_i$ to a list $F_A$ that contains all failed tasks after $A$. If $i$ reports completion, CWC simply copies the next task in $X_i$ and again waits for reports. If on the other hand, $i$ reports failure for $last_i$, the report contains additional information: **(a)** how much of the input was processed by $i$ by the failure instance, and **(b)** what was the intermediate (partial) result associated with the processing. CWC still inserts $last_i$ (and all that remains in $X_i$) in $F_A$, but now $last_i$ is inserted with only the part of the input not processed by $i$ (and the intermediate results are saved). Now assume that we have a new schedule to be computed at time instant $B$. Some new tasks have entered the system at this point and awaiting scheduling. Now, CWC computes a schedule for all such new tasks and $F_A$ combined. The reason that we avoid immediate re-scheduling of tasks in $F_A$ and wait until $B$ is to account for the possibility that failed phones may re-enter the system after a short period of unavailability (e.g., the user plugs her phone to the charger after a few minutes or the connectivity is restored). Note that this is in contrast with typical MapReduce architectures, where failures may result in long periods of unavailability [133].

## 6.6 Implementation and Evaluation

In this section, we first describe our prototype implementation of CWC. Subsequently we present our evaluations using the prototype deployment.

**Implementation and the Testbed:** Our testbed consists of 18 Android phones with varying network connectivity and CPU speeds. The network interfaces vary from WiFi (both 802.11a and 802.11g are considered) to EDGE, 3G and 4G. The CPU clock speeds vary from 806 MHz to 1.5 GHz. Each phone registers with a central server and reports its CPU clock speed. We measure $b_i$ values with the *iperf* tool.

The phones host the CWC software, which maintains a persistent TCP connection with the server and permits dynamic task execution as instructed by the scheduler. To maintain long-lived flows, we use the $SO\_KEEPALIVE$ option in the connection sockets as well as implement custom application layer keep-alive messages. The latter also serve as a means of detecting offline failures. If a phone fails to respond to a preset number of keep-alive requests from the server, it is marked as failed. In our implementation, the keep-alive message period is 30 seconds and the number of response failures tolerated, is 3.

Several techniques exist to migrate failed tasks and resume their state on a target machine; for example, the authors in [134] modify the Android virtual machine (VM) itself to migrate the state of execution but this requires changes to the original Android VM. To make it more user and developer friendly, we ported JavaGO – a Java program migration library [135] to the Android. JavaGO is based on a "source-code-level" transformation technique, where the JavaGo translator takes user program as input and outputs the migratory Java code. The translated code can run on any Java interpreter and can be compiled by any Just-in-Time (JIT) compiler. JavaGo provides flexibility by allowing programmers to annotate their code using three added language constructs to the Java language, namely *go*, *undock* and *migratory*. The *go* statement specifies the IP address of the machine where the failed application will be resumed. The *undock* construct specifies the area to be migrated in the execution stack while *migratory* construct declares which methods are migratory so that only those methods are modified by the JavaGo compiler. In CWC, we translate the annotated java task files with JavaGo translator to produce the migratory `.jar` task file. In case of a failure, the state of a task is saved and transmitted to the central server (via the *go* construct). Our server records the transmitted state but does not itself resume the computation at that state. At the next scheduling instant, the server sends the recorded state of each failed task to a newly assigned phone, which then resumes the task. Details on migration can be found in [135].

The server is implemented as a multi-threaded Java non-blocking IO (NIO) server. The use of threads with NIO allows the server to concurrently copy data to a phone while reading the completion reports of other phones. We host the central server on a small Amazon EC2 instance to showcase its lightweight implementation and its economical viability. The small instance is the default configuration offered by EC2 and a single virtual core together with the 1.7 GB of memory that it offers represents a machine that is far less capable than state-of-the-art workstations. Currently, Amazon charges 8 cents per hour for a small Linux instance. This (although the exact values may change over time) clearly shows that the lightweight central server in CWC incurs a very small cost for a typical enterprise.

**Prototype Evaluation:** In evaluating CWC, we use a variety of tasks. The first task involves counting the occurrences of prime numbers in an input file. The second task is to count the number of occurrences of a word in the input file and the third task is to blur the pixels in a photo. While we are able to directly use the desktop Java versions of the first two tasks, doing this was not possible for the photo blurring task. The challenge relates to the lack of compatibility between the graphics classes on the desktop Java virtual machine and the Dalvik virtual machine in Android. While the code works on JVM, the reflection class loader on Android complained about the part of the code that reads the pixels from the image file (in particular BufferedImage class does not exist in Android). To eliminate the phone's reading the pixels directly from the image, we do the following modification. We first pre-process the pictures to read the pixels (at the central server) and create text files that contain a pixel value in each of its lines. Each phone was able to process the text files as before. After this, the server re-creates each photo from the blurred pixels returned by each phone.

***Comparison with simple practical schedulers:*** At the server, we also implement simpler scheduling alternatives to CWC. The first alternative splits each breakable job into $|\mathcal{P}|$ pieces without accounting for the different bandwidth and CPU speeds of phones in $\mathcal{P}$. The atomic jobs are assigned to phones in a round-robin manner (phone 1 gets atomic job 1 and so on). In the second alternative, both breakable and atomic jobs are assigned in a round-robin manner.

***Setup:*** We distribute our 18 phones in three houses (the locations are shown in Fig. 6.11). In two of these houses, we have a 802.11g WiFi network and an abundance of interfering residential access points using the 2.4 GHz band. In the third house, we have a 802.11a WiFi AP without interference from neighboring APs. Of the 6 phones we place in each house (phones are plugged to power outlets), we associate 2 phones with the WiFi AP and 4 phones are configured to use varying cellular technologies (from the slowest EDGE to the fastest 4G). Before running the CWC scheduler, we initiate *iperf* sessions from each phone to the EC2 server and log the measured data rate in KBps (the inverse of this value is used as $b_i$). The workload comprises of the following. We have 50 instances of task 1 (counting primes) with varying input data sizes, we have 50 instances

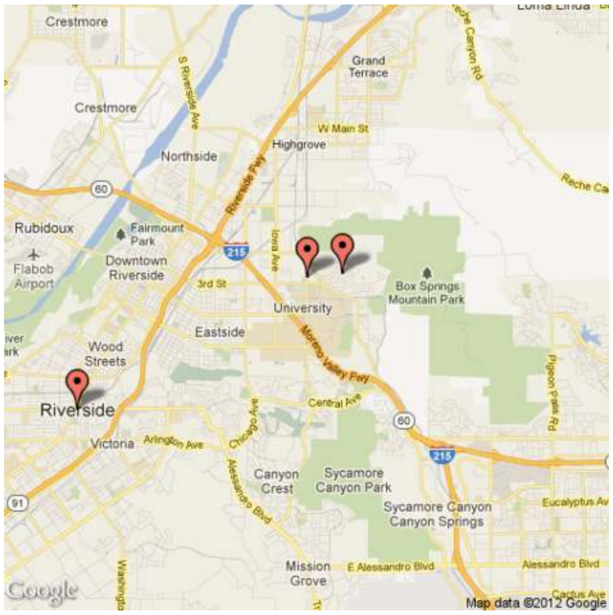
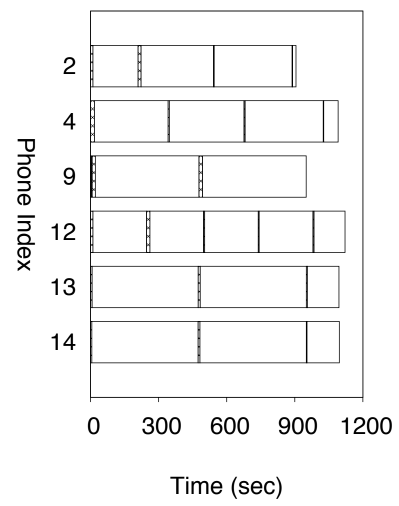


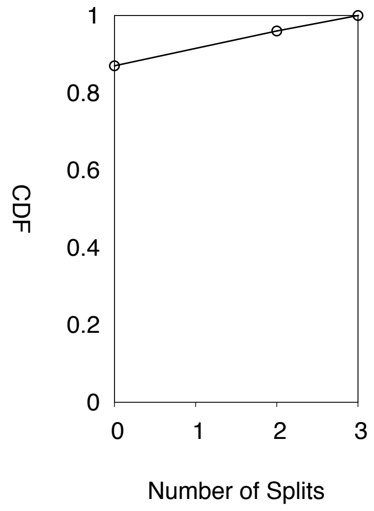
Figure 6.11: Map of the phone locations.

of task 2 (counting word occurrences) with varying input sizes and, 50 variable size photos to be blurred (atomic task 3).

***Results:*** In the first experiment, we run our greedy scheduler followed by the two alternate scheduling strategies described above; we do not consider phone failures. Fig.6.12(a) presents the task execution timeline for a select set of phones. We do not show the plots for every phone for better visualization (the patterns are similar across the phones). The vertical black stripes in a phone's timeline correspond to the time intervals where the phone is receiving the task executable and the corresponding input partition from the server. The white regions correspond to the time intervals where the phone executes the task locally. From Fig. 6.12(a), we observe that while some phones (2 and 9) finish their tasks earlier than others, the load is well balanced for most of the phones (4, 12, 13, 14 complete at similar time instants). Phones 2 and 9 finished early because of a mismatch between the expected speedup and the measured speedup (recall Fig. 6.6 in Section 6.5). In particular, these phones are faster than what is indicated by their CPU clock speeds and thus, finish earlier than the scheduler's prediction. We see that the difference in the completion times between the earliest phone (2 finishes at around 900 seconds) and the last phone (12 finishes at



(a) Timeline of CWC's task execution



(b) CDF of the number of input partitions

Figure 6.12: Our greedy scheduler balances the load and produces very few input partitions.

around 1100 seconds) is ≈20% of the makespan. In addition, our scheduler's predicted makespan of 1120 seconds was only 20 seconds more than the actual makespan of 1100 seconds. Fig.6.12(b) shows the CDF of the number of input partitions for each of the 150 tasks considered. An input partition of 0 indicates that the task was atomically assigned to a single phone. While 33% of the tasks by definition cannot be partitioned (the photo tasks are atomic), we observe that our scheduler preserves atomicity for most of the tasks (close to 90%) and thus, significantly reduces the aggregation cost at the server. In contrast, we observe that the equal split strategy achieved a makespan of 1720 seconds, and produced a large number of input partitions since each breakable task is split into $|\mathcal{P}|$ pieces. While the round-robin strategy avoided excessive input partitions, it achieved a makespan of 1805 seconds. *In summary, our greedy scheduler is around* $1.6x$ *faster than the alternative schedulers and it achieves this while with almost negligible aggregation costs.*

In Fig. 6.13, we plot the timeline for a different run of the above experiment. Here, we introduce failures by unplugging three phones (phones 1, 6 and 17) at random instances during task execution (the plot again shows a subset of phones). As discussed in Section 6.5, in the next round of scheduling, our scheduler re-schedules tasks from previously failed phones across the remaining

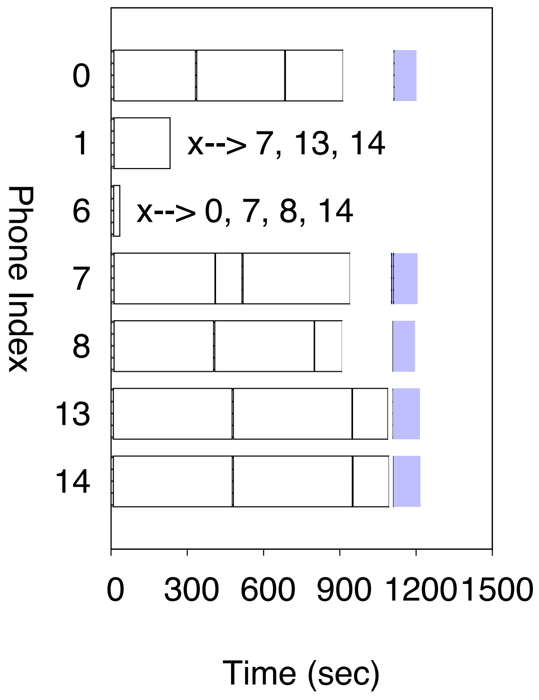


Figure 6.13: Support for failure recovery in CWC.

set of phones. The $x$ marks on Fig. 6.13 indicate the assignment of the failed tasks of a phone. The shaded task executions depict the execution of re-scheduled tasks. We observe that phone 1's tasks were partitioned across phones 7, 13 and 14. On the other hand, phone 6's failed tasks were re-scheduled across phones 0, 7, 8 and 14. Since phone 6 failed at the very beginning of its schedule, it had more tasks to be re-scheduled. Our scheduler mainly chose faster phones (phones 0, 7 and 8 completed ahead of time in the original schedule) to re-schedule these failed tasks. Overall, re-scheduling failed tasks required an additional 113 seconds after the original makespan.

**Benchmarking the Scheduler:** Next, we try to get a lower bound on the makespan to benchmark the performance of our algorithm. This requires optimally solving the quadratic integer program formulated in Section 6.5, which is NP-hard owing to the integral nature. While the integral part can be relaxed by allowing the variables $u_{ij}$ and $l_{ij}$ to take fractional values and hence producing a loose lower bound, we still cannot use standard LP solvers to compute the bound owing to the quadratic nature. To address this, we can re-formulate the problem by transforming the first

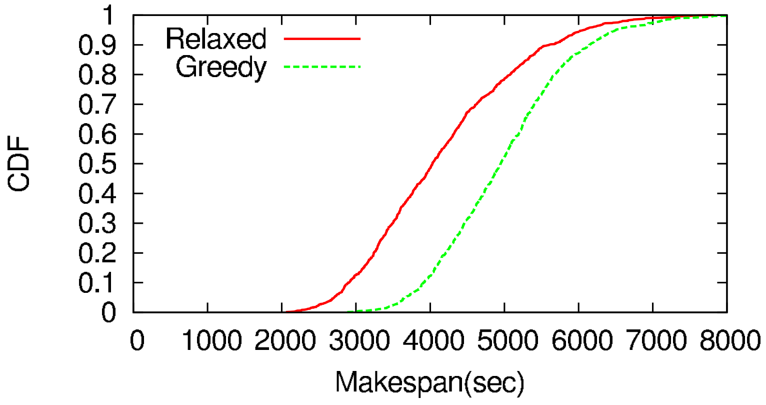


Figure 6.14: Comparison of the makespans of the greedy scheduler and the solution to the relaxed problem.

constraint to $\sum_j u_{ij} * (E_j * b_i) + l_{ij} * (b_i + c_{ij}) \leq T$, where now $u_{ij}$ applies only to the first term. However, to prevent the solution from allocating jobs to a phone ($l_{ij} > 0$) without accounting for the shipping cost of its executable ($u_{ij} = 0$), we add another constraint $(1 - u_{ij})l_{ij} = 0,\ \forall i, j$. The latter can now easily be relaxed as $l_{ij} \leq L_j * u_{ij}$, thereby resulting in an LP relaxation, which can be solved to obtain a loose lower bound (smaller makespan than optimal due to relaxation) on the solution. Thus, we have $T_{relaxed} \leq T_{optimal} \leq T_{cwc}$, where $T$ is the makespan produced by each of these solutions.

To understand how close we are to the optimal, we input various combinations of tasks and bandwidth profiles and solve the problem with **(a)** our greedy scheduler and **(b)** using the relaxed formulation above. We simulate the solutions by generating random $b_i$ values between 1 and 70 milliseconds (the minimum and maximum values measured in our experiments). We consider the same set of 150 tasks and we get the $c_{ij}$ values from the phones in our testbed. We generate 1000 random configurations and for each configuration, we first obtain the makespan for the relaxed formulation and subsequently we obtain the makespan produced by our greedy scheduler. Fig. 6.14 shows the CDF (over random configurations) of these makespans. It is seen that the median makespan of our greedy scheduler is approximately $18\%$ worse than the relaxed formulation's solution.

## 6.7 Conclusions

In this chapter, we envision building a distributed computing infrastructure using smartphones for the enterprise. Our vision is based on several compelling observations including (a) enterprises provide their employees with smartphones in many cases (b) the phones are typically unused when being charged and, (c) such an infrastucture could potentially yield significant cost benefits to the enterprise. We articulate the technical challenges in building such an infrastructure. We address many of them to design CWC, a framework that supports such an infrastructure. We have a prototype implementation of CWC on a testbed of 18 Android phones. Using this implementation, we demonstrate both the viability and efficacy of various components within CWC.

# Bibliography

[1] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot. Measurement-Based Self Organization of Interfering 802.11 Wireless Access Networks. In *INFOCOM*, 2007.

[2] A. Mishra, S. Banerjee, and W. Arbaugh. Weighted coloring based channel assignment for WLANs. In *ACM SIGMOBILE Mobile Computing and Communications Review*, volume 9, July 2005.

[3] I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, and V. Mhatre. MDG: Measurement-Driven Guidelines for 802.11 WLAN Design. In *ACM MobiCom*, 2007.

[4] 802.11n without Channel Bonding is Just Stupid. http://www.theruckusroom.net/2008/03/80211n-without.html.

[5] M. Y. Arslan, K. Pelechrinis, I. Broustis, S. V. Krishnamurthy, S. Addepalli, and K. Papagiannaki. Auto-configuration of 802.11n WLANs. In *ACM CoNEXT*, 2010.

[6] K. Sundaresan and S. Rangarajan. Efficient Resource Management in OFDMA Femto Cells. In *ACM MobiHoc*, 2009.

[7] M. Y. Arslan, J. Yoon, K. Sundaresan, S. V. Krishnamurthy, and S. Banerjee. Experimental Characterization of Interference in OFDMA Femtocell Networks. In *IEEE Infocom Mini-Conference*, 2012.

[8] M. Y. Arslan, J. Yoon, K. Sundaresan, S. V. Krishnamurthy, and S. Banerjee. FERMI: A Femtocell Resource Management System for Interference Mitigation in OFDMA Networks. In *ACM MobiCom*, 2011.

[9] V. Navda, A. P. Subramanian, K. Dhansekaran, A. Timm-Giel, and S. Das. Mobisteer: Using Steerable Beam Directional Antenna for Vehicular Network Access. In *ACM MobiSys*, Jun 2007.

[10] Gartner, Inc. April Press Release. http://www.gartner.com/it/page.jsp?id=1622614. 2011.

[11] Gartner, Inc. March Press Release. http://www.gartner.com/it/page.jsp?id=1570714. 2011.

[12] Enterprise Smartphone Usage Trends. `http://bit.ly/loIqE1`.

[13] IEEE 802.11n Standard, 2009.

[14] V. Shrivastava, S. Rayanchu, J. Yoon, and S. Banerjee. 802.11n Under the Microscope. In *ACM IMC*, 2008.

[15] V. Visoottiviseth, T. Piroonsith, and S. Siwamogsatham. An Empirical Study on Achievable Throughputs of IEEE 802.11n Devices. In *ACM WinMee*, 2009.

[16] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance Anomaly of 802.11b. In *IEEE INFOCOM*, 2003.

[17] Wireless Open-Access Research Platform. http://warp.rice.edu/.

[18] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaugh. Partially overlapped channels not considered harmful. In *ACM SIGMETRICS*, 2006.

[19] L. Hanzo, M. Munster, B.J. Choi, and T. Keller. *OFDM and MC-CDMA for Broadband Multi-User Communications, WLANs and Broadcasting*. IEEE WILEY. ISBN 0-470-85879-6.

[20] H. Rahul, F. Edelat, D. Katabi, and C. Sodini. Frequency-aware Rate Adaptation and MAC Protocols. In *ACM MobiCom*, 2009.

[21] C. Tang and V.J. Stolpman. A Gradient-Based Method for OFDM Sub-Carrier Power Allocation. In *IEEE VTC*, 2005.

[22] Atheros AR5001J. http://www.atheros.com/news/AR5001J.html.

[23] IEEE 802.11 Standard, Chapter 17, 2007.

[24] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A Case for Adapting Channel Width in Wireless Networks. In *ACM SIGCOMM*, 2008.

[25] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan. Load-Aware Spectrum Distribution in Wireless LANs. In *IEEE ICNP*, 2008.

[26] A. Mishra, D. Agrawal, V. Shrivastava, S. Banerjee, and S. Ganguly. Distributed channel management in uncoordinated wireless environments. In *ACM MobiCom*, 2006.

[27] A. Mishra, S. Banerjee, and W. Arbaugh. Weighted Coloring based Channel Assignment for WLANs. In *MC2R, Volume 9 , Issue 3*, 2005.

[28] A. Kumar and V. Kumar. Optimal association of stations and APs in an IEEE 802.11 WLAN. In *NCC*, 2005.

[29] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh. A client-driven approach for channel management in wireless LANs. In *INFOCOM*, 2006.

[30] T.S. Rappaport. *Wireless Communications*. Prentice-Hall. ISBN 0-13-042232-0.

[31] S.M. Alamouti. A simple transmit diversity technique for wireless communications. In *IEEE Journal on Selected Areas in Communications*, October 1998.

[32] Wi-Fi Wireless LAN Frequency, Bands and Channels. http://www.networkdictionary.com/Wireless/Wi-Fi-Wireless-LAN-Frequency.php.

[33] JPcap Library. http://netresearch.ics.uci.edu/kfujii/jpcap/doc/.

[34] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons. ISBN 0-471-50336-3.

[35] UCR Wireless Testbed. http://networks.cs.ucr.edu/testbed.

[36] S. Z. Asif. Aligning business and technology strategies-an evolution of a third generation wireless technology. In *Engineering Management Conference*, 2002.

[37] S. Bali, S. Machiraju, H. Zang, and V. Frost. On the performance implications of proportional fairness (PF) in 3G wireless networks. In *PAM*, 2007.

[38] D.C. Tsilimantos, D.A. Zarbouti, G.V. Tsoulos, G.E. Athanasiadou, and D.I. Kaklamani. Fairness and Throughput Trade-Off Analysis for UMTS WCDMA Network Planning. In *WPC*, 2009.

[39] D. Panigrahi and F. Khaleghi. Enabling trade-offs between system throughput and fairness in wireless data scheduling techniques. In *WWC*, 2003.

[40] K. Sundaresan and K. Papagiannaki. The Need for Cross-Layer Information in Access Point Selection Algorithms. In *ACM IMC*, 2006.

[41] T. R. Jensen and B. Toft. *Graph Coloring Problems*. John Wiley & Sons. ISBN 0-471-02865-7.

[42] IEEE. Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation. In *IEEE Standard 802.1f*, July 2003.

[43] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-Layer Wireless Bit Rate Adaptation. In *ACM SIGCOMM*, 2009.

[44] CRAWDAD. http://uk.crawdad.org/meta.php?name=ilesansfil/wifidog.

[45] Atheros Linux Drivers. http://linuxwireless.org/en/users/Drivers/ath9k.

[46] R. Morris, E. Kohler, J. Janotti, and M. F. Kaashoek. The Click Modular Router. In *ACM Symposium on Operating Systems Principles*, 1999.

[47] GNU wget. http://www.gnu.org/s/wget/.

[48] R. Van Nee and R. Prasad. OFDM for Wireless Multimedia Communications. *Artech House*, 2000.

[49] S. Yeh, S. Talwar, S. Lee, and H. Kim. WiMAX femtocells: a perspective on network architecture, capacity, and coverage. *IEEE Communication Magazine*, 46:58–65, 2008.

[50] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015. February 2011.

[51] Femto Forum. Global Consumer Research on Femtocells - Key Findings of a Six-Nation Study - Parks Associates. January 2011.

[52] V. Chandrasekhar and J. G. Andrews. Uplink capacity and interference avoidance for two-tier femtocell networks. *IEEE Transactions on Wireless Communications*, 8(7), July 2009.

[53] J. Yun and S. Kang. CTRL: A Self-Organizing Femtocell Management Architecture for Co-Channel Deployment. In *ACM MobiCom*, Sept 2010.

[54] T. Quek, Z. Lei, and S. Sun. Adaptive Interference Coordination in Multi-cell OFDMA Systems. In *IEEE PIMRC*, 2009.

[55] T. Lee, J. Yoon nad S. Lee, and J. Shin. Resource allocation analysis in OFDMA femtocells using Fractional Frequency Reuse. In *Proc. of IEEE PIMRC*, 2010.

[56] R. Chang, Z. Tao, J. Zhang, and C.-C. J. Kuo. A Graph Approach to Dynamic Fractional Frequency Reuse (FFR) in Multi-cell OFDMA Networks. In *Proc. of IEEE ICC*, 2009.

[57] Y. Yuan, P. Bahl, R. Chandra, P. Chou, I. Ferrell, T. Moscibroda, S. Narlanka, and Y. Wu. KNOWS: Kognitiv Networking Over White Spaces. In *Proc. of IEEE DySPAN*, April 2007.

[58] L. Yang, W. Hou, Z. Zhang, B. Zhao, and H. Zheng. Jello: Dynamic Spectrum Sharing in Digital Homes. In *IEEE INFOCOM*, 2010.

[59] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A Case for Adapting Channel Width in Wireless Networks. In *Proc. of ACM SIGCOMM*, August 2008.

[60] K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, J. Zhang, and Y. Zhang. Fine-grained Channel Access in Wireless LAN. In *ACM SIGCOMM*, August 2010.

[61] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh. Client-driven Channel Management for Wireless LANs. In *IEEE Infocom*, 2006.

[62] IEEE 802.16e 2005 Part 16. Air Interface for Fixed and Mobile Broadband Wireless Access Systems . *IEEE 802.16e standard*.

[63] PicoChip. http://www.picochip.com.

[64] Accton. http://www.accton.com.

[65] TeraSync. http://www.terasync.net.

[66] IEEE 802.16m 08/003r3. IEEE 802.16m System Description Document[Draft].

[67] D. Lopez-Perez, G. Roche, A. Valcarce, A. Juttner, and J. Zhang. Interference Avoidance and Dynamic Frequency Planning for WiMAX Femtocells Networks. In *Proc. of IEEE ICCS*, 2008.

[68] 3GPP. Technical specification group radio access networks; 3G home NodeB study item technical report (release 8). *TR 25.820 V1.0.0 (2007-11)*, Nov 2007.

[69] S. Kittipiyakul and T. Javidi. Subcarrier Allocation in OFDMA Systems: Beyond water-filling. In *Proc. of Signals, Systems, and Computers*, 2004.

[70] Y. Sun, R. P. Jover, and X. Wang. Uplink Interference Mitigation for OFDMA Femtocell Networks. *IEEE Transactions on Wireless Communications*, 11(2), Feb 2012.

[71] H. Jo, C. Mun, J. Moon, and J. Yook. Interference Mitigation Using Uplink Power Control for Two-Tier Femtocell Networks. *IEEE Transactions on Wireless Communications*, 8(10), Oct 2009.

[72] V. Chandrasekhar and J. G. Andrews. Uplink Capacity and Interference Avoidance for Two-Tier Femtocell Networks. *IEEE Transactions on Wireless Communications*, 8(7), Jul 2009.

[73] Y. Kim, S. Lee, and D. Hong. Performance Analysis of Two-Tier Femtocell Networks with Outage Constraints. *IEEE Transactions on Wireless Communications*, 9(9), Sep 2010.

[74] WMF-T33-118-R016v01. Femtocells Core Specification.

[75] J. R. S. Blair and B. W. Peyton. An Introduction to Chordal Graphs and Clique Trees. http://www.ornl.gov/info/reports/1992/3445603686740.pdf.

[76] A. Berry, J. R. S. Blair, P. Heggernes, and B. W. Peyton. Maximum Cardinality Search for Computing Minimal Triangulations of Graphs. In *Journal Algorithmica*, volume 39, May 2004.

[77] A. Berry, P. Heggernes, and Y. Villanger. A Vertex Incremental Approch for Dynamically Maintaining Chordal Graphs. In *Algorithms and Computation, 14th Int. Symp. (ISAAC)*, December 2003.

[78] R. Jain, A. Durresi, and G. Babic. Throughput Fairness Index: An Explanation. In *ATM Forum Document Number: ATM Forum / 990045*, February 1999.

[79] S. Yeh and S. Talwar. Multi-tier Simulation Methodology IEEE C802.16ppc-10/0039r1. 2010.

[80] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015. Feb 2011.

[81] M. Blanco et. al. On the effectiveness of switched beam antennas in indoor environments. In *Passive and Active Measurements (PAM)*, Apr 2008.

[82] M. Andrews and L. Zhang. Scheduling algorithms for multi-carrier wireless data systems. In *ACM MOBICOM*, Sept 2007.

[83] S. Lee, I. Pefkianakis, A. Meyerson, S. Xu, and S. Lu. Proportional Fair Frequency-Domain Packet Scheduling for 3GPP LTE Uplink. In *IEEE INFOCOM Mini-conference*, Apr 2009.

[84] D.J. love, R.W. Heath, and T. Strohmer. Grassmannian Beamforming for Multiple-Input Multiple-Output Wireless Systems. *IEEE Trans. on Information Theory*, 49(10):2735–2747, Oct 2003.

[85] G. Song and Y. Li. Cross-layer optimization for OFDM wireless networks - Part I: Theoretical Framework. *IEEE Transactions on Wireless Communications*, 4(2), Mar 2005.

[86] A. Pokhariyal, G. Monghal, K. I. Pedersen, P. E. Mogensen, I. Z. Kovacs, C. Rosa, and T. E. Kolding. Frequency Domain Packet Scheduling Under Fractional Load for the UTRAN LTE Downlink. In *IEEE VTC*, 2007.

[87] S. Deb, S. Jaiswal, and K. Nagaraj. Real-Time Video Multicast in WiMAX Networks. In *IEEE INFOCOM*, 2008.

[88] B. Bai, W. Chen, Z. Cao, and K. B. Letaief. Uplink Cross-Layer Scheduling with Differential QoS Requirements in OFDMA Systems. *EURASIP Journal on Wireless Communications and Networking*, 2010(168357), 2010.

[89] C. Nie, M. Venkatachalam, and X. Yang. Adaptive polling service for next-generation ieee 802.16 wimax networks. In *IEEE GLOBECOM*, 2007.

[90] P. Svedman, S. K. Wilson, Jr. L. J. Cimini, and B. Ottersten. Opportunistic Beamforming and Scheduling for OFDMA Systems. *IEEE Trans. on Communications*, 55(5), May 2007.

[91] N. Wei, A. Pokhariyal, T. B. Sorensen, T. E. Kolding, and P. E. Mogensen. Performance of MIMO with Frequency Domain Packet Scheduling in UTRAN LTE Downlink. In *IEEE VTC*, 2007.

[92] L. Fleischer, M. Goemans, V. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *ACM SODA*, pages 611–620, 2006.

[93] M. Fisher, G. Nemhauser, and G. Wolsey. An analysis of approximations for maximizing submodular set functions- II. In *Mathematical Programming Study*, 1978.

[94] M. Bansal and V. Venkaiah. Improved Fully Polynomial time Approximation Scheme for the 0-1 Multiple-choice Knapsack Problem. In *International Institute of Information Technology Tech Report*, 2004. `http://people.csail.mit.edu/mukul/01MCKP.pdf`.

[95] G. Calinescu, C. Chekuri, M. Pl, and J. Vondrk. Maximizing a Submodular Set Function subject to a Matroid Constraint (Extended Abstract). In *Proc. of 12th Conf. on Integer Programming and Combinatorial Optimization*, 2007.

[96] P.R. Goundan and A.S. Schulz. Revisiting the Greedy Approach to Submodular Set Function Maximization. In *Optimization Online Pre-print*, 2007. `http://www.optimization-online.org/DB_FILE/2007/08/1740.pdf`.

[97] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-Layer Wireless Bit Rate Adaptation. In *ACM SIGCOMM*, 2009.

[98] Iphone in Business. `http://bit.ly/LE4dAp`.

[99] Novartis: Apps for good health. `http://bit.ly/KEdJbh`.

[100] Lowe's : Building better customer service. `http://bit.ly/MiT9bk`.

[101] IBM gets US$2mn data center contract from Novartis. `bit.ly/L51P8i`.

[102] NVIDIA says Tegra 3 is a "PC-class CPU". `http://engt.co/srvibU`.

[103] S. Harizopoulos and S Papadimitriou. A Case for Micro-Cellstores: Energy-Efecient Data Management on Recycled Smartphones. In *DaMoN*, 2011.

[104] Variable SMP - A Multi-Core CPU Architecture for Low Power and High Performance. `bit.ly/n65KzQ`.

[105] Smart Phone Chips Calling for Data Centers. `bit.ly/LdM9fS`.

[106] WiFi Bandwidth Use in the U.S. Home Forecast to More Than Double in the Next Four Years. `http://yhoo.it/L9Po9A`.

[107] PhoneLab. `http://bit.ly/NYwRhI`.

[108] Almudeua Díaz Zayas and Pedro Merino Gómez. A testbed for energy profile characterization of IP services in smartphones over live networks. *Mob. Netw. Appl.*

[109] E. Cuervo, P. Gilbert, Bi Wu, and L.P. Cox. CrowdLab: An architecture for volunteer mobile testbeds. In *COMSNETS*, 2011.

[110] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson. Seattle: A Platform for Educational Cloud Computing. In *SIGCSE*, 2009.

[111] SETI@home. `http://setiathome.berkeley.edu`.

[112] P. R. Elespuru, S. Shakya, and S. Mishra. MapReduce System over Heterogeneous Mobile Devices. In *SEUS*, 2009.

[113] Tathagata Das, Prashanth Mohan, Venkata N. Padmanabhan, Ramachandran Ramjee, and Asankhaya Sharma. PRISM: platform for remote sensing using smartphones. In *ACM MobiSys*, 2010.

[114] D. Estrin. Participatory Sensing: Applications and Architecture. In *ACM MobiSys*, 2010.

[115] N. Eagle. txteagle: Mobile Crowdsourcing. In *Internationalization, Design and Global Development*, 2009.

[116] Earl Oliver. The challenges in large-scale smartphone user studies. In *ACM HotPlanet*, 2010.

[117] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. A first look at traffic on smartphones. In *IMC*, 2010.

[118] Alex Shye, Benjamin Scholbrock, Gokhan Memik, and Peter A. Dinda. Characterizing and modeling user activity on smartphones: Summary. In *ACM SIGMETRICS*, 2010.

[119] Earl Oliver. Diversity in smartphone energy consumption. In *ACM workshop on Wireless of the students, by the students, for the students*, 2010.

[120] Mohammad Hajjat, Xin Sun, Yu-Wei, Eric Sung, David Maltz, and Sanjay Rao. Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud. In *ACM SIGCOMM*, 2010.

[121] Timothy Wood, Emmanuel Cecchet, K.K. Ramakrishnan, Prashant Shenoy, Jacobus van der Merwe, and Arun Venkataramani. Disaster Recovery as a Cloud Service: Economic Benefits & Deployment Challenges. In *USENIX HotCloud*, 2010.

[122] Azbayar Demberel, Jeff Chase, and Shivnath Babu. Reflective control for an elastic cloud application: an automated experiment workbench. In *USENIX HotCloud*, 2009.

[123] Thomas A. Henzinger, Anmol V. Singh, Vasu Singh, Thomas Wies, and Damien Zufferey. Static Scheduling in Clouds. In *USENIX HotCloud*, 2011.

[124] Quad-core smartphones: This is their year. `http://cnet.co/xvlHX5`.

[125] Coremark benchmark. `http://www.coremark.org/`.

[126] C. Peng, S. Lee, S. Lu, H. Luo, and H. Li. Traffic-Driven Power Saving in Operational 3G Cellular Networks. In *ACM MobiCom*, 2011.

[127] Justin Manweiler, Sharad Agarwal, Ming Zhang, Romit Roy Choudhury, and Paramvir Bahl. Switchboard: a matchmaking system for multiplayer mobile games. ACM MobiSys, 2011.

[128] Andrew Krioukov, Prashanth Mohan, Sara Alspaugh, Laura Keys, David Culler, and Randy Katz. NapSAC: Design and Implementation of a Power-Proportional Web Cluster. In *Workshop on Green Networking*, 2010.

[129] Green grid data center power efficiency metrics: PUE and DCIE. `bit.ly/MioRIt`.

[130] Sebastian Herbert and Diana Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. ISLPED '07, 2007.

[131] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2004.

[132] JR. E. G. Coffman, M. R. Garey, and D. S. Johnson. An Application of Bin-Packing to Multiprocessor Scheduling. *SIAM Journal of Computing*, 7(1), Feb 1978.

[133] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *USENIX OSDI*, 2004.

[134] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. CloneCloud: elastic execution between mobile device and cloud. ACM EuroSys, 2011.

[135] Tatsurou Sekiguchi, Hidehiko Masuhara, and Akinori Yonezawa. A Simple Extension of Java Language for Controllable Transparent Migration and Its Portable Implementation. COORDINATION, 1999.