Stony Brook University

On Selected Problems on Wireless Networks

A Dissertation Presented

by

Navid Hamed Azimi

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

May 2014

Stony Brook University

The Graduate School

Navid Hamed Azimi

We, the dissertation committee for the above candidate for the Doctor of Philosophy degree, hereby recommend acceptance of this dissertation.

Himanshu Gupta – Dissertation Advisor
Associate Professor, Department of Computer Science

Samir Das – Chairperson of Defense
Professor, Department of Computer Science

Vyas Sekar
Assistant Professor, Department of Computer Science
Carnegie Mellon University

Jon Longtin
Professor, Department of Mechanical Engineering
Stony Brook University

This dissertation is accepted by the Graduate School.

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

On Selected Problems on Wireless Networks

by

Navid Hamed Azimi

Doctor of Philosophy

in

Computer Science

Stony Brook University

2014

This dissertation is a collection of several pieces of mostly unrelated projects conducted during the several past years in Wings lab in Department of Computer Science in Stony Brook University. The only common theme of these various project is the use of wireless links as the underlying medium of communication and building block of the network.

The first chapter is dedicated to the pice de rsistance of this thesis. The chapter explores a practical and cost competitive method for constructing a mostly wireless data center network. Conventional wired data center (DC) network designs offer extreme cost vs. performance tradeoffs. Recent results make the case for a promising alternative to the current dominant architectures where an oversubscribed network is augmented with reconfigurable inter-rack wireless or optical links.

Inspired by the promise of reconfigurability, the chapter presents FireFly, an inter-rack network solution that pushes DC network design to the extreme on three key fronts: (1) all links are reconfigurable; (2) all links are wireless; and (3) the non-ToR switches are eliminated altogether. This vision, if realized, can offer significant benefits in terms of increased flexibility, reduced equipment cost, and minimal cabling complexity.

In order to achieve this vision, one need to look beyond traditional RF wireless solutions due to their large interference footprint which limits range and data rates. Thus, this work make the case for using free-space optics (FSO). The chapter demonstrates the viability of FSO by building a proof-of-concept prototype of a steerable small form factor FSO device using commodity components. In addition, it address practical algorithmic and system-level challenges in network design and management to near-optimally leverage the benefits of the FireFly vision.

The second chapter addresses the problem of preserving generated data in a sensor network in case of node failures. The chapter focus on the type of node failures that have explicit spatial shapes such as circles or rectangles (e.g., modeling a bomb attack or a river overflow). Two different schemes for introducing redundancy in the network is considered, simply replicating data or by using erasure codes, with the objective to minimize the communication cost incurred to build such data redundancy. It is proven that the problem is NP-hard using either replication or coding. A $O(\alpha)$-approximation algorithm for each of the schemes is proposed, where α is the "fatness" of the potential node failure events. In addition, a distributed approximation algorithm using erasure codes is designed. Simulation results show that by exploiting the spatial properties of the node failure patterns, one can substantially reduce the communication cost, compared with resilient data storage schemes in the prior literature.

The third chapter studies the effect of introducing delayed scheduling of user traffic on the performance of broadband cellular networks. The work is motivated by the studies which have indicated the traffic load on the cellular base stations varies significantly over time. This gives an opportunity to accommodate additional traffic with the same network capacity if some of the traffic (e.g., p2p, cloud sync) can be amenable to delayed scheduling without hurting the user experience any significantly. In this chapter, various algorithmic problems that can arise in this context is studied. Using a model where all flows can have certain flexibility in scheduling (via use of a deadline), optimal or near-optimal algorithms to determine the minimum network capacity for two different models is developed. In addition, various semi-online and online algorithms for online scheduling of flows, and analyze their performance are developed.

In particular, even though the online scheduling problem is shown to be intractable, the proposed semi-online algorithm can schedule flows optimally if aided by historical data and slightly additional network capacity over the optimal. Finally, using flow level traffic traces collected at the core of a commercially operated cellular network, the effectiveness of these techniques is

evaluated. Evaluations show that delayed scheduling, when done efficiently (using an offline optimal algorithm), can accommodate the same traffic with much lower network capacity (up to 50only modest delays. While such an optimal solution needs an offline approach, one can demonstrate that online scheduling can be almost equally effective when historical traffic data can be exploited for estimation purposes.

The fourth chapter studies the problem of channel assignment in femtocells. Femtocells are short-range devices deployed to provide increased coverage and capacity in a small area. They offer a way to increase the capacity of a cellular network by relaying cellular traffic to the wired network. In this chapter, the problem of optimizing the overall capacity of a femtocell network, as defined by Shannon's law and physical interference, by appropriate power and channel assignment to the femtocells is studied. In particular, an approximation algorithm for the objective of maximizing the total network capacity is designed for large uniform networks with arbitrary coverage regions. The second objective of maximizing the minimum capacity at a femtocell in the network is considered, and an algorithm for arbitrary networks is designed which has an appropriate performance guarantee if there is a lower-bound on the distance of any two femtocells. Through simulations, it is demonstrated the performance of our designed algorithms by comparing them with a bound on the optimal values.

The fifth chapter addresses the problem of optimal spectrum management in continuous frequency domain in multiuser interference channels. The objective is to maximize the sum of user capacities. The main results are as follows: (i) For frequency-selective channels, it is proven that in an optimal solution, each user uses maximum power; this result also generalizes to the case where the objective is to maximize the product of user capacities (i.e., proportional fairness). (ii) For the special case of two users in flat channels, we solve the problem optimally.

Contents

List of Figures

List of Tables

Acknowledgements

A major research project like the one I carried out in the twilight years of my Ph.D. years is never the work of anyone alone. The contributions of many different people, in their different ways, have made this possible. I would like to extend my appreciation to each and everyone of them.

Foremost, I would like to express my sincere gratitude to my advisor Prof. Himanshu Gupta for the continuous support of my Ph.D study and research, for his patience, motivation and enthusiasm. His guidance helped me in all the time of research and writing of this thesis and I can not imagine having a better advisor and mentor.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Samir Das, Prof. Vyas Sekar, and Prof. Jon Longtin, for their encouragement, insightful comments and guidance.

Last but not the least, I would like to thank my parents for giving me the freedom, the support and the knowledge to find my way in this world.

Chapter 1

FireFly: A Reconfigurable Wireless Data Center Fabric using Free-Space Optics

1.1 Introduction

A robust data center (DC) network must satisfy several goals: high throughput [1, 2], low equipment and management cost [1, 3], robustness to dynamic traffic patterns [4–7], incremental expandability [8, 9], low cabling complexity [10], and low power and cooling costs. With respect to cost and performance, conventional designs are either (i) overprovisioned to account for worst-case traffic patterns, and thus incur high cost (e.g., fat-trees or Clos [1, 2, 11]), or (ii) oversubscribed (e.g., simple trees or leaf-spine architectures [12]) which incur low cost but offer poor performance due to congested links.

Recent work suggests a promising middleground that augments an oversubscribed network with a few reconfigurable links, using either 60 Ghz RF wireless [5, 6] or optical switches [7]. Inspired by the promise of these flexible DC designs,[1] we envision a radically different DC architecture that pushes the network design to the logical extreme on three dimensions:

1. All inter-rack links are flexible;
2. All inter-rack links are wireless;
3. We get rid of the core switching backbone!

This extreme vision, if realized, promises unprecedented qualitative and quantitative benefits for DC network operators and applications. First, it can reduces infrastructure cost without compromising on performance. Second, topological flexibility increases the effective operating capacity and can improve application performance by alleviating transient congestion. Third, it unburdens DC operators from dealing with operational headaches from cabling complexity and its attendant overheads (e.g., obstructed cooling) [10]. Fourth, it can enable DC operators to experiment with, and benefit from, new topology structures that would otherwise remain unrealized because of cabling costs. Finally, the ability to flexibly turn on/off links can take us closer to the vision of energy proportionality (e.g., [13]).

This chapter describes FireFly ,[2] a first but significant step toward realizing this vision in practice. Figure 1.1 shows a high-level overview of FireFly. Each ToR is equipped with reconfigurable wireless links which can connect to other ToR switches. However, we need to look beyond traditional radio-frequency (RF) wireless solutions (e.g., 60GHz) as their interference characteristics limit range and capacity. Thus, we envision a new use-case for Free-Space Optical communications (FSO) as it can offer very high data rates (tens of Gbps) over long ranges using low transmission power and with low interference footprint [14]. A logically centralized FireFly controller reconfigures the topology and forwarding rules to adapt to changing traffic patterns.

[1] We use the terms flexible and reconfigurable interchangeably.

[2] FireFly stands for Free-space optical Inter-Rack nEtwork with high FLexibilitY.

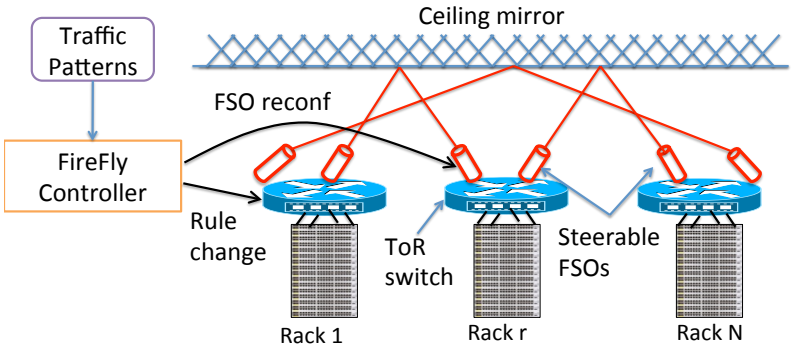


Figure 1.1: High-level view of the FireFly architecture. The only switches are the Top-of-Rack (ToR) switches.

While prior work made the case for using FSO links in DCs [15, 16], these fail to establish a viable hardware design and also do not address practical network design and management challenges that arise in reconfigurable designs. Our work bridges this gap along three dimensions:

- **Practical steerable FSO devices (S1.3):** Commodity FSO designs are bulky, power-hungry, and offer fixed point-to-point links. Our vision imposes new form-factor, cost, and steerability requirements which are fundamentally different from traditional FSO use-cases. To this end, we establish the viability for a small-form factor FSO design which repurposes commodity optical devices. We demonstrate two promising "steering" technologies using switchable mirrors [17] and Galvo mirrors [18].
- **Network provisioning (S1.4):** Given budget and physical constraints, the FireFly network hardware must be suitably provisioned to handle unforeseen traffic patterns. Here, we argue that flexible network designs should strive to optimize a new notion of dynamic bisection bandwidth. While it is hard to analytically reason about network topologies that optimize this metric, we show that random regular graphs are surprisingly good in practice.
- **Network management (S1.5,S1.6):** The FireFly controller needs fast and efficient topology selection and traffic engineering algorithms to optimally adapt to changing traffic conditions. However, state-of-art off-the-shelf solvers fail to scale beyond 32-rack DCs. Thus, we develop fast heuristics that achieve near-optimal performance (S1.5). In addition, the FireFly controller must ensure that performance is not adversely impacted during reconfigurations. We design simple but effective mechanisms that ensure that the network always remains connected, there are no black holes, and that the per-packet latency is bounded.

We evaluate our FSO prototype using a range of controlled lab experiments and a longitudinal study in a real DC setting. We find that the links are robust to

real environmental disturbances and achieve wireline-equivalent throughput, and the steering mechanisms are fast, precise, and accurate. We evaluate the end-to-end performance of FireFly using a combination of detailed packet-level simulations [19], large scale flow-level simulations, and virtual emulation platforms [20]. We compare FireFly against state-of-art "augmented" designs and overprovisioned DC architectures. Overall, we find that FireFly can achieve close-to-optimal performance of a full bisection bandwidth network at 40-60% cost, and can outperform existing augmented architectures by up to $1.5\times$ on realistic workloads.

Even looking beyond the (favorable) cost and performance, we believe there is value in exploring this research agenda—FireFly is an enabler that fundamentally changes DC designs by eliminating the "pain points" of constantly managing/upgrading core switches, rethinking topologies for higher data rate, and dealing with cabling issues! We hope that our work establishes that all-wireless, coreless, and fully flexible designs are not pipe dreams, and that the technology to achieve this vision is within our reach.

1.2 Motivation and Overview

We begin with motivating key aspects of our vision: full flexibility, wireless links, and use of free-space optics.

1.2.1 Case for Full Flexibility

A key intuition behind FireFly's design is that a fully-flexible inter-rack network can yield near-optimal bisection bandwidth even without any core (non-ToR) switches.

To provide the basis for this intuition, we consider an abstract model of a DC network with n racks (and hence n ToR switches) and l servers per rack. We consider two abstract DC designs: (a) FBB: a full-bisection bandwidth network, and (b) Flexible(f): an architecture with only ToR switches, each of which has f flexible ports that can be rewired to connect another ToR switch. (The flexible ports are in addition to the ports connected to the servers.)

Next, we evaluate the evacuation time (i.e., time to fully satisfy) a given inter-rack traffic matrix. Computing evacuation time for FBB is straightforward as there is no congestion. For Flexible(f), we can compute the evacuation time optimally by computing a sequence of reconfigurations (rewiring plus traffic engineering) using bi-partite matchings. The reconfigurations are done at most once for each non-zero matrix entry.

Using these algorithms, we compute the evacuation times over a range of synthetically generated demand matrices. Figure 1.2 show the normalized performance and cost of Flexible(f) w.r.t. FBB, as we vary the number of flexible ports on each

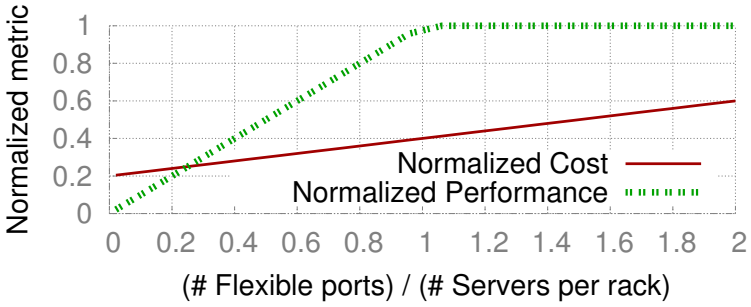


Figure 1.2: A fully-flexible network can offer optimal (full-bisection) performance at a fraction of the cost.

ToR. We model the cost as a (constant) multiple of the total number of ports in the architecture. The key takeaway is that the coreless fully-flexible Flexible(f) architecture yields optimal performance when f is equal to the number of servers per rack. At this point, its cost is only 40% of FBB; for a complete FatTree (i.e., $n = l^2/2$ [1]. Note that this result is independent of the number of racks, number of servers/rack, and the traffic distribution.[3]

An actual realization of Flexible(f) will be less optimal because of limited flexibility, non-zero reconfiguration latency, and other system-level inefficiencies. We show that our instantiation of Flexible(f) via FireFly results in only a minimal degradation in this cost-performance tradeoff.

1.2.2 Case for Wireless via Free-Space Optics

To realize a Flexible(f)-like inter-rack network, conceptually we need a "patch-panel" between racks. Of course, this is infeasible on several fronts: (1) it requires very high fanout and backplane capacity (potentially nullifying the cost benefits), (2) the cabling complexity would be high [10], and (3) it introduces a single-point of failure [4, 21]. Thus, we turn to reconfigurable wireless links between the ToR switches.

Given the trajectory of prior work, the seemingly natural solution is RF-based wireless (e.g., 60GHz) [5, 6]. However, eliminating interference in RF links is hard, even with highly directional antennas that use narrow beams [5, 6, 22]. In effect, we need RF beams with angular divergence of about 1 milliradian, which in turn requires antennas of unrealistic sizes—a few meters in size even in microwave frequencies.This issue is fundamental due to the large wavelengths of RF and holds for any RF band or technology. In addition, regulations over RF bandwidth and transmit power further limit achievable data rates.

[3]We can analytically show that the normalized performance of Flexible(f) w.r.t. FBB is $\min(f/l, 1)$. We refer the interested reader to Appendix A.1.

Such RF-specific limitations can be circumvented if we use a different part of the EM spectrum with much lower wavelengths. In particular, free-space optics (FSO) is a relatively established technology that uses modulated visible or infrared (IR) laser beams [14]. Unlike traditional optical links, the laser beam is transmitted through the air, instead of being enclosed in a fiber. It is possible to create very narrow FSO beams resulting in minimal interference and power attenuation. Further, optical spectrum is unregulated, with no bandwidth limitations. Thus, FSO links can easily offer Gbps–Tbps bitrates at long distances (several kms) using low transmit power (few watts) [14, 23, 24].

1.2.3 FireFly System Overview

Building on the previous insights, FireFly uses a fully-flexible inter-rack fabric enabled by wireless FSO links (Figure 1.1). FireFly uses traditional wires for intra-rack connections. We assume an out-of-band control network to configure the ToR switches and the FSO devices (e.g., [25]).

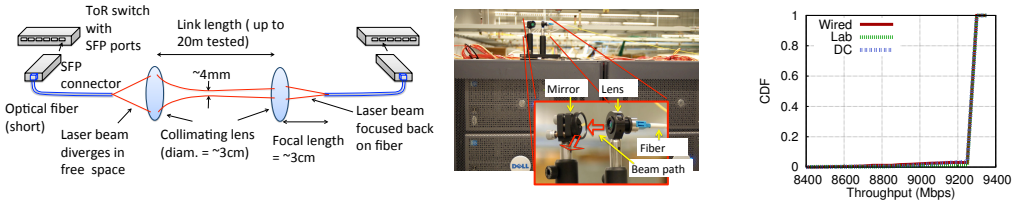
FSO Links. Each ToR is equipped with a number of steerable FSO devices. We exploit the space above the racks to establish an obstruction-free optical path. To ensure that the FSO devices do not obstruct each other, we use ceiling mirrors [5] as shown in Figure 1.1. The requirements of such mirrors are quite minimal and conventional mirrors work sufficiently well ($S1.3$).[4] The FSO devices export APIs to the controller for reconfiguration.

Network Provisioning. In the limit, we would like to have a very large number of FSO devices per ToR. In practice, however, there are physical and geometric constraints. For instance, FSO devices will have a finite size that constrains the number of such devices per ToR. Thus, we need to suitably provision or preconfigure the network so that the network is robust to future (and unforeseen) traffic patterns.

Network Management. The FireFly controller dynamically selects the runtime topology and configures forwarding paths, based on prevailing traffic demands and events. Following prior work, we leverage software-defined networking (SDN) capabilities to implement the FireFly data plane [26–28]; i.e., each ToR switch in FireFly is SDN-capable. Each SDN-capable ToR switch also reports observed traffic demands to inform the controller logic. FireFly can use other demand estimation algorithms; e.g., host buffer sizes [28] or new switch features [27]. Since our focus is on FireFly-specific aspects, we do not discuss these extensions.

In the following sections, we describe the design of a viable steerable FSO link, network preconfiguration, and run-time network management.

[4] One alternative to a ceiling mirror is to vertically position the FSO devices on each ToR at different heights to avoid obstructing each other. We do not explore this in this work.



(a) FSO link design (b) DC set up (c) TCP throughput

Figure 1.3: FSO link design, prototype and performance. (a) Optical path and associated networking gear (drawing not to scale). (b) One end point of the 10 Gbps link prototype running in a DC over $\approx$ 20m); the inset shows a zoomed in version. The FSO link connects to a similar set up on the other end. (c) Distribution of per-sec TCP throughputs on a 10 Gbps FSO link over $\approx$ 20m on optical bench (Lab) and DC racks (DC), over days of continuous runs. Throughput distribution on wired optical fiber is used for comparison.

1.3 Practical Steerable FSO Design

In order for the FireFly vision to deployed in a DC, the FSO devices must ideally have a small form factor (e.g., so we can pack several devices on each ToR), be low-cost commodity devices (e.g., as we envision thousands of such devices in a single DC), with low power footprint relative to traditional switches, and be steerable to enable flexibility.

At first glance, these requirements seem to be fundamentally at odds with the trajectory of today's commercial FSO devices [29]. They are bulky, expensive, and power-intensive. The main technical challenge has been that achieving robust links at high data-rates and long ranges (a requirement in both traditional deployments and FireFly) is hard. It has traditionally required relatively powerful lasers, expensive and custom mechanisms for dynamic alignment for outdoor use. The problem in context of FireFly is even more challenging, since FireFly requires steerable links while the conventional FSO deployments target fixed point-to-point links.

In this section, we demonstrate (perhaps surprisingly) that (a) it is viable to repurpose commodity DC-centric optical networking gear to establish robust and sufficiently long FSO links in a DC, and (b) we can leverage existing commodity optical technologies to steer the FSO beam with high precision and low latency.

1.3.1 FSO Link Engineering

As a first step, we demonstrate that it is possible to engineer an FSO optical link using commodity DC-grade optical networking equipment that can achieve high data rates, at ranges sufficient for DC-scale deployment, and with sufficient

(mis)alignment tolerance. The intuition on why this is technically feasible without the additional overheads in commercial FSO devices, is that in the indoor controlled DC setting concerns about outdoor environmental factors largely disappear! However, we do need to carefully design the optical path to balance the tradeoff between the laser beam divergence and misalignment tolerance, as we discuss below.

We engineer the FSO system by coupling two optical fiber end points *directly* with a free-space link *without any opto-electric conversion* thus saving on both power and cost. This Fiber–FSO–Fiber link connects to standard optical interconnect technology widely used in DCs (e.g., 10GBASE-SR). A typical example of this interface is optical SFP (small form-factor pluggable) or its variants such as SFP+.

This approach requires optical designs on both ends: (i) on the transmit side, where the fiber 'launches' the laser beam in free space, and (ii) on the receive side, where the laser beam is received into the fiber (Figure 1.3(a)). Normally, when the laser beam comes out of the fiber into free space it diverges with a significantly large angle. To minimize divergence, we collimate the beam using a suitably designed lens located at its focal length from the transmitting fiber endpoint.[5] A similar lens near the receiving fiber end point focuses the beam back to the fiber.

The above optical design is done carefully to ensure that the laser beam maintains a sufficient width [31] so that *it can tolerate minor misalignments due to rack vibrations and other effects.* However, this presents a tradeoff: wider beams can tolerate misalignments better, but suffer from a poorer power density at the receiving end. The design we develop shows that a good balance is indeed possible using optical SFPs used for long range fiber communications (e.g., 10GBASE-LR can go up to 10 km). They use highly sensitive detectors that can work with very little received power. Our current prototype (described below) has been tested for up to about 20 m link length suitable for a small DC. The 20 m tests are primarily a limitation of our lab set up, but the general design can easily extend to 100 m (see Appendix A.2).

This approach satisfies all the design requirements except steering. The lens is small (about 3 cm diameter) with focal length about the same. Even considering additional hardware (e.g., mounts or adapters), the footprint of the assembly is only 5 cm across. The costs are also very modest when procured in volume: $\approx$ \$50 for the lens and \$50 for the assembly. We acknowledge that there might be an additional cost of using optical SFP ($\approx$\$100), if (wired) optical links are not already used. Finally, there is no additional power burden beyond the SFP power consumption as the design does not add any new opto-electronic conversion.

Prototype. We have developed a proof-of-concept prototype following the design

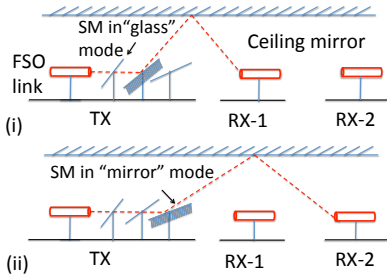
[5]For brevity, we skip the details of the optical design. We only state the basic approach and relevant tradeoffs. A similar approach has been used in [30] but in a different context.

outlined above. We have been able to successfully use both 1 Gbps and 10 Gbps links with very similar optical set ups. The prototype links use 1000BASE-LX (1GBASE-LR) SFP (SFP+) for the 1 Gbps (10 Gbps) case and multi-mode fibers for their larger diameter. We first use standard optical bench set up to validate the design using controlled experiments and then test the link in a production DC. Figure 1.3(a) shows the general setup.[6] For the current design the collimated laser beam maintains a ≈4 mm diameter after it converges. To get up to a 20 m length on a small optical bench, we use a standard technique used by optical engineers: the beam path is made to reflect multiple times back and forth via mirrors. *This also validates use of mirrors on the beam path and demonstrates that any optical loss due to reflections is well tolerated.*

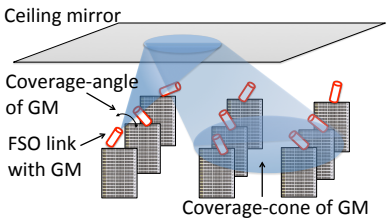
Link Performance. We test the link by running continuous TCP transfers over the FSO link for several days at a time for several selected link lengths with the set up on the optical bench. The results are very similar for different lengths. For brevity we only report the results for the longest tested case (≈20m) for the 10 Gbps link. See Figure 1.3(c). Note that the distribution of TCP throughputs is almost identical to that observed over regular fiber links, demonstrating no additional loss in the FSO links. To study misalignment tolerance, we shift the transmit side set up in tiny incremental steps (using a translating mount) perpendicular to the beam axis keeping the receive side fixed. We see no throughput loss until 6 mm shift, beyond which the link becomes unstable. As we will see below, this 6 mm tolerance is sufficient to handle minor misalignments due to rack vibrations and environmental issues in a DC.

To understand the link performance "in the wild," we set up the link in a production (university run) DC environment. Unlike the optical bench, this real environment has several key differences that can produce mis-alignments: (1) racks experience vibrations due to several factors (e.g., server fans, discs, HVAC and UPS units [33]) and (2) the beam could 'wander' due to fluctuating air density caused by temperature variations. We set up the FSO link with the optical components placed on top of the rack using magnetic bases. Two racks are used at ≈20 m apart. See Figure 1.3(b). We use mirrors on the beam path - one on each end - for ease of alignment. The reader can view these mirrors as proxies for mirrors to be used in steering (next subsection). Alignment is done manually with the help of an infrared viewer (more on this in $S1.9$). The TCP transfer experiment is run continuously over several days as before. The statistics of per-sec TCP throughput is almost identical again to the optical bench and wired cases (Figure 1.3(c)). This estab-

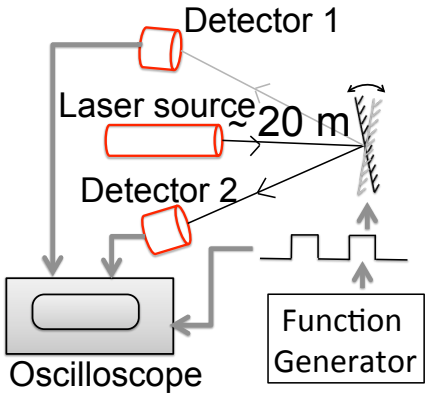
[6]Note that typical optical SFPs require two fibers and thus two optical paths for duplex communication. However, single fiber SFPs [32] that use WDM principles to multiplex both links on a single fiber are beginning to be available. For 1 Gbps experiments we use such SFPs. Due to current unavailability of such devices for 10 Gbps, we use fiber for the return path for the 10 Gbps experiments; all reported performance measurements are from the FSO path.



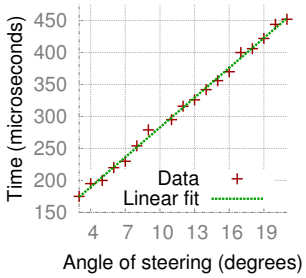
(a) Switchable mirror operation



(b) Galvo mirror operation



Experimental set up



Response time

(c) Galvo mirror response

Figure 1.4: (a) Using switchable mirrors (SMs) with FSOs. (i) The 2nd SM is in mirror mode (steering the beam to RX-1). (ii) The 3rd SM is in mirror mode (steering the beam to RX-2). RX-s are also equipped with aligned SMs to direct the beam to its detector, but not shown for clarity. (b) Galvo mirror (GM) on an FSO device can steer the beam within its coverage-cone. (c) Evaluating Galvo mirror response.

lishes the potential of our design. The average TCP throughput is ≈ 9.3 Gbps — note that *no commodity RF technology exists in small-form factor that can deliver such throughput at 20 m range.*

In summary, the above design is the basis for a low-cost, commoditizable, and small-form factor FSO link at high data rates over ranges sufficient for DC scale. Our experiments suggest that the link is likely robust to realistic (mis)alignment concerns due to environmental effects in DCs. The only remaining issue is that this link is still point-to-point and not steerable; we address this next.

1.3.2 Developing Steering Mechanisms

Having established the viability of a point-to-point FSO link using commodity devices, the only requirement left is to make the beam steerable to enable flexibility to point to other racks. Our goal is to establish a design roadmap that is commoditizable. We explore two promising solutions: switchable mirrors and Galvo mirrors. We do not claim these are optimal in any sense or that the only steering alternatives. Our choice is pragmatic in that we want to establish a feasible roadmap using off-the-shelf components. (There are a variety of other beam steering approaches [34], but they are not off-the-shelf technologies.) Both solutions offer different tradeoffs w.r.t. latency, degree of flexibility, and cost/power, and at this time, no one solution is strictly better. Thus, we believe it is instructive to understand and evaluate the promise of both alternatives and the tradeoffs they offer.

Switchable Mirrors (SMs). Switchable mirrors (SM) are made from a special liquid crystal material that can be electrically controlled to rapidly switch between reflection (mirror) and transparent (glass) states at millisecond timescales [17]. While the intended use cases are different (e.g., rear-view mirrors that switch between a regular mirror and a back-up camera display), we can use them for beam steering as shown Figure 1.4(a).

Each FSO device has multiple SMs, with each SM aligned (during a preconfiguration step as discussed in $S1.4$) to target a point on a ceiling mirror and thus, a receiving FSO. The link is established by switching one of the SMs to the mirror state, while leaving the rest in the transparent state. (This is done at both ends, but we only show transmit side for clarity.)

SMs directly satisfy our design requirements. It can be miniaturized as it only needs to be slightly larger than the beam diameter: 1 cm^2 is sufficient. A SM of this size is expected to have low cost ($<$ \$5) at volume [35]. Power consumption is also low: only 40 mW for the stated size [17]. We have evaluated the reconfiguration latency using an off-the-shelf 12" x 15" SM [17], and it is $\approx$250 msec. The switching

latency decreases with the decrease in the surface area, and is estimated [35] to be ≈10-20 msec latency for the 1 cm^2 SM size we envision.

Galvo Mirrors (GMs). Galvo mirrors (GMs) [18] are typically used in laser scanning applications. Here, a small mirror, few mm across, rotates (up to specific angular limits) around an axis on the plane of the mirror in response to an electrical signal. The laser beam is made to reflect from this mirror. The mirror rotation deflects the reflected beam by a specified angle depending on the signal. Using a pair of such mirrors at right angles, we can steer the beam within a desired rectangular cone. In our context, equipping an FSO device with a GM enables us to target any receiver within a pre-configured rectangular cone chosen offline. See Figure 1.4(b).

As proof of concept, we evaluate the response parameters of GMs using an off-the-shelf GM [36] using the setup shown in Figure 1.4(b). The mirror rotation is controlled programmatically changing the applied voltage. Here, two detectors receive the reflected beam from the mirror alternately as the mirror is fed by a square wave (100 Hz) from a function generator. We measure the time between the instant the voltage trigger is initiated (via the square wave generator) and the time the mirror settles to its new position. Figure 1.4(c) shows that the steering latency is linear w.r.t. the steering angle and ≤ 0.5 ms even for angles up to about $\pm 20^{\circ}$. We measured the pointing error to be $\leq 10\,\mu$rad, which translates into ≈1 mm positioning error at 100 m, which is well within the 6 mm tolerance of the FSO link.

The GM is inexpensive (≈$100) and small (few inches across). But, off-the-shelf GMs have a somewhat higher average power consumption (7 W measured) due to the use of an electro-mechanical system. That said, MEMS-based scanning mirrors that provide the same functionality as GMs are already being commoditized [37] and can reduce the power to a few milliWatts.

1.3.3 Design Summary

Our hardware design and experiments establish the viability of using FSO in the DC and also confirm the promise of both SM and GM-based approaches. In summary, the device roadmap we outlined will have: (1) a rough form factor of 3"x6"; (2) a range of ≈100m and a misalignment tolerance of 6mm; (3) a power footprint of 3W (most of this is in SFP, assuming MEMS-based GMs); and (4) an estimated per-port cost of 300$ (100$ for the SFP and 200$ for the FSO+steering when produced in volume).

The two steering mechanisms introduce slightly different constraints and trade-offs for the FireFly network design (discussed next): (1) k SMs at an FSO can switch the FSO beam between a set of k arbitrarily chosen but *pre-aligned* receivers, and (2) A GM on an FSO can steer the beam to any receiver within the coverage-cone that the GM has been *pre-oriented* to target.

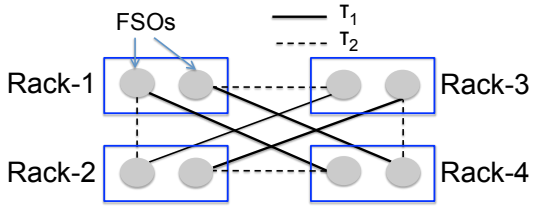


Figure 1.5: A PCFT with candidate links (solid and dashed). The set of solid links (when active) represents one possible realizable-topology (τ_1), and the set of dashed lines represents another (τ_2).

1.4 Network Pre-Configuration

Ideally, we want a dense flexible network by placing a large number of FSO devices on each rack, with each FSO device equipped with a large number of SMs or high-coverage GMs. In practice, we have physical limitations, e.g., the size/cost of the FSO devices, size of SM, angle of GMs etc. Given these constraints, our goal is to design a high performance DC network.

At a high level, there are two network design problems in FireFly, that occur at different timescales:

- First, we need to provision the network hardware (e.g., how many FSOs) and also pre-align/pre-orient the SMs/GM at each FSO. This is done offline in a **pre-configuration** phase.
- Second, given a pre-configured network, we need to reconfigure the network in near real-time to implement a runtime topology suited for the current traffic.

We focus on the first problem in this section, and defer the second problem to the next section.

1.4.1 Preliminaries and Objective

Preliminaries. Consider a FireFly network, i.e., a set of FSOs on each rack with pre-aligned SMs or pre-oriented GMs. We can establish a candidate (bi-directional) link between a pair of FSOs a and b if (i) a has an SM aligned towards b and vice-versa or (ii) a is located in the coverage-cone of the GM at b and vice-versa. At any instant, only one candidate link per FSO can be an active link. For example, in Fig. 1.4(a), links (TX, RX-1) and (TX, RX-2) are candidate links, and link (TX, RX-1) is active in Fig. 1.4(a)(i) while (TX, RX-2) is active in Fig. 1.4(a)(ii).

We refer to the set of all candidate links as the pre-configured flexible topology (PCFT). Given a PCFT, we refer to a set of candidate links that can be active simultaneously as a realizable-topology. Because only one candidate link per FSO can be active at any time, a realizable topology is a matching in the PCFT graph over FSOs (Figure 1.5).

Metric of Goodness. Our goal is to provision the DC network with a PCFT that can deliver the best performance. If we knew the expected (set of) traffic demands, then we can design a customized PCFT. However, given that DC workloads are variable and unpredictable [2], we need a traffic-oblivious analogous to the traditional bi-section bandwidth metric [38]. Unfortunately bisection bandwidth only applies to a static topology, and is not meaningful for a flexible network. More formally, given a topology t and considering all possible partitions P of t into two equi-sized sets of racks, the bisection bandwidth is defined as $\min_{p \in P} \mathrm{BW}(t, p)$, where $BW(t, p)$ is the cut-size in t corresponding to p. In a flexible design, the topology t can be changed, and the bisection bandwidth notion fails to capture this aspect.

Thus, we introduce a new notion of dynamic bisection bandwidth (DBW) as the metric of goodness to evaluate a PCFT. The dynamic bisection bandwidth of a PCFT Π can be defined as follows. Let T be the set of realizable-topologies of a given PCFT Π. Then, the dynamic bisection bandwidth (DBW) for a PCFT Π is defined as: $\min_{p \in P} \max_{t \in T} \mathrm{BW}(t, p)$. Note that this reflects the ability to choose the best realizable-topology t for each given partition p.

To illustrate this, consider the PCFT in Figure 1.5 again. If we consider τ_1 (solid lines) as a static topology, its bisection bandwidth is zero due to the partition $\{$(2,3), (1,4)$\}$ of racks. Similarly, the bisection bandwidth of τ_2 (dashed lines) can be seen as 2. However, the DBW of the overall PCFT is 4, since τ_1 yields a bandwidth of 4 for all equi-partitions except for $\{$(2,3), (1,4)$\}$, for which τ_2 yields a bandwidth of 4.

Constrained Optimization. Our goal is to design a PCFT that operates within the given cost and physical constraints and optimizes the DBW. For clarity, we focus on the SM and GM problems independently in this chapter and defer hybrid architectures for future work (S1.9). In each case, we solve the overall budgeted PCFT selection problem in two steps. First, we develop techniques to design a PCFT with maximum DBW for a fixed configuration (i.e., fixing #FSOs, coverage angle, and #SMs per FSO). Then, given the price/size constraints, we exhaustively search the space of feasible combinations of these network parameters and pick a feasible PCFT with the highest DBW. Since preconfiguration runs offline, this brute-force step is reasonable.

1.4.2 SM-PCFT Design Problem

Problem Formulation. Given the number of racks n, number of FSOs m per rack, and the number of SMs k per FSO, the SM-PCFT problem is determine the alignments of each SM such that the resulting PCFT has maximum DBW.

Said differently, we want a PCFT with maximum DBW, under the constraint that the number of candidate links at each FSO is at most k. From this view, the

SM-PCFT problem falls in the class of network design problems [39], but is different from prior work due to the novel DBW objective. For instance, even the special case of $k = 1$, the SM-PCFT problem reduces to constructing an m-regular graph over n nodes with maximum (static) bisection bandwidth. Even this simple case is harder than the well-studied problem of determining an upper-bound on the bisection bandwidth of m-regular graphs of size n, for which approximate results are known only for very small values of m and n [40]

Random Graphs for SM-PCFT. One promising approach to constructing a SM-PCFT solution is to consider random regular graphs. This is based on the intuition that graphs with (static) bisection bandwidth are likely to have high DBW. (Because random graphs have near-optimal spectral gap [41], they are good "expanders" and have high static bisection bandwidth.) We can construct an n-node regular graph of degree mk, and then group the mk edges on each node into m sets of k edges each (corresponding to each of the m FSOs). For every edge connecting a pair of FSOs (a, b), we align one SM each of a and b towards each other. Because of the randomness, there is a small chance of some random instance performing poorly; thus, we generate many different solutions, and pick the one with the best DBW.[7]

1.4.3 GM-PCFT Design Problem

Problem Formulation. Given the data center layout, the number of racks n, number of FSOs per rack m, and uniform coverage-angle (see Fig.1.4(b)) of GMs, the GM-PCFT problem is to determine the orientation of the GM on each FSO such that the resulting PCFT has the maximum DBW.

Note that we cannot directly use a random graph as a GM-PCFT solution, since an FSO a's neighbors in a PCFT must be colocated in a coverage-cone of the GM at a. Thus, this problem imposes certain geometric constraints. In particular, for a pair (a, b) to form a (bi-directional) candidate link in the resulting PCFT, the coverage-cone of GM at a must cover b and vice-versa. A naive approach is to iteratively pick a pair of FSOs (a, b) at a time and orient their GMs towards each other. However, this approach may create only one candidate link per FSO/GM, and hence, could result in a sparse PCFT with poor DBW.

Block-based Heuristic. To address the shortcomings of the above strawman approach, we use a "block"-based approach. The intuition here is to create a random graph at a coarser block granularity, where each block is a group of nearby FSOs that fall within a GM's coverage cone.

[7] One subtle issue is even computing DBW is hard. To estimate the DBW for a given random instance, we extend the Kernighan-Lin [38] heuristic for estimating the bisection bandwidth. Our experiments suggest this is within 5-7% of the true DBW. Due to space constraints, we do not discuss the DBW estimation in depth.

The approach runs in m iterations, and in each iteration we fix the orientation of the GM on the i^{th} FSO of each rack, as described below. (The numbering of FSOs is arbitrary; we just need some ordering.) In each iteration, we randomly partition the set of racks into disjoint blocks. The only requirement here is that each block of racks is colocated and small enough to be covered by a GM (when oriented appropriately) on any FSO in the DC. That is, for each block B and FSO $a \notin B$, there exists an orientation of GM at a such that all racks in B fall within its coverage cone. At first glance, this partitioning requirement may seem complex, but we observe that a simple grid-based partitioning scheme is sufficient for all practical purposes. Next, we create a random block-level matching M_i over the blocks. Now, for each edge $(B_1, B_2) \in M_i$, we orient the GM on each i-FSO in each rack within block B_1 (correspondingly B_2) towards B_2 (B_1). By construction, the partitioning algorithm guarantees that a GM on any i-FSO in B_1 can cover (with some orientation) all i-FSOs on racks in B_2.

We note that the partitioning in each iteration i can be different. In particular, we can create random partitioning schemes: starting from the basic grid, we can do a random offset to create a new partitioning scheme. Finally, as in the case of SM-PCFT, we generate many randomized GM-PCFT solutions, and pick the best.

1.5 Real-time Reconfiguration

We consider two types of reconfigurations in FireFly: (1) periodically optimizing the network based on estimated demands; and (2) triggered by certain network events (e.g., planned migrations or elephant flows).

1.5.1 Periodic Reconfiguration

Given a PCFT and the prevailing traffic load, the periodic-reconfiguration problem is to optimally select a realizable-topology and set up routes for the current traffic-flows.

This constrained optimization is captured by the integer linear program (ILP) shown in Figure 1.6.[8] Let κ be the set of candidate links in the PCFT, C be the (uniform) link capacity, E be the given epoch size (say a few seconds), and $D_{i,j}$ be the estimated traffic demand (volume) between a pair of racks (i, j). This demand can be obtained by using the measurement counters from the SDN switches from previous epoch(s). We use the subscripts a, b, c, d to refer to FSOs, and i, j, k to refer to racks, and $FSOs(k)$ to denote the set of FSOs on the top of rack k.

[8]This problem is much harder than the optimization problem considered in $S1.2.1$, since we were assuming arbitrary flexibility or that the PCFT was essentially a complete graph.

$$\max \sum_{i,j} T_{i,j},\ subject\ to: \tag{1.1}$$

$$\forall b: \sum_{a\ s.t\ (a,b)\in\kappa} l_{a,b} \leq 1;\ \forall a: \sum_{b\ s.t.\ (a,b)\in\kappa} l_{a,b} \leq 1 \tag{1.2}$$

$$\forall a,b: \sum_{i,j} f_{a,b}^{i,j} \leq l_{ab} \times C \times E \tag{1.3}$$

$$\forall i,j,k: \sum_{a} \sum_{b\in FSOs(k)} f_{a,b}^{i,j} = \sum_{b\in FSOs(k)} \sum_{d} f_{b,d}^{i,j} \tag{1.4}$$

$$\forall i,j: \sum_{a\in FSOs(i)} \sum_{b} f_{a,b}^{i,j} = \sum_{a} \sum_{b\in FSOs(j)} f_{a,b}^{i,j} = T_{i,j} \tag{1.5}$$

$$\forall i,j: T_{i,j} \leq D_{i,j} \tag{1.6}$$

$$\forall (a,b)\in\kappa: l_{a,b} \in \{0,1\};\ \ \forall i,j,a,b: f_{a,b}^{i,j} \geq 0 \tag{1.7}$$

Figure 1.6: ILP formulation for periodic reconfiguration.

There are two key sets of control variables: (i) The binary variable $l_{a,b}$ captures topology selection and is 1 iff a candidate link (a, b) is chosen to be active; and (ii) $f_{a,b}^{i,j}$ captures the traffic engineering (TE) strategy and captures the flow volume corresponding to the inter-rack traffic between i and j routed over the link (a, b). Let $T_{i,j}$ be the total traffic volume satisfied for the flow (i, j).

For clarity, we consider a simple objective function that maximizes the total demand satisfied across all rack pairs as shown in Eq (1.1). Eq (1.2) captures the requirement that each FSO can have at most 1 active link. Eq (1.3) ensures that the total flow on each link (on average) does not exceed the capacity. Eq (1.4) are flow conservation constraints, for each flow (i, j) and a rack k. Eq (1.5) captures the volume of the demand satisfied using a constraint over the ingress and egress racks. Eq (1.6) ensures that the volume satisfied is at most the demand, for each rack pair. Finally, we have constraints on the range of the control variables.

Unfortunately, solving the given ILP using state-of-art solvers like `Gurobi` or `CPLEX` can takes several hours (S1.8.4). Given the poor performance of current solvers, we follow a heuristic strategy and decouple the optimization problem into two stages. First, we solve the "integer" problem of selecting the active links, and then given this realizable-topology, we compute the flow routes.

Greedy Matching for Topology Selection. Recall from S1.4 that a realizable-topology is essentially a matching over FSOs in the PCFT graph. Thus, a simple starting point is to select the maximum-weighted matching, where each candidate link (a, b) is weighted by the inter-rack traffic demand $D_{i,j}$ between the racks i and j. In effect, this maximizes the total demand that can be served using direct links.

However, this can be very inefficient if the given PCFT does not have direct links between racks with high traffic demands.

The high-level idea behind our heuristic is to extend the traditional Blossom algorithm for computing the maximum matching to incorporate multi-hop traffic. Recall that the Blossom algorithm improves the matching at each stage, by computing an alternating path. We define a new "benefit" function that captures multi-hop traffic and then pick the path with the highest benefit. Specifically, we use the intuition that shorter inter-rack paths imply lower resource usage and higher network throughput [9]. Thus, we define the benefit of an alternating path L as the decrease in the weighted-sum of inter-rack distances if L were used to modify the current matching. More formally, given a matching τ, the benefit of a L, that would modify the matching τ to τ', is the total reduction in the network footprint: $\sum_{ij} D_{i,j}(h_{i,j} - h'_{i,j})$, where $h_{i,j}$ and $h'_{i,j}$ are the inter-rack distances between racks (i, j) in τ and τ' respectively (when seen as graphs over racks).[9]

We run this extended Blossom algorithm until there is no alternating path that can improve the network footprint and then output the final topology at this stage.

Flow Routing. Given a specific realizable-topology (i.e., values of $l_{a,b}$), the residual TE problem is solvable in polynomial-time as a multi-commodity flow (MCF) problem. However, even this takes hundreds of seconds on 256- or 512-rack DCs (S1.8.4), which is not acceptable.

To address this, we use a greedy algorithm to compute the values of these flow variables. Essentially, we extend the traditional augmenting-path approach for max-flow algorithms and greedily pick an augmenting path for a yet-unsatisfied commodity. We run the algorithm until no more augmenting paths can be picked; i.e., the network is saturated. From this solution, we use the "path stripping" idea to convert the values of the $f_{a,b}^{i,j}$ variables into end-to-end paths.

1.5.2 Triggered Reconfigurations

In addition to periodically reconfiguring the network, FireFly can also run more localized reconfigurations triggered by certain traffic events. Such reconfigurations may be very frequent but likely require minimal and localized (topology and flow-route) changes. We currently support two types of triggers. First, if we detect elephant flows that have sent more than 10 MB of aggregate data [27] we activate links to create a shorter or less-congested path for this flow. Second, if traffic between a particular pair of racks exceeds some configurable threshold, then we create a direct link between them, if this does not require deactivation of recently-activated and/or heavily-used links.

[9] If there is no path, we just use a large constant.

1.6 Correctness during Reconfigurations

Reconfigurations inherently cause some network flux as link activations may cause other links to be deactivated and/or forwarding table updates. This raises natural concerns about network performance during these reconfigurations.

Requirements and Challenges. A reconfiguration essentially specifies: (i) addition and/or deletion of (candidate) links from the given realizable topology, and (ii) corresponding changes to the network forwarding tables (**NFTs**). Our goal is to ensure that: (i) network remains connected at all times, (ii) there are no black holes (e.g., all forwarding table entries refer to available/usable links), and (iii) packet latency remains bounded (and thus, delivery is guaranteed).

The main challenges in implementing sound data plane strategies arise from two factors: (i) Activation or deactivation of candidate links incur a non-zero latency (few msecs); and (ii) We may need to execute reconfigurations concurrently if the triggers occur frequently (e.g., for every elephant flow arrival). At a high level, these are related to the problem of consistent updates [42, 43]. The key difference is that we can engineer simpler domain- and requirement-specific solutions rather than use more general-purpose but heavyweight techniques proposed in prior work.

1.6.1 Handling sequential reconfigurations

We begin by focusing on correctness when we are executing one reconfiguration at a time and defer concurrent execution to the next subsection.

Avoiding Black Holes

To see why "black holes" may arise, consider a reconfiguration that changes the network's (realizable) topology from τ to τ' by "steering" FSOs a and b towards each other, and in the process activating the link (a, b) and deactivating some link (a, c). Suppose the NFTs change from $\mathcal{F}$ to $\mathcal{F}'$. Now, there is a period of time (when GM/SMs at a is changing state) during which neither (a, b) nor (a, c) is available for communication. During this period, irrespective of when the NFTs get updated (say, even atomically) from $\mathcal{F}$ to $\mathcal{F}'$, some entries in the NFTs may refer to either (a, b) or (a, c), inducing black holes in the network.

Our Solution. To avoid black holes, we split a reconfiguration into multiple steps such that: (i) link deletion is reflected in the NFTsbefore their deactivation is initiated, and (ii) link addition is reflected only after the activation is complete. Thus, a reconfiguration that involves deactivation (activation) of a set of links $\bigtriangledown$ ($\bigtriangleup$) is translated to the following sequence of steps:

S1: Update the NFTs to reflect deletion of $\bigtriangledown$.

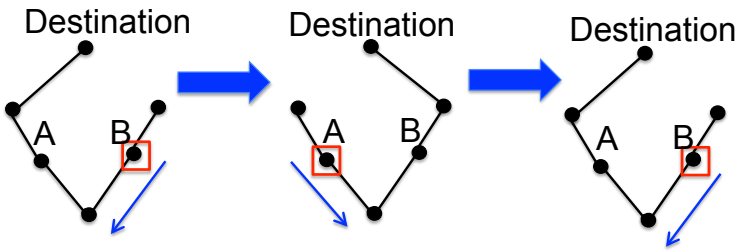


Figure 1.7: Packet (in-flight location shown by a square) continues to "swing" from B to A and back, due to a rapid sequence of reconfigurations.

S2: Deactivate $\bigtriangledown$ and activate $\bigtriangleup$.
S3: Update the NFTs to reflect addition of links $\bigtriangleup$.

One additional invariant we maintain is that every switch has a "default" low priority rule at all times to reach every destination rack via some active outgoing link. We do so to explicitly ensure that packets can reach their destination, possibly on sub-optimal paths, as long as the network is connected (see below).

Maintaining Connectivity

To ensure network connectivity at all times, we simply reject reconfigurations that might result in a disconnected network in step **S1** above. That is, we add a step S0 before the three steps above.

S0: Reject the reconfiguration, if deletion of links $\bigtriangledown$ disconnects the network.

To reduce the chance of such rejections, we also extend our reconfiguration algorithms to retain a connected subnetwork from the prior topology. The high-level idea here is to construct a rack-level spanning tree using the current graph, and explicitly remove these links/FSOs from consideration during the greedy matching step.

Bounded Packet Latency

If reconfigurations occur at a very high frequency, then we may see unbounded packet latency. Fig. 1.7 shows a simple example where a packet can never reach its destination because the links/routes are being reconfigured quite rapidly.

Our Solution. The example also suggests a natural strategy to avoid such cases—we can delay or reject reconfigurations to allow the in-flight packets to use one of the interemdiate topologies to reach its destination. We introduce a small delay of x units between two consecutive NFTs-updates, where x is the maximum packet latency in a fixed realizable topology. This ensures that each packet "sees" at most two configurations during its entire flight. This, bounds the packet latency by $(2x + z)$ where z is the total NFTs-update time.

1.6.2 Handling Concurrent Reconfigurations

Computing and executing a reconfiguration can take a few tens of msecs in a large DC. To achieve good performance, we may need to reconfigure the network frequently; e.g. for every elephant flow arrival in the network, which may happen every msec or less. Thus, we need mechanisms that allow reconfigurations to be executed concurrently. (We could also batch reconfigurations, but that merely delays the problem rather than fundamentally solving it because the batch may not complete before the next set of reconfigurations arrive.)

We observe that to handle concurrent reconfigurations, we need to extend the approach from S1.6.1 to handle two concerns.

- Connectivity: One concern of course is that each reconfiguration in isolation may not disconnect the network but combining them might. Thus, to ensure network connectivity, the controller maintains a atomic global topology variable $\mathcal{G}$, and uses this variable to accept/reject in step S0. ($\mathcal{G}$ is also updated by accepted reconfigurations in S1 and S3.)
- Conflicting reconfigurations: In step S0, we also reject any reconfiguration that "conflicts" (in terms of link activiations or deactivations) with already-accepted but yet-unfinished reconfigurations. That is, we follow a non pre-emptive strategy of allowing outstanding reconfigurations to complete.

We note that no other changes are required to S1.6.1 to handle concurrency. Black holes are still avoided since only non-conflicting reconfigurations are executed concurrently and packet latency is bounded since a minimum time-interval already precludes concurrent processing of different NFTs-updates.

1.6.3 Overall Scheme

Based on the previous building blocks, our overall scheme is as follows. Each reconfiguration ρ that deletes and adds a set of links $\bigtriangledown$ and Δ, is translated into the following four steps. Here, $\mathcal{G}$ is as described in S1.6.2.

C0: Accept ρ if (i) deletion of links $\bigtriangledown$ does not disconnect $\mathcal{G}$, and (ii) ρ doesn't conflict with any unfinished accepted reconfigurations.
C1: Update $\mathcal{G}$ and NFTs to reflect deletion of $\bigtriangledown$.
C2: Deactivate $\bigtriangledown$ and activate Δ.
C3: Update $\mathcal{G}$ and NFTs to reflect addition of links Δ.

In addition, as suggested in S1.6.1, we ensure (a) availability of default rules, and (b) a minimum time-interval of x (= maximum packet latency) units between consecutive NFTs-updates.

We can analytically prove (see Appendix A.3) that the above overall scheme ensures that (i) there are no black holes, (ii) network remains connected, and (iii)

packet latency is bounded by $(2x + z)$, where z is the NFTs-update time. This claim holds irrespective of how the NFTs are updated across the network; i.e., we do not require atomic updates.

1.7 FireFly Controller Implementation

We implement the FireFly controller as modules atop the POX controller. We chose POX/OpenFlow primarily for ease of prototyping and experimentation. We use custom C++ modules for the PCFT generation and reconfiguration optimization algorithms. For reconfiguration, we implement heuristics to "dampen" reconfigurations by checking if there is a significant (e.g., more than 10%) improvement in the objective function from Figure 1.6. We use a simple data plane translation logic to convert the output of the optimization solver. Specifically, we translate the "flow" variables into prefix-range based forwarding entries [44] and use these in the common case. For elephant flows, we set up exact flow rules to explicitly ensure that elephants traverse the intended path. We use a simple demand estimation module that leverages existing OpenFlow capabilities to estimate the inter-rack demands and uses the observed traffic from the previous epoch as the input to the controller. Our prototype does not implement elephant flow detection; for the evaluations, we currently assume this information is available out of band.

1.8 Evaluation

We established the performance of individual steerable FSO links in S1.3. In this section, we focus on:

1. Performance w.r.t. other DC architectures (S1.8.1);
2. Impact on performance during reconfigurations (S1.8.2);
3. Optimality of the preconfiguration algorithms (S1.8.3);
4. Optimality and scalability of reconfiguration (S1.8.4);
5. Sensitivity analysis w.r.t. degree of flexibility and reconfiguration latency (S1.8.5); and
6. Cost comparison w.r.t. prior DC architectures (S1.8.6).

For (1), we use a combination of detailed packet-level simulations using `htsim` and augment it with larger-scale flow-level simulations using a custom flow-level simulator. For (2), we use a detailed system-level emulation using MiniNet. For (3) and (4), we use offline trace-driven evaluations. For (5), we use the flow-level simulation. Finally, for (6) we use public cost estimates and projections from S1.3.

1.8.1 System Performance

Setup and Workloads. We consider three classes of architectures: (1) FireFly (both SM and GM) with 10Gbps links, (2) (wired) 10Gbps full-bisection bandwidth networks such as FatTree [1], and (3) augmented architectures such as c-Through [7] and 3D-Beamforming (3DB) [5] with a 5Gbps (i.e., 1:2 oversubscribed) core. (We do not compare Flyways [6] since it is subsumed by 3D-Beamforming.) By default, FireFly has 48 FSOs per rack with each equipped with 10 SMs; we assume a rack size of $4' \times 2'$, which is sufficient to hold up to 64 FSO devices (S1.3.3). We also evaluated Jellyfish [9], but do not show this for ease of visualization since the result was close to FatTree ($\approx 10\%$ lower). We assume an overall reconfiguration latency of 20 msecs for FireFly, and conservatively use zero latency for c-Through/3DB. We use ECMP routing for FatTree and backbone cores of 3dB and c-Through, and route the "overflow" traffic to their augmented links [5]. For FireFly, we use the flow distribution and data plane translation from S1.5,1.7.

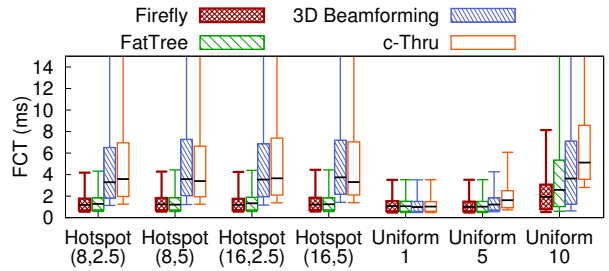
Following prior work, we use synthetic traffic models based on DC measurements [2, 5]. As a baseline, we consider a `Uniform` model where flows between pairs of racks arrive independently with a Poisson arrival-rate λ/s, with an empirically-derived flow size distribution [2]. We use λ as the knob to tune the link saturation level. Based on prior observations, we also consider the `Hotspot` model [2]. Here, in addition to the `Uniform` baseline, a subset of rack pairs have a higher arrival-rate λ_2 and a fixed large flow size of 128MB [5]. For `Uniform` loads, we use the label `Uniform X` where X is average load per server (in Gbps) by choosing suitable λ. For `Hotspot` loads, we use the label `Hotspot(Y,X)` where Y is the % of racks that are hotspots and X is the additional average load on each hotspot server; all `Hotspot` workloads use a baseline `Uniform 5` as background traffic.

Performance Comparison. There are two key metrics here: (1) the average throughput per server, and (2) flow completion time (FCT). For ease of visualization, we do not show error bars over multiple runs, since the results were consistent across multiple runs. We also do not show FireFly-GM (with 40° coverage-angle GMs) results, since they are similar to the default FireFly-SM.

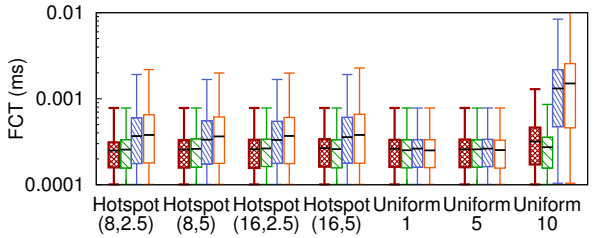
As a starting point, we use `htsim` for a detailed packet-level simulation. We extended `htsim` to support short flows, arbitrary traffic matrices, and route reconfigurations. Due to scaling limitations of `htsim`, even on a high-end server (2.6GHz, 64 GB RAM), we could only scale to a 64-rack DC at our 10 Gbps workloads. Figure 1.8(a) and 1.8(b) show a box-and-whiskers plot of the FCT for long/short flows respectively for a 30 secs run. The result shows that FireFly's performance is close to the full-bisection bandwidth network in both cases. c-Through and 3DB do not perform as well because their augmented network is not sufficient to compensate

for the oversubscription. Thus, their tail performance for long flows suffers. We also see that the FCT for short flows is similar across FireFly and FatTree.

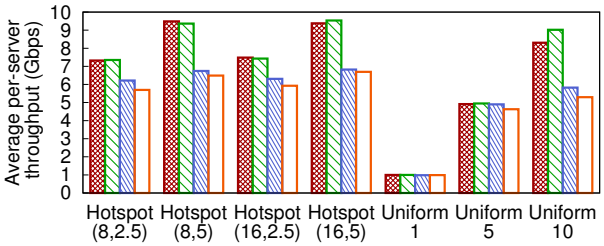
Figure 1.8(c) shows the effective average per-server throughput in the 64-rack setup for different workloads. For the `Uniform` the average is over all servers whereas for `Hotspot` the average is over the hotspot servers. In short, we see that FireFly's performance is close to the full-bisection bandwidth network and $\approx 1.5\times$ better than the augmented architectures.



(a) Flow completion for long flows



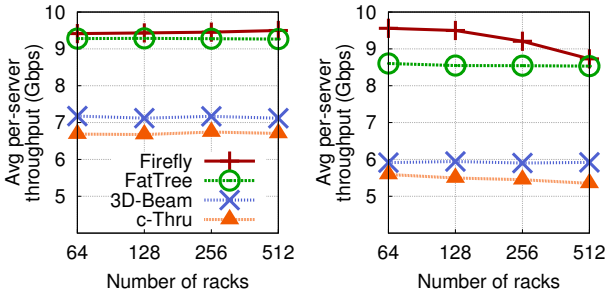
(b) Flow completion for short flows



(c) Average throughput per server (Gbps)

Figure 1.8: Flow completion times (FCT) and average throughput per-server using the `htsim` simulator on a 64-node topology for different workloads

To scale to larger DCs, we use a custom flow-level simulator. We do so after confirming that these simulations roughly match the packet-level simulations. In general, the flow-level simulations overestimates the throughput 5-7% for all architectures since it does not model packet-level effects. Since our goal is to compare the relative performance of these architectures, these simulations are still instructive. Figure 1.9 shows that the earlier performance results continue to hold for the



(a) Hotspot(16,5) (b) Uniform10

Figure 1.9: Scalability evaluation using a flow-level simulator

most saturating workloads even at larger scales. The only drop is at 512 racks for `Uniform10`; here the number of FSOs/rack is slightly sub-optimal as the number of racks grows. We revisit this in S1.8.5.

We also measured the packet latency (number of hops) statistics and found that the average latency were 3.91 (FireFly), 4.81 (FatTree), and 3.9 (3dB, c-Through), while the maximum was 5 for FireFly and 6 for the rest.

1.8.2 Performance during Flux

Because packet- or flow-level simulations do not give us a detailed replay of the events at the FireFly controller and in the network, we use `Mininet` for this evaluation [20]. Due to scaling limitations, we scale down the DC size to 32 racks and the link rates to 10 Mbps, and correspondingly scale the workload down. Since our goal is to understand the relative impact of reconfigurations w.r.t. the steady state behavior, we believe this setup is representative. For the following result, we consider a `HotSpot` workload, with seven distinct reconfigurations as elephant flows arrive.

We poll the virtual switches to obtain link utilization and loss rates and use a per-rack-pair ping script to measure inter-rack latency. We bin these measurements into two logical bins: (a) During reconfigurations and (b) Steady state (i.e., no active reconfiguration). Figure 1.10 shows the distribution link utilization, loss rate, and inter-rack latency for each bin. While there is a small increase in the "tails", the overall distributions are very close. This suggests that the impact on the network during reconfigurations is quite small and that our mechanisms from S1.6 work as expected.

1.8.3 Preconfiguration Efficiency

As before, we use 48 FSOs per rack and 10 SMs per FSO for SM-PCFT, and assume GMs with an coverage-angle of 40° for GM-PCFT. We generate $\approx 15n$ (n is the number of racks) random instances and pick the best. We normalize the estimated

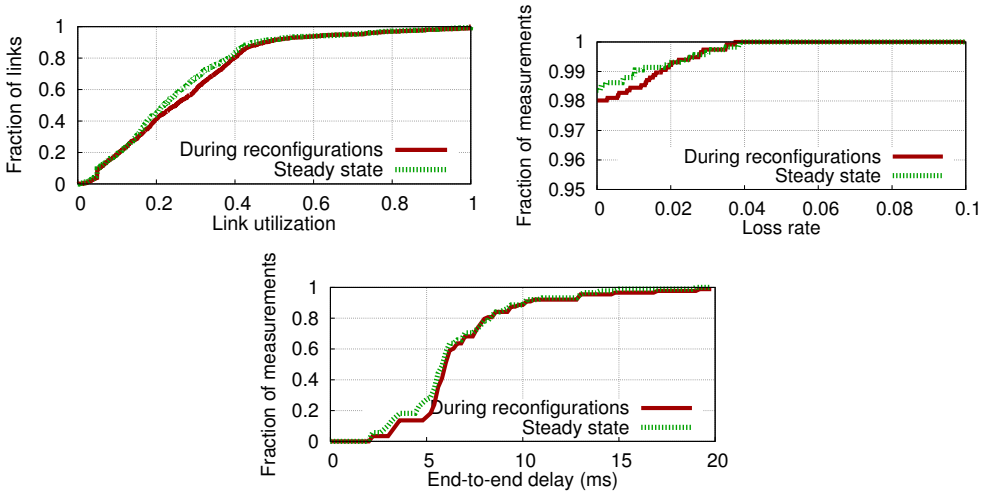


Figure 1.10: Comparing network performance during reconfigurations and in steady state

#Racks	Normalized DBW w.r.t. upper bound	
	SM-PCFT	GM-PCFT
64	0.96	0.84
128	0.93	0.84
256	0.91	0.85
512	0.94	0.88

Table 1.1: Efficiency of the PCFT algorithms

DBW w.r.t an upper bound of $\frac{nm}{2}$.[10] Table 1.1 shows that the SM-PCFT and GM-PCFT solutions achieve $\geq 91\%$ and $\geq 84\%$ of the upper bound across different DC sizes. The lower performance of GM-PCFT is likely because of less randomness due to a block-level construction. (This does not however impact the performance of the runtime topology in practice for the workloads we consider.)

We also evaluate an incremental expansion scenario where we want to retain most existing PCFT as we add new racks similar to Jellyfish [9]. We find that incrementally constructed PCFTs perform nearly identical w.r.t. a PCFT computed from scratch (not shown). We posit that this stems from the incremental expandability of random graphs [9].

1.8.4 Reconfiguration Efficiency

Table 1.2 shows the computation time and optimality gap of the FireFly two-step heuristic from $S1.5.1$. We consider two points of comparison: (a) Full-LP, a LP

[10] Any equi-sized partition of n racks with m FSOs can have at most $nm/2$ active links (one per FSO) in a "cut".

# Racks	Time (ms)			Optimality Gap
	Full-LP	Greedy-LP	FireFly two-step	(%)
32	138	110	27	2.8%
128	4945	208	54	2.6%
256	1.7×10^6	3.3×10^4	60	1.3%
512	6.4×10^8	1.9×10^7	68	2.8%

Table 1.2: Scalability and optimality of the FireFly reconfiguration algorithm.

relaxation of Figure 1.6, which also yields an upper-bound on the optimal, and (b) Greedy-LP which uses greedy topology selection but solves the flow routing LP using `Gurobi`. Our approach is several orders of magnitude faster—Full-LP and Greedy-LP simply do not scale for $\geq$ 32 racks. This is crucial as the FireFly controller may need to periodically reoptimize the network every few seconds. Moreover, the (upper bound) on the optimality gap is $\leq 2.8\%$. Finally, we note that triggered reconfigurations (S1.5.2) incur only 5-10 msec (not shown). Most of the time is actually spent in route computation, which can be run in parallel to allow a high rate of concurrent reconfigurations.

	FSO=24			FSO=36			FSO=48		
$\#SM \rightarrow$	5	10	15	5	10	15	5	10	15
n=256	4.3	8.4	9.1	5.84	9.14	9.24	8.6	9.2	9.32
n=512	3.4	4.3	5.7	3.84	5.96	9.21	4.3	8.7	9.14

(a) Varying # of FSOs/rack and SMs/FSO in SM-based networks

	FSO=24	FSO=36	FSO=48
n=256	8.95	9.12	9.2
n=512	9.01	9.29	9.37

(b) Varying # of FSOs/rack in GM-based networks

Table 1.3: Average throughput per-server on `Uniform10` for varying network parameters.

1.8.5 Sensitivity Analysis

Given that we are projecting cost and form-factors for a new technology, a natural concern is the robustness of our results w.r.t. key parameters: number of FSOs/rack, number of SMs per FSO, and the reconfiguration latency.

Table 1.3 shows the average per-server throughput on a range of FireFly instantiations. The key observation is that performance of GM-based networks degrades minimally as the #FSOs decreases even for a 512-racks network. For SM-based networks, performance at 512-rack degrades drastically vs. #FSOs, while the 256-rack network performs close to full-bisection FatTree even when #FSOs=24. Interestingly, increasing the number of SMs can compensate for decrease in #FSOs; a

Architecture	Equip (M$)		Cable (M$)		Power (M$)		Total	
	Cu	Fiber	Cu	Fiber	Cu	Fiber	Cu	Fiber
FatTree	44	29	7	2	1	2	53	35
3DB	26	17	5	2	1	1	32	21
cThru	26	17	5	2	1	1	32	21
FireFly	20		0		1		20	

Table 1.4: Cost of equipment, power, and cabling assuming 512 racks with 48 servers/rack. Since these are estimates, we round to the nearest million.

512-rack DC with 36 FSOs still performs well with #SMs=15. Overall, this sensitivity analysis suggests that we have a significant amount of leeway in our cost (see below) and form factor estimates, before FireFly's performance suffers.

Finally, with respect to total reconfiguration latency, we observe that varying the latency from 10msec to 50 msec has minimal (¡ 5%) impact on FireFly's performance (not shown). Again, this is a positive result that we can achieve pretty good performance even with unoptimized steering delays and network update times.

1.8.6 Cost Comparison

Table 1.4 summarizes the equipment, power, and cabling cost of different DC architectures. We consider a DC with 512 racks and 48 servers/rack, and compute costs for both copper- and fiber-based realizations for the wired architectures.

We estimate equipment costs based on a per-port 10GbE cost of 200$. Fiber-based architectures (including FireFly) also incur a cost of 100$ per-port for SFPs (10GbE SFP+ ports) [45]. FireFly uses a 96-port (10G) ToR switch on each rack with 48 FSOs, the full-bisection FatTree needs 1536 96-port (10G) switches, while the 1:2 oversubscribed cores of c-Through/3DB use roughly half the ports of FatTree. FireFly has an additional cost for FSO devices, which we estimate to be 200$ per device, including SMs or a GM (see $S1.3$). For 3DB, we assume there are 8 60 GHz radios per rack with each assembly costing 100$. For c-Through, we conservatively assume the 512 optical switch to be 0.5M$. We assume ceiling mirrors (FSO, 3DB) have negligible cost.

For cabling, we assume an average cost of $1 and $3 per meter for copper and optical-fiber respectively, and use an average per-link length of 30m [3]. We estimate the 5-yr energy cost using a rate of 6.5cents/KWHr, per-port power consumption of 3W (fiber) and 6W (copper), ignore the energy cost of SMs, 60GHz radios, and optical switch.

We see that the total cost of FireFly is 40-60% lower than FatTree and is comparable (or better) than the augmented architectures. Furthermore, as the previous

section showed, FireFly's performance will not suffer even if we use only 2/3 of the #FSO devices per rack.

1.9 Discussion

Hybrid SM-GM Architectures. To simplify the discussion, we considered SM and GM designs independently. As future work, we can consider combining the benefits in complementary ways in the PCFT design; e.g., use GMs to target farther racks (as the coverage-cone grows with distance) and SMs to target nearby racks.

Pre- and re-alignment. We envision using external (portable) tools for pre-aligning SMs/GMs, as this will be done infrequently and speed is not critical. (Precomputing the alignment/orientation parameters is a simple geometry computation.) While our FSO design can tolerate minor misalignments (S1.3), long term operation may need occasional alignment corrections. Here, we can use the feedback from the digital optical monitoring support available on optical SFPs for realignment; GMs can directly use such feedback to realign, but SMs may need additional micro-positioning mechanisms (e.g., piezoelectrics).

Beyond 10 Gbps. Current long-range connector standards for 40/100 Gbps (e.g., 40GBASE-LR4 or 100GBASE-LR4) use WDM to multiplex lower rate channels on the same fiber, one in each direction. However, just like the 10 GbE standard that we have used, there are still two optical paths (with two fibers) for duplex links. Single-fiber solutions are not commodity yet at these speeds as the market is still nascent. We expect, however, single-fiber commodity solutions will be available in future at all speeds just like the 1 GbE case [32] we have used. Otherwise, we will need two optical paths for each (duplex) link or develop custom single path solutions.[11]

1.10 Related Work

Static Wired Topologies. Early DCs used tree-like structures, which had poor performance due to oversubscription. This motivated designs that provide full bisection bandwidth [1, 9, 11], which are overprovisioned to handle worst-case patterns. In addition to high cost, such structured networks are not incrementally expandable [9]. In contrast, FireFly is flexible, eliminates cabling costs, and amenable to incremental expansion. Other efforts proposed architectures where servers act

[11] Short range connector standards for 40/100 Gbps use multiple ($\leq$12) fiber strands rather than WDM. We do not know if our optical design can extend to such multi-strand connectors.

as relay nodes (e.g., [46]). However, they are not cost competitive [3] and raise concerns about isolation and CPU usage.

Optical Architectures. High traffic volumes coupled with the power use of copper-based Ethernet, has motivated the use of optical links. Early works such as c-Through [7] and Helios [21] suggested hybrid electric/optical switch architectures, while recent efforts have considered all-optical designs [4, 47]. The use of free-space optics in FireFly avoids the cabling complexity that such optical designs will also incur. Furthermore, by using multiple FSOs per rack, FireFly can create richer topologies (at the rack level) than simple matchings [4, 7, 21, 47]. Moreover, FireFly doesn't need optical switching, thus eliminating concerns about cost/scalability. Finally, optical switching can disconnect substantial portions of the optical network during reconfiguration. While FireFly also has transient link "off" periods, these are localized enabling us to avoid black holes and disconnections using simpler data plane strategies (S1.6).

Wireless in DCs. The FireFly vision is also inspired by Flyways [6] and 3D-Beamforming [5]. However, RF wireless technology suffers from high interference and range limitations and limits performance. The use of free-space (wireless) optics in FireFly eliminates interference concerns. Shin et al., consider a static all-wireless (not only inter-rack) DC architecture using 60 Ghz links [22]. However, this requires radical restructuring of DC layout and has poor bisection bandwidth due to interference.

Consistency during Reconfigurations. Recent work identified the issue of consistency during network updates [42, 43]. FireFly introduces unique challenges because the topology changes as well. While these techniques can also apply to FireFly, they are more heavyweight for the specific properties—no black holes, connectivity, and bounded packet latency—we are interested in. Thus, we can devise simpler domain-specific solutions. Other work focuses on avoiding link congestion [48] during updates. While FireFly's mechanisms do not explicitly address congestion, our results (S1.8.2) suggest that this impact is quite small.

Chapter 2

Data Preservation Under Spatial Failures in Sensor Networks

2.1 Introduction

In this chapter, we address the problem of data preservation in a wireless sensor network after node failures. We focus on smart dust type networks [49] where the network consists of a large number of cheap unreliable nodes. This design philosophy contrasts with the traditional 'centralized' sensing mechanism in which a small number of powerful sensing stations were used (e.g., the weather stations). Here we use the sheer number of sensors for both wide area coverage and high resolution data collection. Having a large number of nodes also increases the system redundancy and robustness to failures. Since nodes are cheap and unreliable, they are likely to fail for many reasons. Nevertheless we have prepared redundant nodes in the proximity to take over both the sensing tasks and the data that has been generated.

Sensor nodes may fail to operate for many reasons. Since the nodes are inexpensive (thus crappy), they may suddenly stop functioning for no reason. The nodes may also be destroyed by animals, humans, or natural disasters (earthquake, fire, river overflow, etc). They may be destroyed by adversarial attacks (a bomb explosion for example). The nodes may also be temporarily disabled by jamming, traffic congestion, or energy depletion. In case of such unfortunate events we can revoke the replacement sensors to take over the sensing tasks. However the data stored on the nodes that have been destroyed is lost unless we design data storage schemes resilient to node failures, which is the topic of this chapter.

We focus in the chapter on node failures with some spatial patterns, which are arguably the most common type. There is often strong spatial correlation among the failed nodes. The events that destroyed one node may very likely influence a nearby node and destroy it as well. We model such spatial failure patterns by some explicit geometric shapes, where all the nodes contained in the shape fail at the same time. The location and orientation of the shape could be variable or known depending on the scenarios. An example could be a bomb attack with a circular shape and variable location (it can happen anywhere in the network) or break of a dam with fix location and orientation (the location of the dam and area affected are known).

We assume that there are k nodes within the network generating data of interest and some additional n nodes that can be used for extra storage and relay for communication. To preserve data generated in the network, we necessarily need to introduce sufficient redundancy by storing the data at some other node. Due to the small form factor and cheap costs, individual nodes memory cannot scale to the size of the network. More specifically, we assume over time the data generated by one data node would eventually occupy almost all of an individual node's memory. The nodes also have severe communication, computation and memory limitations. Thus our algorithm will try to use as little communication as possible.

Contributions. We address the problem of introducing sufficient redundancy with minimal communication cost to a network such that the entire network data can be retrieved after a failure. We use two approaches to introduce data redundancy in the network.

- **Replication:** Each data packet is copied into multiple storage nodes in the network. The retrieval algorithm is simply pulling the data out of the storage nodes that contains a copy.

- **Erasure codes:** An erasure code of multiple data packets would be computed and stored in the storage node. In particular, we use random linear codes as in [50]. That is, each codeword is a linear combination of the original data (called symbols) with random coefficients. The retrieval process consists of one sensor node locally pulling relevant data from the network so as to solve a system of linear equations to decode the data.

Each method has its own advantages and disadvantages. Generally speaking, data replication allows for straight-forward data recovery from a surviving node holding the data. Using erasure codes, we can potentially use the limited storage nodes in a more efficient and effective manner, since a storage node can possibly hold information helpful for multiple data nodes. The downside is that data recovery requires the decoding cost of solving a linear system of equations.

Using any of the two approaches, our objective is to minimize the amount of data transmissions for introducing redundancy. We prove that such an optimization problem is NP-hard using any of the two approaches. Therefore, we propose $O(\alpha)$ approximation algorithms, where α is the fatness' of the given potential spatial node failures.

chapter Organization. The rest of the chapter is organized as follows. In Section 2.2, we present our network and cost models, and give an exact formulation of the problems. In Section 2.3, we discuss the problem with replication as the choice method of generating redundancy. In Section 2.4, we discuss the problem using erasure codes. We present our simulation results in Section 2.5. We defer some of the tedious proofs of our results to Section 2.6.

2.2 Problem Formulation and Related Work

In this section, we start by describing our network model. We then give the formal formulation of the problem using each of the redundancy schemes and then, we discuss the related work done in this area.

Network Model. Consider a network of sensor nodes deployed in a plane. In our network models, each node is either a data node or a storage node, but never both. Each data node generates data of interest, while the storage nodes can be used for storage of replicated data. We assume that over time each data node generates data whose size is almost the size of an individual node memory and each data node stores a copy of its own data. Thus, a data node cannot be used for storage of other nodes' data. Throughout the article, we assume that the total number of storage nodes is more than the total number of data nodes in the network. Also, each node is aware of its own location and the coordinates of its neighbors relative to itself.

The major focus of the chapter is to design a redundant data storage scheme with minimal communication cost such that the network data can be recovered after node failures with spatial patterns. Below, we formally define our model of communication cost and node failure patterns.

Communication Graph and Costs. We use r to denote the uniform transmission radius of the sensor nodes, and two nodes can communicate directly with each other if the Euclidean distance between them is less than r. The communication graph of the network is defined over the set of all nodes as vertices and has an undirected edge between two nodes i and j if they can communication directly with each other. The communication distance between two nodes i and j is the distance between i and j in the communication graph.

Definition 1 (Communication Cost.) Let d be a data node and S be a set of storage nodes. We use $C(d, S)$ to denote the cost of transmitting one packet of data from d to all the nodes in S. For simplicity, we assume $C(d, S)$ to be equal to the size of S plus the communication distance between d and the nearest destination in S. □

The above cost model is reasonable if S is clustered in a region and we use Geocast [51] like technique to broadcast a message to S. We note that the above communication cost model is only for the sake of simplicity of presentation, and the results of this chapter hold even for the most general cost model wherein $C(d, S)$ is the size of the minimum Steiner tree over $d \cup S$ in the communication graph.

Failure Pattern (the geometric definition): A failure pattern on the network is a closed 2-dimensional geometric shape. Both location and orientation of the failure in the plane can be known or unknown. When a failure occurs, the location and orientation of the failure pattern is determined and all the nodes contained in the failure area simultaneously fail (are destroyed).

As an example, a bomb attack can be defined as a failure pattern with circular shape with variable location, since it can happen anywhere in the network domain. Destruction of a dam on the other hand can be defined as a stripe shape failure which can only happen in a predefined fix location.

An important observation is that a spatial failure with fixed location and orientation always destroys a fixed subset of nodes. Furthermore for every given subset of nodes a closed shape can be found to enclose only the nodes in the subset. Following these observations we can present a failure as a subset of nodes it destroys rather than its shape. This gives a combinatorial definition of failures.

Failure Patterns (the combinatorial definition): A failure pattern can be defined as a subset of nodes in the network.

We assume that no two failures happen simultaneously, i.e. after a failure there is sufficient time for the network to reorganize. We also assume that the cost of data recovery is not a concern. Once a failure is sensed a mobile central station with unlimited resources can do the recovery.

Next we present two problem formulations, for the two methods of introducing redundancy.

2.2.1 Redundancy with Replication

In this subsection, we give the formal definition of the problem when, to allow recovery from failures, we simply replicate data at other storage nodes. We start with defining storing set as follows.

Definition 2 (Storing Set.) For a given data node d, a storing set is a set $S(d)$ of storage node such that no failure destroys the data node and all the nodes in the set $S(d)$. □

Note that in our model the size of data is almost the size of the available storage of a node, and thus, we cannot store more than one data packets in each node. Therefore, the storing sets for different data nodes should be disjoint. We can now formally define the problem as follows.

Minimum Cost Data Replication (MCDR) Problem. Given a network with data and storage nodes and a set of failure patterns, find a set of disjoint storing sets, one for each data node, such that the sum of communication costs from a data node to its storing sets is minimized. Formally, if D is the set of data nodes, then for each $d \in D$, we find a storing set $S(d) \subseteq S$ such that $\sum_{d \in D} C(d, S(d))$ is minimum.

Theorem 1 *The MCDR problem is NP-hard. Moreover, it is also NP-hard to approximate the MCDR problem within any finite approximation ratio.* ■

We defer the proof of the above theorem to Section 2.6.

2.2.2 Redundancy with Erasure Codes

The second scheme to introduce redundancy is to use decentralized erasure codes as introduced in [50]. In this scheme, each of the data nodes pre-routes its packets to specific storage nodes. Each storage node creates and stores a codeword of all the data packets it receives. The codeword is constructed by cascading linear combination of chunks of input data packets and in order to decode the data one should pull the network data and solve a system of linear equations. The details of coding and decoding procedures for decentralized random linear codes can be found in [52].

One can represent this erasure coding scheme by a bipartite graph between data nodes and storage nodes such that there is an edge between a data node d and a storage node s if d pre-routes its packet to s. Since each data node stores a copy of its own data, in the recovery phase, the data of surviving data nodes can be eliminated from the codewords of surviving storage nodes. Hence the necessary condition for recovering data after a failure is to recover the data of destroyed data nodes from the surviving storage nodes. As shown in [50], the necessary condition for successful recovery is the existence of a maximal matching between destroyed data nodes and the surviving storage nodes [53, 54]. We can now formulate our problem as follows.

Minimum Cost Data Coding (MCDC) Problem. Given a network with D and S as the set of data and storage nodes respectively and a set of failure patterns, the MCDC problem is to construct a bipartite graph $G'(D \cup S, E')$ with minimum sum of edge weights where:

- The weight of an edge (s, d) in E' is the cost of communication between s and d, and
- The set of edges E' is such that for any given failure pattern F, the induced subgraph in G' over $(D \cap F) \cup (S - F)$ has a matching of size $|D \cap F|$. Here, we have used F to denote the set of nodes destroyed by F.

Note that for a failure F, $(D \cap F)$ is the set of destroyed data nodes and $(S - F)$ is the set of surviving storage nodes. Thus, the above condition ensures that there is a matching of size equal to the number of destroyed data nodes between the set of destroyed data nodes and the surviving storage nodes.

Theorem 2 *The MCDC problem is NP-hard.* ■

We defer the proof of the above theorem to Section 2.6.

2.2.3 Related Work

Increasing data persistence in the presence of node failure has been a subject of increasing research in recent years. One popular approach to this problem is to use network coding. The usefulness of network coding for data storage was investigated in [55] where authors show a simple distributed scheme using network coding can perform as well as the case when there is complete coordination between nodes.

Dimakis et. al. in [50, 52, 56] and Lin et. al. in [57, 58] purposed two different schemes for decentralized implementation of Fountain codes in wireless sensor networks. Both algorithms use limited global information such as the total number of nodes and sources. Aly et.al in [59] use simple random walks to implement a decentralized Fountain code without presence of any global information. Kamra et. al. in [60] purposed a different approach with the goal of enhancing data persistence in network in case of node failure. It uses growth codes to maximize the amount of information available for decoding at any chosen moment.

All of the previous research on this topic use a probabilistic node failure model without any spatial pattern for node failure. To the best of our knowledge, this is the first word addressing the problem of preserving the data in case of spatial attacks/failure on a wireless sensor network. Although any of previous methods can be used for designing failure tolerant network, the present of spatial pattern for node failure allows for reducing the required redundancy and communication significantly.

2.3 Approximation Algorithm for the MCDR Problem

In this section, we design an algorithm for the MCDR problem which yields a solution with a near-optimal cost on an average, for uniformly random networks.

Survival Matching Algorithm (SMA). Note that in our MCDR problem if we restrict the size of storing sets to be just one storage node, then the MCDR problem can be easily reduced to the minimum-cost 2D-matching problem (in bipartite graphs). For the unrestricted MCDR problem, our approximation algorithm called the Survival Matching Algorithm (SMA) essentially solves the restricted MCDR problem optimally (i.e., restricts the storing sets to just one storage node and finds the minimum-cost 2D-matching in an appropriately defined bipartite graph). We will later show that SMA's solution is within a constant factor of the unrestricted optimal solution for uniformly random networks.

SMA Description. Given a network of data and storage nodes, SMA starts with constructing a bipartite graph $G_s(D \cup S, E_s)$ between data and storage nodes wherein

Algorithm 1 Survival Matching Algorithm (SMA)

- Create a weighted bipartite graph $G_s = (D \cup S, E_s)$ over the set of data nodes D and storage nodes D. An edge (d, s) is in E_s if and only if there is no failure set that contains both d and s. The weight on each edge $(d, s) \in E_s$ is equal to the communication cost between d and s.
- Find the minimum-weighted perfect matching M in G_s.
- For each edge $(d, s) \in M$, create the storing set $\{s\}$ for the data node d.

there is an edge between a data node d and a storage node s if and only if d and s do not exist together in any particular failure set. Essentially, an edge $(d, s) \in E_s$ signifies that $\{s\}$ can be picked as a storing set for the data node d. Thus, a perfect matching in G_s yields a set of disjoint storing sets of size one each, and hence is a solution (not necessarily optimal) to the MCDR problem. In addition, each edge (d, s) in G_s is assigned a weight equal to the communication cost between d and s, and we actually find a minimum-cost perfect matching in G_s. Note that minimum-cost perfect matching problem can be solved in polynomial time [61]. See Algorithm 1 for a formal description of SMA.

Approximation Proof. We now show that in uniformly random networks with spatial failures, SMA delivers a solution whose cost is at most α times the optimal cost with high probability. Here, α is the fatness (as defined below) of the given set of spatial failures.

Fatness of Spatial Failures. Fatness is a well-known concept in geometry which quantifies how a geometric object is spread in all directions [62]. For a given object O, its fatness is defined as follows. Let C and C' be two concentric circles such that C fully contains O and C' is fully contained in O. Then, the fatness of O is defined as the maximum possible value for the ratio $\frac{Radius(C)}{Radius(C')}$ over all possible such concentric circles C and C'. In this work, we extend the concept of fatness to a set of geometric objects (spatial failures) as below.

Definition 3 (Fatness) For a given set of spatial failures F, let (i) R' be the radius of the largest circle that can be contained in each of the failures, and (ii) R be the radius of smallest circle that can contain each of the failures. The fatness α of the set of spatial failures F is defined as the ratio R/R'. □

Proof Outline. We compute the approximation ratio of SMA by estimating (i) a lower bound on the optimal cost, and (ii) the expected cost of the SMA solution, in the below two lemmas respectively.

Lemma 1 *For any instance of the MCDR problem, the optimal cost of a solution is at least $(|D|R')/(2r)$ where R' (as defined above) is the radius of the largest circle that can be contained in each of the failures, $|D|$ is the number of data nodes, and r is the transmission radius of the nodes.* ■

PROOF: Consider a storing set and two (storage) nodes s_1 and s_2 in it that are farthest from each other. Distance between s_1 and s_2 must be at least R', since otherwise all the nodes in the storing set can be contained in a circle of radius R' and thus in any given spatial failure with appropriately chosen location and orientation (which contradicts the definition of a storing set). Now, the minimum communication cost between a data node to (s_1, s_2) is at least half the communication cost between s_1 and s_2 which is at least R'/r. Thus, the total cost of a set of $|D|$ storing sets is at least $(|D|R')/(2r)$. ■

The proof of the below lemma is rather tedious, and is deferred to Section 2.6.

Lemma 2 *Consider a uniformly random network, i.e., with uniformly, independently, and randomly distributed data and storage nodes, in a square region of size $q \times q$. Lets given failure patterns be such that $\widehat{R} \leq q/2$, where $\widehat{R}$ (as defined in Definition 3) is the radius of the smallest circle that can contain each of the given failure patterns.*

In such networks, SMA delivers a valid solution with a very high probability (converges to 1 for large networks). In addition, the expected cost of the delivered solution is $O(|D|R/r)$, where $|D|$ is the total number of data nodes, and r is the transmission radius of the nodes.

■

From the above two lemmas, we get the following theorem.

Theorem 3 *For uniformly random networks in a square region of $q \times q$, where given failure patterns are such that $\widehat{R} \leq q/2$, SMA delivers a valid solution with a very high probability and the expected approximation ratio (over all random networks for which SMA delivers a valid solution) is $O(\alpha)$ where α is the fatness of the given failure patterns.*

■

Significance of Theorem 3. Note that if we allow arbitrarily large failures, then no valid solution may not even exist. Further, since the decision version of MCDR problem is NP-complete (as shown in Section 2.6.3), it is unlikely that a polynomial algorithm can always return a valid solution if one exist. Thus, an algorithm returning a solution with high probability is the best we can hope for.

Finally, since it is NP-hard to approximate the MCDR problem in general (see Theorem 1), the above average approximation-ratio over almost all random networks is of significance.

Distributed Algorithm. To the best of our knowledge, there are no efficient distributed approximation algorithm known for the minimum-cost matching problem, which is a special case of our MCDR problem. We plan to address this direction in our future work.

2.4 Approximation Algorithms for the MCDC Problem

In this section, we store erasure codes of data packets to generate data redundancy. We start by showing that SMA of the previous section can be used to also solve the MCDC problem with an approximation ratio for random networks. We also design a distributed approximation algorithm, and prove its performance guarantees.

Using SMA for the MCDC Problem. Here, we show that SMA for the MCDR problem can also be used to yield an approximation solution for the MCDC problem in random networks. Firstly, note that the output of SMA, viz., a set of disjoint storing sets, can be used to construct a valid solution G' for the MCDC problem by connecting each data node to each storage node in its storing set. Similar to the arguments in Lemma 1, we can show that the optimal cost of any instance of an MCDC problem is at least $|D|R'/2r$. Thus, by Lemma 2, which also applies to the MCDC solution yielded by SMA, we have the following approximation result.

Theorem 4 *For uniformly random networks, the MCDC solution delivered by SMA (as described above) has an average approximation ratio of $O(\alpha)$, where α is the fatness of the given spatial failures.*

■

Distributed Storage Algorithm (DSA). The main advantage of storing linear combination of data (as in the MCDC problem) over simple replication (as in the MCDR problem) is that the decisions of where to store the data can be made locally by the data nodes, i.e. the data nodes don't have to globally compete for exclusive use of storage nodes. However, the drawback of this linear combination approach is that recovery of data cannot be guaranteed for all failures, since in case of some failure the relevant remaining linear equations may not yield a full rank system. Below, we present a distributed algorithm that guarantees recovery of data with a high probability in random networks.

DSA Description. Consider a uniformly random network of data nodes and storage nodes. Without loss of generality, we assume the density of the network to be one unit. For each data node, we define its **storage region** to be the rectangle of size $2 \times \phi$ at a vertical distance of $2R$ below, as shown in Figure 2.1. Here, R is the radius of the smallest circle that can contain all failures, and ϕ is a constant (see Equation 2.1) which depends on desired probability of recovery and ratio of number of data to storage nodes and is defined later (see Equation 2.1). More formally, if (x, y) are the coordinates of the data node, then its storage region is a rectangle with the left-most top coordinate equal to $(x - 1, y - 2R)$ and has a length of 2 and a height of ϕ.

The bipartite graph $G'(D \cup S, E')$ returned by DSA consists of edges that connect a data node to each storage node in its storage region. The implementation of DSA entails each data node broadcasting its data to all the storage nodes in its storage region, and each storage node stores a linear combination of all the data packets received from various data nodes.

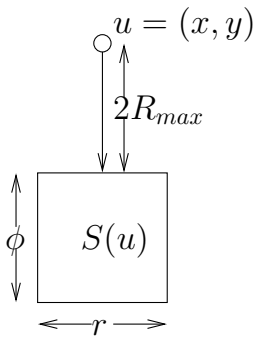


Figure 2.1: Storage region of a data node in DSA.

Theorem 5 *Given a uniformly random network and a set of failure patterns, the solution returned by DSA allows successful recovery of data with a probability of $(1 - \varepsilon)$, when there is a single failure, if the value of ϕ is chosen as:*

$$\phi = \frac{1}{n}\mathbf{max}((n + k)c^2 + (k - n)R, c^2(n + k)), \tag{2.1}$$

where k and n are the number of data and storage nodes respectively, R is the radius of the smallest circle that can contain each failure, and c is the smallest real number such that $g(c) > (1 - \varepsilon)$ where $g(x)$ is the Gaussian error function. Note that when $k < n$, the above equation simplifies to $\phi = c^2(n + k)/n$.

■

The proof for the above theorem is quite involved and hence, is deferred to Section 2.6. We now prove the average approximation ratio of DSA.

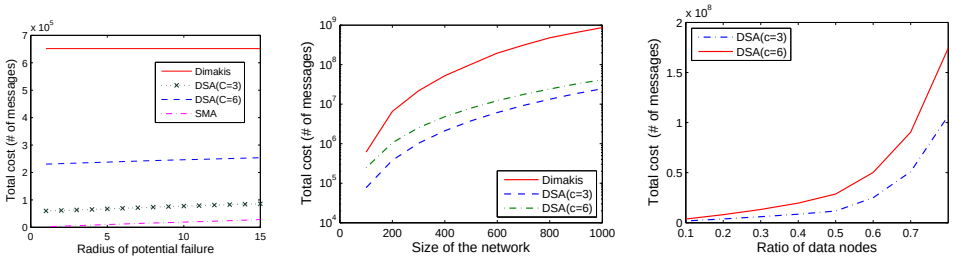


Figure 2.2: The communication cost of various algorithms for increasing (a) potential failure sizes, (b) network size, and (c) ratio of data nodes to storage nodes.

Theorem 6 *For uniformly random networks, DSA returns a solution with an expected approximation-ratio of* $O(\alpha)$. ■

PROOF: Using arguments similar to Lemma 1, we can show that the minimum cost for a solution to MCDC is $(|D|R')/(2r)$. Below, we show that the expected cost of DSA's solution is $O(|D|R/r)$ which will prove the theorem.

In DSA, a data node broadcasts its data to a rectangular region of size $r\phi$ located at a vertical distance of $2R$. Since our choice of ϕ value is $O(1)$ (see Equation 2.1) when number of data nodes is less than the number of storage nodes, the communication cost incurred by each data node is $O(2R/r) = O(R/r)$. Thus, the total expected cost of the DSA solution is $O(|D|R)$. ■

2.5 Simulations

In this section, we provide simulation results of our algorithm. We compare our algorithms with Dimakis' algorithm [56] in terms of the total communication cost under spatial failures with different radii in networks of different sizes. We show that our algorithm incurs much less communication cost than Dimakis' algorithm, and achieves very similar successful recovery probability.

Comparison of Communication Costs. Figure 2.2(a) compares the cost of SMA and DSA to the decentralized erasure codes as used in [56]. Our experiment is performed on a network with $10,000$ sensor nodes, 20% of them are data nodes. All nodes are uniformly distributed in a 100×100 rectangle area. The communicate radius is 2.5. The X-axis is the radius of circular potential failure, varying from 5 to 15. The Y-axis is the total communication cost. For the algorithm in [56], each data node sends its data to exactly $\log k$ storage nodes. Since the algorithm in [56] does not adjust to different failure size, the curve of [56] on the figure is a line. From the figure, we can find SMA and DSA need much fewer messages than the algorithm in [56]. And the recovery probability is very similar, the theoretical recovery probabilities are all more than 99.99%. In our simulations, all three algorithms can successfully recover data.

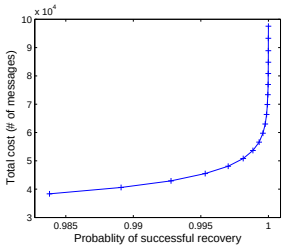


Figure 2.3: The total communication cost for different desired probability of successful recovery.

Fig 2.2(b) is the comparison of total communication costs when the network size is increasing. The X-axis is the network size. In particular, x^2 nodes are uniformly randomly distributed in an $x \times x$ area. x varies from 100 to 1000. For each network, 20% nodes are data nodes, the communication radius is 2.5, and the potential failure radius $0.1x$. The Y-axis is the total message costs, on a log scale. We can find, in every size of the network, our DSA algorithm is one order of magnitude better.

We are also concerned about the communication costs for networks with different fractions of data nodes. Fig 2.2(c) shows the result. This time we fix the network size. We have $250,000$ nodes uniformly randomly distributed in a 500×500 area, but the ratio of data nodes changes. The X-axis is the ratio of the data nodes, varying from 10% to 80%. The communication radius is 2.5, and the potential failure size is 50. The Y-axis shows the total cost of messages to distribute data. We can find that when the ratio of data nodes increases, the cost is increasing fast. Especially when the radio is bigger than 0.5, which means there are more data nodes than storage nodes. However, for the type of network we are considering there are enough number of redundant (i.e., storage) nodes, so this situation is rare.

Probability of Successful Recovery for DSA. For DSA, the size of the storage rectangle would affect the probability of successful recovery. To achieve higher probability, we need a larger storage rectangle, which means higher cost during the data distributing phase. Fig 2.3 shows the relationship between the communication cost and the probability of successful recovery. X-axis is the recovery probability, varying from 0.9837 to $1 - 7 \times 10^{-7}$. Y-axis is the total communication cost. All simulations are ran on a sensor network with 10,000 nodes uniformly randomly distributed in a 100×100 area. 20% nodes are data nodes, and the potential failure size is 10, and communication radius is 2.5. Our result shows that the communication cost grows slowly if we keep the success probability to be lower than 0.9998.

In the last two sets of simulations, we did not include SMA algorithm. SMA is a centralized algorithm with running time $O(n^3)$. It is computationally too heavy for the large network that we are testing.

2.6 Proofs

In this section, we present proofs of Lemma 2, Theorem 5, and the NP-hardness results in the three subsections.

2.6.1 Proof of Lemma 2

Lemma 2: Consider a uniformly random network, i.e., with uniformly, independently, and randomly distributed data and storage nodes, in a square region of size $q \times q$. Lets given failure patterns be such that $\widehat{R} \leq q/2$, where $\widehat{R}$ (as defined in Definition 3) is the radius of the smallest circle that can contain each of the given failure patterns.

In such networks, SMA delivers a valid solution with a very high probability (converges to 1 for large networks). In addition, the expected cost of the delivered solution is $O(|D|R/r)$, where $|D|$ is the total number of data nodes, and r is the transmission radius of the nodes.

Proof of Lemma 2. Here, we prove only the second claim, i.e., the expected cost of the SMA solution is $O(|D|R/r)$. The first claim that SMA delivers a valid solution with high probability is observed in Corollary 1 later in this subsection.

Recall that the MCDR problem is to find disjoint storing sets of arbitrary size. In contrast, SMA finds singleton storing sets (i.e., a maximal matching) which it can do it optimally since min-cost matching problem can be solved in polynomial time. Thus, in order to bound the cost of the SMA solution, it is sufficient to show that in uniformly random networks there exists a matching cost $O(|D|R/r)$ with high probability.

We show the above by introducing an algorithm (CMA) that finds a matching of expected cost $O(|D|\widehat{R})$ with probability that converge to 1 for large networks. Note that unlike SMA, CMA doesn't always find a matching when a matching exist. However, the number of such instances converge to 0 as the size of network increase and the cost of those instances is not infinte(bound by the size of the network), hence we can claim that the expected cost of SMA is the same as expected cost of CMA.

Cell Matching Algorithm (CMA). CMA works by creating a grid in the network with initial unit square cells. Then at each if there exist unmatched data nodes CMA try to find a matching for it in the cell assigned to it. At each stage the size of the cells and cost of matching increase however the number of nodes needs to be matched decrease with a faster rate, which result in a expected cost of $O(k.\widehat{R})$.

CMA starts by dividing the network region into cells of unit square size, and then repeats the following steps until all the data nodes have been paired/matched to some storage node.

- If all the data nodes of a cell are already matched (in an earlier step), mark the

cell as `complete`; remaining cells are considered incomplete at this stage. Note that in the initial stage, all cells are incomplete.

- For each incomplete cell, we assign the cell at vertical distance of $2R$ below to be its "storage cell".
- For each yet-unmatched data node in an incomplete cell, try to find an unmatched storage node in the storage cell of its cell.
- Merge pairs of all (complete as well as incomplete) horizontally-adjoining cells to construct new cells of double the width (but the height remains unit).

In the end, when each cell has become a full row, we match the remaining data nodes to the remaining unmatched (possibly, very far away) storage nodes in the network.

See Figure 2.4 for a brief illustration of CMA. In the first step, cell 3 is the storage cell of cell 1. In the second step, cells 1 and 2 are merged to form a cell(1,2). Similarly, cells 3 and 4 are merged to get a cell (3,4). In the next step, the storage cell of $(1, 2)$ is the new cell $(3, 4)$.

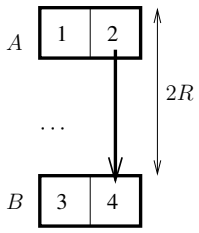


Figure 2.4: Cell Matching Algorithm.

Estimating the Cost of CMA Matching. To estimate the cost of the matching delivered by CMA, let us first compute an upper bound on the cost of matching nodes in each cell at each step. At the i^{th} step (for the first step, we use $i = 0$), we have the following.

- The expected number of nodes in each cell is 2^i, since the size of a cell is $2^i \times 1$ and we assume unit density.
- The maximum communication distance between two nodes that can be matched is $O\left((1/r)\sqrt{R^2 + 2^{2i}}\right) = O(2^i R/r)$.

To bound the cost incurred in matching/pairing nodes in the i^{th} step, let us assume the worst case scenario that each node in each incomplete cell actually gets matched. In such a case, an incomplete cell incurs a maximum cost of $O(2^{2i}R)$ in the i^{th} step.

In the very last stage (after each cell is a full row), all the yet-unmatched nodes are matched wherever possible. But, since at this stage, 2^i is equal to length of the network, the cost is still bounded by $O(2^{2i}R)$.

Finally, we show in Lemma 3 that the expected number of incomplete cells at each step i is ((Number of Cells) $\times O(e^{(-2^{(i-1)/2})})$). Now, since the number of cells containing a data node is bounded by $|D|$ at each step, we get the overall cost of CMA solution as:

$$\sum_i |D|O(e^{(-2^{(i-1)/2})})O(2^{2i}R/r) = O(|D|R/r).$$

Lemma 3 *In CMA (as described in the above lemma), the probability of any particular cell being incomplete in the i^{th} step is $O(e^{(-2^{(i-1)/2})})$).* ■

PROOF: For a cell to be incomplete in step i, it should contain an unmatched data node. This means that the number of data nodes in at least one of its two constructing cell at step $i-1$ should be more than the number of the storage node in its storage area. For step i, let $U-i$ to be the difference between data nodes of a cell and storage nodes of its storing cell. The probability of a cell being incomplete at step i is at most twice the probability of $U_{i-1} < 0$. Below we compute the probability distribution function (pdf) of U_i and show that it decrease with a super exponential rate as i increases. To be more precise, we show that the probability of $U_i < 0$ is $O(e^{(-2^{i/2})})$ which means the probability of a cell being incomplete at step i is $O(e^{(-2^{(i-1)/2})})$.

Computing pdf of U_i. For an arbitrary rectangle X over a uniformly random network, both the number of data nodes and number of storage nodes in X are random variables (denoted as D_X and S_X respectively). By computing the pdf of D_X and S_X we can compute the pdf of the number of data and storage nodes (denoted as D_i and S_i respectively) in a cell at step i. Since we have $U_i = S_i - D_i$ we can use pdf's of D_i and S_i to compute the pdf of U_i.

As mentioned before, for sake of simplicity, we assume the network density to be one unit. We use n and k to denote the total number of data and storage nodes in the network. Since the density of the network is 1, the total area of the network is $n+k$.

Computing distribution of D_X and S_X. Recall that for a region/set X, D_X and S_X are the number of data and storage nodes respectively in X. For a randomly selected rectangle X of length l and height w, D_X and S_X are random variables with a binomial distribution of number of experiments equal to k and n respectively and a

success probability of $lw/(n+k)$.[1] Since both n and k are large numbers, we can use the normal approximation [63] for the above binomial distributions. Thus, we have

$$D_X \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{klw}{(n+k)}, \\ \sigma^2 = \frac{klw}{(n+k)}\left(1 - \frac{lw}{(n+k)}\right)\end{array}\right)$$
$$S_X \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{nlw}{(n+k)}, \\ \sigma^2 = \frac{nlw}{(n+k)}\left(1 - \frac{lw}{(n+k)}\right)\end{array}\right)$$

where μ is the mean and σ is the variance of the distribution.

Probability distribution of U_i. In step i of our algorithm, each cell has a size of $(l = 1, w = 2^i)$. We have,

$$D_i \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{k2^i}{(n+k)}, \\ \sigma^2 = \frac{k2^i}{(n+k)}\left(1 - \frac{2^i}{(n+k)}\right)\end{array}\right)$$
$$S_i \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{n2^i}{(n+k)}, \\ \sigma^2 = \frac{n2^i}{(n+k)}\left(1 - \frac{2^i}{(n+k)}\right)\end{array}\right)$$

And the probability distribution of $U_i = S_i - D_i$ is

$$U_i \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{(n-k)2^i}{(n+k)}, \\ \sigma^2 = 2^i\left(1 - \frac{2^i}{(n+k)}\right)\end{array}\right)$$

Computing Pr($U_i < 0$). Since we want to compute an upper-bound for the probability of $U_i < 0$ we can replace the variance with a higher value. This allow us to omit the term $1 - \frac{2^i}{(n+k)}$. We get,

$$U_i \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{(n-k)2^i}{(n+k)}, \\ \sigma^2 = 2^i\end{array}\right)$$

[1]The probability of a particular node i to be in X is equal to (Area of X)/(Total Area of the Network). The number of data nodes in X is the repeat of this single trial k times.

Base on normal distribution properties [63] the probability of $U_i < 0$ is $O(1 - g(c_i))$ where $g(x)$ is the Gaussian error function and

$$c_i = \frac{\mu}{\sigma} = \frac{n-k}{n+k} 2^{i/2}$$

We have $(1 - g(c_i)) < e^{-{c_i}^2}$ [63], Hence the probability of $U_i < 0$ is $O(e^{-{c_i}^2})$. Notice that at different stages of CMA, n and k are constants and, $\frac{n-k}{n+k} > 0$. Hence, the probability of $U_i < 0$ is $O(e^{-2^{i/2}})$.

■

Lemma 4 *CMA (as described above) finds a matching with a very high probability, which converges to 1 for large networks.* ■

PROOF: For a network of size $q \times q$, CMA performs $\log q$ steps. By Lemma 3, the probability of a cell containing unmatched data nodes at the end of the final step is $O(e^{-2^{lgq/2}}) = O(e^{-\sqrt{q}})$. Since the total number of cells at the final step is q, the probability of existence of an incomplete cell at the end of the final step is at most $O(q.e^{-\sqrt{q}})$ which converges to 0 as q goes to infinity. Thus, the probability of CMA not finding a matching converges to zero for large networks. ■

Corollary 1 *In uniformly random network in a square region of size $q \times q$, where given failure patterns are such that $\widehat{R} \leq q/2$, SMA delivers a valid solution with a very high probability (converges to 1 for large networks).*

2.6.2 Proof of Theorem 5

In this subsection, we present proof of Theorem 5. We use the following two basic notations throughout this subsection. Let $G'(D \cup S, E')$ be the solution returned by DSA for a network with D and S as the set of data and storage nodes.

- We use D_X and S_X to denote the set of data and storage nodes respectively in X, where X is a geographic region in the network or a set of network nodes.
- For a set of data nodes δ, we use $N(\delta)$ to denote the set of storage nodes that are connected by an edge in G' to some data node in δ. In other words, $N(\delta)$ is the set of storage nodes that lie in the storage region of some data node in δ.

Theorem 5.

Given a uniformly random network and a set of failure patterns, the solution returned by DSA allows successful recovery of data with a probability of $(1-\varepsilon)$, when there is a single failure, if the value of ϕ is chosen as:

$$\phi = \frac{1}{n}\mathbf{max}((n+k)c^2 + (k-n)R, c^2(n+k))$$

where k and n are the number of data and storage nodes respectively, R is the radius of the smallest circle that can contain each failure, and c is the small real number such that $g(c) > (1-\varepsilon)$ where $g(x)$ is the Gaussian error function.

Proof of Theorem 5. As discussed in Section 2.2.2, for successful recovery of data when a failure F occurs, the induced subgraph of G' over $(D \cap F) \cup (S - F)$ must have a matching of size $|D \cap F|$. Here, we are using F to also denote the set of nodes destroyed by F. Thus, to prove the theorem, we need to show that such a matching exists with a probability of $(1-\varepsilon)$.

Without loss of generality, let us assume the given failure F to be a square region of size $2R \times 2R$. Recall that R is radius of the smallest circle that can contain each given failure. We will prove the theorem using the following sequence of claims.

- First, note that F does not destroy any node in $N(D_F)$, since the storage region of a data node is more than a distance of $2R$. Thus, it suffices to show that with a probability of $(1-\varepsilon)$, there is a matching of size $|D_F|$ in the induced subgraph in G' over $(D_F \cup N(D_F)$; note that $D_F = (D \cap F)$ and that only the nodes in $N(D_F)$ are useful in finding a matching.

- By Hall's Theorem [64], the above desired matching does not exist iff there is $\delta \subset D_F$ such that $|\delta| > |N(\delta)|$. We show that the probability of this event is at most ε using the following two steps.

  - First, we show in Lemma 5 that if there exist a set $\delta \subset D_F$ such that $|\delta| > |N(\delta)|$ then there is rectangular region L in the region F such that $|D_L| > |N(D_L)|$.
  - Then, in Lemma 6, we show that the probability of such a rectangle L existing is less than ε. Intuitively, this is true due to proportionally smaller size of L in comparison to the union of the storage regions of the data nodes in L.

We now prove the two lemmas used in the above proof.

Lemma 5 *If there exists a set $\delta \subset D_F$ such that $|\delta| > |N(\delta)|$, then there is a rectangular region L in the failure region F such that $|D_L| > |N(D_L)|$.* ∎

PROOF: Consider $\delta \subset D_F$ such that $|\delta| > |N(\delta)|$. Let B_δ be the smallest axis-aligned rectangle that contains δ. Without loss of generality, let us assume that $\delta \subset D_F$ is the set with smallest B_δ that satisfies $|\delta| > |N(\delta)|$.

Since δ is contained in B_δ, we have $|D_{B_\delta}| > |\delta|$. Since, $|\delta| > |N(\delta)|$, we get $|D_{B_\delta}| > |N(\delta)|$. Below, we show that $N(\delta) = N(D_{B_\delta})$, which will imply that $|D_{B_\delta}| > |N(D_{B_\delta})|$ and thus, showing the B_δ is the desired rectangle L.

Showing $N(\delta) = N(D_{B_\delta})$. Essentially, we wish to show that expanding the set of data nodes from δ to D_{B_δ} doesn't necessarily increase their total storage region.

Let us assume that there is a storage node z such that z is in $N(D_{B_\delta})$ but not in $N(\delta)$; let z be the highest (with largest y-coordinate) such storage node. As shown in Figure 2.5, consider the four rectangles Z_1, Z_2, Z_3, and Z_4. Here, Z_1 and Z_2 partition the rectangle B_δ at a horizontal line at a vertical distance of $2R + \phi$ from z, and Z_3 and Z_4 partition $N(D_{B_\delta})$ based on z.[2] We claim the following.

1. Since z is the highest storage node that is in $N(D_{B_\delta})$ but not in $N(\delta)$, each storage node in Z_3 is in $N(\delta)$.

2. Each data node is Z_1 stores its data only in the storage nodes in Z_3, since DSA stores data in storage nodes that are at a vertical distance of at most $2\widehat{R} + \phi$.

3. Number of data nodes in Z_1 that are in δ must be less than the number of storage node in Z_3 that are in $N(\delta)$, since otherwise δ wouldn't be the data set with smallest enclosing rectangle that satisfies $|\delta| > |N(\delta)|$.

The above three claims imply that if we omit the set of data nodes in Z_1 from δ, we get a set of data nodes δ' with a smaller enclosing rectangle (Z_2) that satisfies $|\delta'| > |N(\delta')|$ — which is a contradiction to our original premise. Thus, no such storage node z exists, and hence, $N(D_{B_\delta}) \subseteq N(\delta)$ which implies $N(D_{B_\delta}) = N(\delta)$. ■

Lemma 6 *For a given failure region F, the probability of existence of a rectangle L in F such that $|D_L| > |N(D_L)|$ is less than ε.* ■

PROOF: Recall from lemma 3, for an arbitrary rectangle X over a uniformly random network, both the number of data nodes and number of storage nodes in X

[2]For simplicity, we have used $N(D_{B_\delta})$ to denote the rectangular region corresponding to the union of the storage regions of nodes in D_{B_δ}.

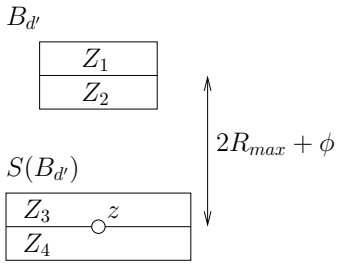


Figure 2.5: Partitioning of rectangles B_δ and $N(D_{B_\delta})$ based on z.

are random variables. We use the probability distribution functions (pdf) of these two random variables to compute the pdf of the difference between the number of data nodes in X and the number of storage nodes in the storage region of X. We then show that if ϕ is chosen as defined in Equation 2.1, the probability is smaller than ε.

As computed in lemma 3 for a rectangle X of size (l, w) expected number of strange node and data nodes is,

$$D_X \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{klw}{(n+k)}, \\ \sigma^2 = \frac{klw}{(n+k)}\left(1 - \frac{lw}{(n+k)}\right)\end{array}\right)$$
$$S_X \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{nlw}{(n+k)}, \\ \sigma^2 = \frac{nlw}{(n+k)}\left(1 - \frac{lw}{(n+k)}\right)\end{array}\right)$$

Distribution of the difference (U).

As computed in lemma 3, for a rectangle of size (l, w) expected number of strange node and data nodes is,

$$D_X \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{klw}{(n+k)}, \\ \sigma^2 = \frac{klw}{(n+k)}\left(1 - \frac{lw}{(n+k)}\right)\end{array}\right)$$
$$S_X \sim \mathcal{N}\left(\begin{array}{c}\mu = \frac{nlw}{(n+k)}, \\ \sigma^2 = \frac{nlw}{(n+k)}\left(1 - \frac{lw}{(n+k)}\right)\end{array}\right)$$

Consider a random rectangle X of size $l \times w$, and let Y be its storage region (i.e., union of the storage regions of the data nodes in X). Note that Y is a rectangle

of size $(l+2) \times \phi$. We now wish to compute the pdf of the random variable $U = S_Y - D_X$. Our goal is to bound the probability of $U < 0$.

Now, given two independent random variables V_1 and V_2 with normal approximations $V_1 = \mathcal{N}(\mu_1, {\sigma_1}^2)$ and $V_2 = \mathcal{N}(\mu_2, {\sigma_2}^2)$ respectively, the probability distribution of $(V_1 - V_2)$ is given by $\mathcal{N}(\mu_0 + \mu_1, {\sigma_1}^2 + {\sigma_2}^2)$. Since the random variables D_X and S_Y are independent, we get the below as the distribution for $U = S_Y - D_X$. Recall that Y is of size $(l+2) \times \phi$.

$$U \sim \mathcal{N} \left(\begin{array}{c} \mu = \frac{n(l+2)(w+\phi)}{ws} - \frac{klw}{(n+k)}, \\ \sigma^2 = \frac{klw}{(n+k)} \left(1 - \frac{lw}{(n+k)}\right) + \\ \frac{n(l+2)(w+\phi)}{(n+k)} \left(1 - \frac{(l+2)(w+\phi)}{(n+k)}\right) \end{array} \right)$$

Simplifications. In order to estimate the probability of $U < 0$, the above pdf must be significantly simplified. We use the following tactics to simplification.

- The mean of U is greater than zero and we want to upper-bound the probability of $U < 0$. Thus, we can consider a higher value for σ and a lower value for μ. Thus, we omit the terms $(1 - \frac{(l+2)(w+\phi)}{(n+k)})$ and $(1 - \frac{(l+2)(w+\phi)}{(n+k)})$ in σ and use $l+2$ instead of l in both μ and σ.
- We can multiply both the mean and standard deviation with a positive number without changing the probability. So, we multiple them both by $(1/(l+2))$.

Applying the above simplifications, we get:

$$U \sim \mathcal{N} \left(\begin{array}{l} \mu = \frac{n(w+\phi)}{(n+k)} - \frac{kw}{(n+k)}, \\ \sigma^2 = \frac{n(w+\phi)}{(n+k)} + \frac{kw}{(n+k)} \end{array} \right)$$

Bounding Pr($U < 0$). Now we want to bound the probability of $U < 0$ by ε, over all possible values of w, by choosing an appropriate value for ϕ. Since U has a normal distribution, the probability of $U < 0$ is less than ε if $\mu > c\sigma$ where $c = g(1-\varepsilon)$ and $g(x)$ is the Gaussian error function.[3] Since both mean and standard deviation of U are positive numbers, we may rewrite the inequality as $\mu^2 > (c\sigma)^2$. Thus, the following equation ensures the desired upper bound on $Pr(U < 0)$

$$\left(\frac{n(w+\phi)}{(n+k)} - \frac{kw}{(n+k)}\right)^2 > c^2 . \left(\frac{n(w+\phi) + \frac{kw}{(n+k)}}{(n+k)}\right)$$

[3]This is a known characteristic of normal distributions.

For given values of n and k, we want to choose ϕ such that the above equation holds for all values of $0 \leq w \leq R$. Solving the above (omitting details), we get

$$\phi = (1/n)\mathbf{max}\left((n+k)c^2 + (k-n)R, c^2(n+k)\right)$$

■

2.6.3 NP-Hardness Proofs

Proof of Theorem 1. We show that our MCDR problem is NP-hard by reducing the well-known 3D-matching (3DM) problem, which is known to be NP-complete [65], to the decision version of the MCDR problem. The decision version of MCDR problem is to check if there is a set of disjoint storing sets (irrespective of the cost), one for each of the data nodes. We start with defining the 3DM problem.

3D-matching. Given three disjoint (unordered) sets X, Y, and Z where $|X| = |Y| = |Z|$, and a relation $T \subseteq X \times Y \times Z$, is there a subrelation $M \subseteq T$ of size $|X|$ (called a maximal 3D-matching) such that for all pairs of elements (x_i, y_i, z_i) and (x_j, y_j, z_j) in M we have $x_i \neq x_j$, $y_i \neq y_j$, and $z_i \neq z_j$. Note that M must contain an element (x_i, y_i, z_i) for each element $x_i \in X$.

Now, consider an instance (i.e., the sets X, Y, Z, and the relation T) of the 3DM problem. Let $T' = X \times Y \times Z - T$. We now construct an instance of our MCDR decision problem from the above as follows.

Constructing an MDNR Instance. Consider a network consisting of data nodes X and storage nodes $Y \cup Z$. We create two types of failure sets:

- For each data node $x \in X$, we add two failures viz., $x \cup Y$ and $x \cup Z$.
- We add a failure set corresponding to each tuple in T'.

Now, we show that the above instance of MCDR has a solution if and only if the 3DM instance has a maximal matching. First, its easy to see that any maximal matching of the 3DM instance gives a solution to the MCDR problem. Below, we show that a solution to the MCDR problem gives a maximal matching to the 3DM instance.

Note that the first type of failure sets dictate that each storing set must contain a node from Y as well as Z. Since $|X| = |Y| = |Z|$ and an MCDR solution must contain $|X|$ disjoint storing sets, each storing set in any MCDR solution must contain exactly two storage nodes (one for each Y and Z). Further, for a data node $x \in X$, if its storing set is (y, z), then (x, y, z) must not be in T' and hence must be in T. Thus, the set of storing sets yields a maximal matching.

To prove that MCDR is NP-hard to approximate, note that an approximate algorithm for MCDR can be also used to solve the above defined decision version of the MCDR problem which is NP-complete. Thus, MCDR is NP-hard to approximate.

Proof of Theorem 2. We prove MCDC is NP-Hard by reducing the GRAPH-COLORING problem to the decision version of the MCDC problem. In the decision version of the MCDC problem, the objective is to determine if there is a solution G' that has a total edge weight of at most a given quantity. Given an instance (G, k) of GRAPH-COLORING, we construct an instance of the MCDC problem as follows.

- The set of data nodes D is V, the set of vertices of G.
- The set of storage nodes are the k colors plus an additional special node s'.
- For each edge $(i, j) \in E$, we construct a failure $\{i, j, s'\}$.
- For each pair of data and storage nodes, the communication cost is one.
- The objective is to find a solution G' of total cost at most $|V| = |D|$.

Note that the above instance of MCDC has a solution of cost at least $|D|$, since in the solution G' each data node must be connected to at least storage node. Also, if there is a solution of cost $|V$, then each data node graph must be connected to exactly one storage node in G'. In such a case, the MCDC solution G' yields a k-coloring of the graph G since (i) each data node is connected to exactly one storage node in G', and (ii) if (i, j) is an edge in G, then data ndoes i and j cannot be connected to the same storage nodes in G' because otherwise the failure $\{i, j, s'\}$ would not allow recovery of data at i or j.

On the other hand, if graph G has a valid k-coloring then we can construct a graph $G'(D \cup S, E')$ of total edge weight $|D|$ by connecting each data node to the storage node corresponding to its assigned color. It is easy to verify that G' is a valid solution to the MCDC problem instance.

Chapter 3

Minimizing Capacity Requirements of Cellular Networks via Delayed Scheduling

3.1 Introduction

Broadband cellular networks are emerging to be the most common means for mobile data access worldwide. Predictions from industry analysts indicate that the volume of data through cellular data networks will increase exponentially in near future [66]. The impact of this data volume on the operators' networks has been carefully analyzed [67]. It is widely anticipated that severe congestion in the cellular network infrastructure is in the offing if not already happening.

The research community has been responding to this challenge using various means. Moving from macro-cells to femto-cells [68] or automatic offloading traffic to WiFi networks [69] have been widely considered. Operators are adding capacity by employing more spectrally-efficient technologies such as WiMax or LTE, adding more spectrum and macro-cells. But these are very capital intensive processes. Spectrum deregulation is also being considered by policy makers [70]. From a more immediate and practical standpoint, cellular operators have started adding pressure on consumers to reduce traffic load by moving away from flat-rate to usage-based pricing model [71], and more recently, throttling data speeds of high volume users [72]. While such strategies can encourage the consumer to optimize usage, they are ultimately detrimental to widespread adoption of cellular networks by discouraging use of bandwidth hungry applications on mobile devices.

Delayed Scheduling. We consider an alternative approach that can be deployed without any additional capital cost while only minimally hurting – if at all – user experience. Several recent studies have reported that the aggregate cellular network traffic load in a region exhibits a diurnal behavior with peak traffic appearing during mid-day and very low volumes during the night [73, 74]. Individual base station traffic also fluctuates widely during the day [73]. Thus, if certain lower-priority traffic can be deferred from peak times to the off-peak times, the congestion issue can be easily alleviated. Many traffic types are amenable to such deferred scheduling. Examples include large downloads such as apps, e-books, videos/pictures, or sync services such as email or cloud-based data. Often, the originating application type (e.g., p2p) is a sufficient hint that a flow can be deferred. At other times, a hint from the programmer or directly from the user may be needed to decide on such flows. Regardless of such mechanics, it is conceivable that a significant reduction of congestion is possible by such deferral. The operator no longer has to design the network for the peak-demand and/or can accommodate much more traffic than is currently possible, without hurting user experience any significantly.

Model. In this work, we address various algorithmic problems that arise in the context of the above philosophy of allowing the flows to be deferred. The basic idea is to have a 'deadline' associated with every flow indicating by when the flow should be completed. The deadline provides a way to specify priority or scheduling laxity.

It can be specified manually or via a profiling mechanism at the application layer depending on the type and size of the flow and probably other contextual information.

Each flow has an associated location, which determines the set of base stations (BS) that can serve the flow. This exploits the fact that the BSes may have overlapped coverage areas. This is a very reasonable assumption in dense deployments. Recent papers on energy savings (e.g., [74, 75]) exploit this fact to turn down BSes to save energy while providing adequate coverage.

We allow a scheduled flow to be preempted to accommodate other (perhaps, more urgent) flows, and rescheduled as many times as necessary, perhaps at neighboring BSes that also covers the location of the flow. Any form of 'rescheduling,' however, only happens at flow arrival or completion (scheduling epoch), allowing such scheduling to work at a higher layer and at a much longer time scale than and independent of the link layer scheduling at the air interface.[1] The actual instantaneous transmission bit rate for the flow could be variable and dependent on the actual radio resource (e.g., bandwidth) allocated at the air interface and the SNR at the mobile client.

Problems Addressed. With the above modeling approach, we address the following problems:

- Determine the minimum capacity needed for the BSes to schedule all the given flows successfully within their respective deadlines. This problem is addressed in two different contexts: all BSes have the same (uniform) capacity or non-uniform capacities. For the former, we design a polynomial-time optimal algorithm, while for the latter, we show the problem to be NP-hard and design a near-optimal algorithm.

- Given the capacity of BSes, schedule the flows in an online manner, so as to maximize the number of flows finished before their deadline. We show the problem to be NP-hard (even, in its offline form). Thus, we consider a special (and more pertinent) case of the problem, and design online and semi-online algorithms with provable performance guarantees.

The focus of this work is to provide efficient solutions for the above problems, under above reasonable modeling assumptions, and to demonstrate potential performance improvements via use of real data (cellular network traces). Encouraged by the results here, our future work will focus on investigating the engineering issues of deploying such mechanisms in a real network.

[1] The median flow inter-arrival time per BS in the data set we are using (described later) is roughly around 100ms to give the reader an idea about the time-scale.

3.2 Model, Problem Formulation, and Related Work

In this section, we describe our model of the cellular network and flow arrivals, give a formal description of the problems addressed, and discuss related work.

3.2.1 Model

Informally, we address the problem of optimizing peak capacity of cellular base stations (BS), when we have the flexibility of delaying (within certain constraints) the incoming flows. Below, we explain our model of the cellular network, cellular BS, and flows.

Cellular Network and Base Station. A cellular network infrastructure consists of a number of cellular base stations (BS) distributed in a two-dimensional geographic region. Each BS is associated with a coverage region of arbitrary shape and size.

Base Station Capacity. Each BS is associated with a capacity. The capacity is best looked upon as the number of channels available to serve the flows. But, in general, the notion of capacity is some measure of the BS's resources to handle the data demands, e.g., amount of bandwidth or number of channels.[2] Capacities may or may not be uniform across BSes; in this work, we consider both settings. The optimization objectives considered in this work are to minimize (i) the uniform capacity, or (ii) the sum of non-uniform capacities.

Flows. Each flow is a sequential stream of data packets, typically semantically related, similar to a TCP or UDP socket connection. For simplicity and clarity, we assume flows and tower capacities to be downlink only; incorporating uplink flows and capacities into your model is straightforward.[3] Put it at the end of "socket connection. Each flow i arrives in the system at a particular time a_i and geographic location l_i, is of a certain size number of bits/packets) s_i, and has a deadline d_i associated with it. The deadline is the time by which the entire flow must be served (as defined below). Note that the deadline value can be used to make a flow "non-deferrable."

Mobility. For simplicity of presentation, we assume that the location l_i remains static (i.e., does not change during its lifetime, even if it is delayed). Our developed techniques easily generalize to the case when the location l_i may change over time, which corresponds to the setting wherein the originating user of the flow is mobile. We discuss generalization of our techniques to mobile users in the end of section 3.3.

[2] We implicitly assume that a BS's capacity is independent of the load and capacity of the neighboring BSes.

[3] To incorporate uplink flows and capacities: (i) our LP formulation of Section III and IV can be easily changed, and (ii) in Section V, we just need to define and use an additional concept similar to the g-capacity. See Appendix for details.

Base Station Serving a Flow; Transmission Rates α_{ijt}. A flow i arriving a location l_i can be served by any BS j whose coverage region contains l_i. A BS serves a flow in its coverage region using exactly one unit of its capacity (e.g., one of its channels). We will relax this assumption, i.e., allow a flow to be served using an arbitrary fraction of a BS capacity, towards the end of Section 3.3 and 3.4. Also, the rate at which a flow i is served by BS j at time t is given by the bit-rate parameter α_{ijt}; this parameter essentially captures the variable link quality dependent bit rate of the downlink.[4]

Preemption and Parallelism. To reflect a practical setting, in our model, we allow a flow being served to be preempted by another flow. The preempted flow can be resumed later, perhaps, at another BS. Thus, essentially, a flow can be broken into parts and each part served at different BSes[5] at different times. However, we do not permit "parallelism," i.e., different parts of the same flow must be served sequentially.

Completely Served. A flow is considered **completely served** if all the parts of it are finished before the deadline.

Model Assumptions and Justifications. We have used some simplifying assumptions to make the problem tractable and to facilitate evaluation over the available network traces which have limited amount of information. We assume that each BS's capacity is constant and independent of the neighboring BSes' capacities. Interference management across BSes is assumed to be perfect (e.g., via prior frequency planning). We assume that the flow size is either known or can be estimated at the flow arrival, and that a flow a continuous stream of packets rather than discontinuous bursts. We do not account for any overhead cost for network controlled hand-offs to move around loads onto different neighboring BSes. But such costs are not hard to account for in the optimization problem. Note that in our schemes such hand-offs only happen at flow arrival or completion times which serve as scheduling epochs.

3.2.2 Problem Motivation, Formulations, and Contributions

In the context of the above described model, we consider the following offline and online problems. The motivation behind the offline problems is to determine optimal capacity needs based on historical traffic information. They can also be used to future traffic growth that can be sustained with the existing network capacities.

[4]In particular, the parameter α allows us to model the fact that neighboring towers may take longer to serve a flow than the original tower where the flow arrives.

[5]We assume that a network controlled hand-off (NCHO) [76, 77] mechanism can be used to achieve this.

More importantly, our offline algorithms are also used to estimate "traffic indicators" which are useful in driving the online algorithms.

1. **Minimize Uniform Capacity (MUC).** Consider a cellular network consisting of BSes with given coverage regions, wherein each BS has a uniform capacity. Given the historical data on a set of flows with associated parameters, the MUC problem is to compute the minimum uniform capacity such that all the given flows can be served within their deadlines.

   In Section 3.3, we design a polynomial-time optimal algorithm for the MUC problem.

2. **Minimize Total Capacity (MTC).** Consider a cellular network consisting of BSes with given coverage regions, wherein different BSes may have different capacities. Given the historical data on a set of flows with associated parameters, the MTC problem is to assign capacities to the BSes such that all the given flows can be served within their deadlines, while minimizing the total sum of capacities.

   In Section 3.4, we show that MTC is NP-hard, and design a polynomial-time near-optimal algorithm for the problem.

3. **Online Scheduling of Flows (OSF).** Consider a cellular network consisting of BSes with given coverage regions and capacities (possibly, non-uniform). At any time instant, a flow may arrive with the associated parameters. The OSF problem is to schedule the flows to BSes in an online manner (i.e., as the flows arrive), while maximizing the number of flows that are completely-served.

   In Section 3.5, we show the above problem to be intractable, and design online and semi-online algorithms for a certain special case of the problem which is more relevant in our context.

3.2.3 Related Work

Theoretical Studies. The offline scheduling problems (MUC/MTC) discussed in this word are similar to the preemptive scheduling problems on identical machines with arrival times and deadlines with the objective of minimizing the number of machines. Although there is a considerable literature on this subject, our model has a key difference: ours is the first preemptive scheduling problem that uses a constraint on the set of machines that can schedule a job. The constraint makes a significant difference as many preemptive scheduling problems with various objective functions [78, 79] (including our MTC problem) are polynomial without such a constraint, but can become NP-hard with the constraint.

The other key difference of our addressed problems with the prior literature is our unique objective of minimizing the number of machines that yield a valid schedule. There has been a considerable amount of work on preemptive scheduling with various objectives such as finding a valid schedule [80], and on minimizing makespan [81][82], number of late jobs [83], lateness [84], job-completion costs [85], etc. However, to the best of our knowledge, there is no work on the objective of minimizing the number of machines for scheduling jobs with arrival times, lengths, and deadlines.

Our online scheduling problem OSF is again a preemptive scheduling problem of jobs with arrival times and deadlines on machines, with the objective of maximizing the number of finished jobs. This problem without the job-machine pairing constraint of our model, has been studied before [86, 87] for a number of different objective functions such as minimizing makespan [88], guaranteed performance [89], etc. However, the constraint on the set of machines a job can use makes our problem much different than the prior-addressed problems.

Load Balancing in Cellular Networks Part of our work is related to the broad topic of load balancing, as we consider "spatial shift" of traffic flows to neighboring BSes that also cover a given flow. This general concept has been widely used at the link layer. For example, see papers on channel assignment, where wireless resources are redistributed rather than traffic [90–92]. In the same note, myriads of scheduling-based approaches are possible at the link layer [93–96]. In contrast, our work reflects scheduling at a higher layer, scheduling at the flow level rather than at the packet/frame level. We thus ignore physical/link layer issues such as power, channels, interference and packet scheduling, instead focus on longer term scheduling of flows – either in whole or in part – assuming the capacity at the BS is largely independent of the traffic in the neighboring BSes. Similar load shifting has been used in cellular networks in the context of energy saving [74].

3.3 Minimizing Uniform Capacity (MUC) Problem

In this section, we address the MUC problem. As mentioned before, the offline MUC problem serves the purpose of determining optimal capacity needs of a network using historical traffic information. In our context, we also use our offline algorithms to estimate "traffic indicators" to drive our online algorithm (as described in Section 3.5). We design an algorithm based on a Linear Programing (LP) formulation, and show that it returns an optimal solution.

LP Formulation, and Challenges. To define a linear program for the MUC problem, we need to divide the time into intervals (not necessarily of same size), and then, for each interval, determine the mapping that defines which flows are served

by each BS. Normally, representation of such a mapping will require use of binary/integer variables, which are not allowed in a linear program. However, our model allows preemption of flows at any time instant – which facilitates representation of the mapping, since preemption allows a BS to serve an arbitrary fraction of a flow, within any time interval. In particular, we represent the mapping in terms of the time used by a flow at a BS for each interval. However, even with the above mapping, we still need to represent, for each interval, an actual "schedule" of flows onto BSes that satisfies the constraints of "non-parallelism." We will show (through Lemma 8) that if intervals and linear constraints are chosen appropriately, then the existence of such a schedule can be guaranteed. In particular, we define the intervals as the time intervals (of possibly different sizes) between time instants of interest (i.e., arrival time or deadline of a flow), as formally defined below; the number of such intervals is polynomial in size of a given MUC problem. The set of equations in our LP formulation are defined as follows.

Variables. Based on the above observations, we define the following notations and variables, for our LP formulation.

- $T = \{T_1, T_2, \dots, \}$ is the finite set of time "instants", where T_t is either an arrival time or a deadline of one of the given flows. We assume T_t's to be in increasing order; thus, $T_t < T_{t+1}$ for all t.
- Variables i, j, t to denote a flow, a BS, and a time instant, respectively.
- Variable k to denote the uniform capacity of BSes.
- Variable x_{ijt} to denote the amount of time the flow i is served by BS j, during the time interval T_t to T_{t+1}.

Equations. On the above variables, we define the following equations:

1. $x_{ijt} = 0$ for all t, and for all i, j, where the location l_i of flow i is not in the coverage region of cell j.
2. $x_{ijt} = 0$ for all j, and for all i, t where $T_t < a_i$ (the arrival time) or $T_t \geq d_i$ (the deadline).
3. $\sum_{t,j} \alpha_{ijt} x_{ijt} = s_i$, for all i, where α_{ijt} is the given bit-rate (a constant in the LP) and s_i is the size of the i flow. This represents the constraint that all the flows must be completely served.
4. $\sum_i x_{ijt} \leq k(T_{t+1} - T_t)$, for all j, t. This represents the constraint that the capacity of each BS j is at most k. Note that the values T_{t+1} and T_t are constants.

5. $\sum_j x_{ijt} \leq (T_{t+1} - T_t)$, for all i, t. This equation attempts to "represent" the non-parallelism constraint, i.e., (a) no flow is served by two different BSes at the same time, and that (b) a flow is served using exactly a unit capacity of a BS. We will show that this equation is sufficient to "represent" the above two constraints, as shown in Lemma 8.

6. **Objective.** Minimize k.

Integral BS Capacities. Note that the above LP returns a real number for the uniform capacity variable k. For example, for the simple MUC instance where there is only one BS and a single flow of size 1 with a deadline of 2, the above LP would return the value of k as 1/2. However, when a flow is served by a unit-capacity only, a BS capacity of 1/2 is not sufficient. In particular, we need to return an integral value for the BS capacity (e.g., when the capacity signifies number of channels and each flow is served by a channel). Thus, our overall algorithm for the MUC problem, called the LP-based algorithm, is to return $\lceil \mathrm{K} \rceil$ as the final value, where K is the objective value returned by the above LP. We now prove the correctness and optimality of this algorithm.

Proof of Correctness and Optimality.

Lemma 7 *The solution ($\lceil K \rceil$) returned by the above LP-based algorithm is a "valid" MUC solution, i.e., using a uniform BS capacity of $\lceil K \rceil$, it is possible to completely-serve all flows.*

PROOF: To prove the lemma, we need to show that an assignment of values to the LP variables that satisfies all the LP equations has a corresponding "schedule" of flows onto BSes (i.e., a function that maps BSes to a set of flows being served for each time instant) satisfying all the constraints of our model of serving a flow at a BS. In essence, we need to prove the following claim:

For any given t, the x_{ijt} values of an LP solution can be converted to a schedule of flows onto BSes such that at any time instant: (a) each BS is using exactly a unit of capacity to serve a flow, and (b) a flow is not being served by multiple BSes.

Once we prove the above claim, it is easy to see that the proof of the lemma follows. The proof of the claim is tedious and rather non-trivial, and hence, deferred to the Appendix (see Lemma 8).

■

Lemma 8 *Given a cellular network with BSes with varying capacities and flows. Let the capacity of BS j be k_j. Consider a time interval $[0, T]$. We are given real values $\{x_{ij}\}$ for each flow i and BS j, signifying that the flow i must be served by BS j for x_{ij} time. The $\{x_{ij}\}$ values are such that (a) for each BS j, $\sum_i x_{ij} \leq k_j T$,*

and (b) for each flow i, $\sum_j x_{ij} \leq T$. We claim that there is a schedule of flows onto the BSes (i.e., mapping of BSes to flows, for each time instant) such that at any time instant: (i) A BS uses exactly a unit capacity to serve a flow, and (ii) Each flow is served by at most one BS. ■

Theorem 7 *The solution ($\lceil K \rceil$) returned by the above LP-based algorithm is an optimal solution of the given MUC problem.*

PROOF: It is easy to see that an optimal solution to the MUC problem satisfies all the equations of our LP formulation. Now, since the LP formulation returns a solution with a minimum value of k, and hence, with a minimum value of $\lceil k \rceil$, the theorem follows. ■

Using Non-Unit Capacity to Serve a Flow. We can easily generalize our model and techniques to the case wherein a flow can be served using an arbitrary fraction of a BS's capacity. Such a model depicts the situation wherein the BS's capacity signifies the size of the available downlink bandwidth, and a flow can be served using any fraction of this bandwidth. To generalize our algorithm to allow use of an arbitrary fraction of BS's capacity to serve a flow, we make the following changes to our LP formulation:

- We let the variable x_{ijt} signify the total amount of BS i's resources (fractional-capacity times allocated-amount-of-time) used to serve flow j in the t^{th} interval.
- We change the 5^{th} equation to:

$$\text{for all } i, t, \sum_j (x_{ijt}/k) \leq (T_{t+1} - T_t).$$

Note that in the above model, the BS capacity can be an arbitrary real number. We can easily generalize Lemma 8 for the above model, which proves the optimality of the solution delivered by the above modified LP, when we allow an arbitrary fraction of a BS's capacity to serve a flow.

Bounding the Capacity. In the above model, we can also bound the amount of capacity that can be used to serve a flow, e.g., a bound on the number of channels that can be used to serve a flow. If c is such an upper bound on the amount of capacity, then we change the 5^{th} equation to:

$$\text{for all } i, t, \sum_j (x_{ijt}/c) \leq (T_{t+1} - T_t).$$

However, we need to change the LP solution K to the next multiple of c, i.e., return $c\lceil K/c \rceil$ as the final solution, for the proof of Lemma 8 to work. This introduces an additive approximation factor of c to the solution delivered by the LP-based algorithm, i.e., if $|\mathrm{OPT}|$ is the optimal BS capacity then the solution returned by the above LP-based algorithm is at most $|\mathrm{OPT}| + c$.

Handling Mobility. Note that mobility of users can be modeled by defining the location attribute l_i associated with the flows to be varying over time. Thus, the location attribute is better represented as l_{it} for each time instant t. The above can be incorporated in our LP formulation as follows:

- Firstly, the time instants of interest will now also include time instants when a location of a flow crosses the boundary of a BS's coverage region.
- Secondly, due to the mobility, the set of BSes, whose coverage region contains the flow's location, changes over time. This can be incorporated by defining the first set of equations of LP appropriately, i.e., constraining x_{ijt} to be zero for every i, j, t where the location of i at time t is not contained in the coverage region of j.

With the above two changes, it is easy to see that the optimality claim of LP-based algorithm can be extended to the MUC problem with mobility.

3.4 Minimizing Total Capacity (MTC) Problem

In this section, we address the MTC problem. As mentioned for the MUC problem, the purpose of an offline algorithm is to optimal the capacity needs of a cellular networks, based on historical traffic information. We start with showing that the MTC problem is NP-hard, and then modify the LP from the previous section to design a near-optimal algorithm for the MTC problem.

Theorem 8 *MTC Problem is NP-Hard.* ■

PROOF: Consider an instance of the disk-set-cover problem, where we are given points and fixed unit-disks in a Euclidean plane, and the problem is to select a minimum number of disks that cover all the given points [97].

Given an instance of a disk-set-cover, we construct an instance of our MTC problem as follows. For each disk, we construct a cell with its coverage region as the disk. For each point, we construct a flow at the point's location, with arrival time as 0, size as 1, and deadline as n, where n is the total number of points in the discrete unit disk cover instance (and hence, the number of flows in the constructed instance of MTC). Thus, all the flows in the system have the same arrival time, deadline, and

size. Now, it is easy to see that any BS can serve all the flows in its coverage region using only a unit-capacity. Thus, the optimal solution of MTC assigns a capacity of either 0 or 1 to the BSes. It is now easy to verify that the optimal solution of MTC yields an optimal solution of the original discrete unit disk cover instance. Thus, the MTC problem is NP-hard. ■

Near-Optimal Algorithm For MTC Problem. Our approximation algorithm is based on the LP formulation from the previous section. We use the same variables and notations from the LP of the previous section, except that we use $\{k_1, k_2, \ldots, \}$ to denote the capacities of the various BSes, i.e., k_j is the capacity of the j^{th} BS. In our below LP formulation for MTC, the non-optimality of the LP solution comes from the fact that the capacity variables are treated as real numbers by the LP (when they are in fact positive integers).

The LP formulation for the MTC problem consists of the same equations as the LP in the previous section for the MUC problem, except that the fourth and sixth set of equations are changed to the following respectively.

4. $\sum_i x_{ijt} = k_j(T_{t+1} - T_t)$, for all j, t.

6. **Objective.** Minimize $\sum_j k_j$.

The above LP returns a solution with real values for the $\{k_j\}$ variables. We take a ceiling of these values to yield integral values for k_j, and return that as the solution for MTC. Thus, if $\{K_j\}$ are the values returned by the LP, we return $\{\lceil K_j \rceil\}$ as the final MTC solution. Below, we show that the solution $\{\lceil K_j \rceil\}$ is such that $\sum_j \lceil K_j \rceil \leq (|\mathrm{OPT}| + J)$, where $|\mathrm{OPT}|$ is the optimal total capacity and J is the total number of BSes in the given problem instance.

Proof of Correctness and Near-Optimality. The following lemma, whose proof is similar to the proof of Lemma 7, states that the above LP-based algorithm delivers a "valid" solution to a given MTC problem.

Lemma 9 *The solution, $\{\lceil K_j \rceil\}$, returned by the above LP-based algorithm is a "valid" MTC solution, i.e., using the BS capacities $\{\lceil K_j \rceil\}$, it is possible to serve all flows within their deadlines.* ■

Theorem 9 *The solution, $\{\lceil K_j \rceil\}$, returned by the above LP-based algorithm is such that the sum of capacities $\sum_j \lceil K_j \rceil$ is at most $|\mathrm{OPT}| + J$, where $|\mathrm{OPT}|$ is the optimal sum of capacities and J is the number of BSes in the input.*

PROOF: It is easy to see that any valid solution of the MTC problem satisfies the equations of the LP formulation. Thus, $|\mathrm{OPT}|$, the value of the optimal solution to the MTC problem is more than $\sum_j K_j$, the value of the optimal LP solution. Since, $\sum_j \lceil K_j \rceil \leq (\sum_j K_j) + J$, we have $\sum_j \lceil K_j \rceil \leq |\mathrm{OPT}| + J$. ■

Corollary 2 *If $|\mathrm{OPT}|$ is at least J, i.e., if each BS uses at least a unit capacity (in other words, no BS is turned off), then the above LP-based algorithm for the MTC problem is 2-approximate.*

Using Non-Unit Capacity to Serve a Flow; Mobility. As in the previous section, we can generalize our techniques to allow use of fractional capacity of a BS to serve a flow. However, in the case of the MTC problem, we need to use a bound c on the units of capacity that can be used to serve a flow in order to maintain the linearity of our LP program. Thus, as before, we let x_{ijt} signify the total amount of resources used in t^{th} interval, and change the 5^{th} equation to:

$$\text{for all } i, t, \sum_j (x_{ijt}/c) \leq (T_{t+1} - T_t).$$

Then, if $\{K_j\}$ is the solution of the LP program, then we return $\{c\lceil K_j/c \rceil\}$ as the solution of the MTC problem. With the above change, we can show that the total capacity of the modified-LP is at most $|\mathrm{OPT}| + cJ$, where $|\mathrm{OPT}|$ is the optimal total capacity and J is the number of BSes in the input.

Finally, mobility can be handled for the case of MTC problem in the similar way as was done for the case of MUC problem.

3.5 Online Scheduling of Flows

In this section, we consider the online version of our problem, i.e., given a cellular network, we want to schedule the arriving flows onto BSes, so as to maximize the number of flows that are completely served. We prove that this problem is NP-hard, and consider the special case of the problem in which the input is such that all the flows can be completely served. For this special case of the problem, we design various semi-online and online algorithms, and prove appropriate performance guarantees. For sake of clarity, we assume a BS uses a unit-capacity to serve a flow. We discuss relaxation of this assumption towards the end of the section.

Online Scheduling of Flows (OSF). Consider a cellular network consisting of BSes with given capacities and coverage-regions. At any instant, a new flow with an associated size and deadline may arrive at a location. The OSF problem is to

schedule the arriving flows in an online manner, so that the number of completely-served flows is maximized. Recall that a flow is considered completely-served if it finishes completely before its deadline, and note that the schedule delivered by an online algorithm may not completely-serve all the flows.

We claim that the OSF problem is intractable. In fact, even the offline version of the problem can be shown to be NP-hard, as the below theorem states. Proof is differed to Appendix.

Theorem 10 *Given a cellular network with possibly non-uniform BS capacities, and flows, the problem of determining (even offline) a schedule of flows onto BSes so as to maximize the number of completely-served flows is NP-hard.* ■

Online Scheduling of Completely-Servable Flows (OSCF). Since the above OSF problem is NP-hard, we consider the OSF problem wherein we restrict ourselves to "completely-servable" instances. A **completely-servable** instance of an OSF problem is an instance for which there exists a schedule of given flows onto BSes such that all the flows are completely-served. We refer to this restricted version as the OSCF problem. We can easily modify our LP formulation to deliver a schedule that completely-serves all flows of a completely-servable instance. However, we have shown through a counter example that there is no optimal online-algorithm possible for the OSCF problem. See below theorem.

Proposition 1 *The offline version of the OSCF problem can be solved optimally in polynomial time.*

Theorem 11 *There is no online algorithm for the OSCF problem that, for every input instance, generates a schedule that completely-serves all the flows.*

PROOF: We prove the theorem using a counter example. Consider a network with two BSes each of unit capacity. We assume the bit-rates to be uniformly unit. We represent a flow i as (a_i, s_i, d_i, R_i), where a_i is the arrival time, s_i is the size, d_i is the deadline, and R_i is the set of BSes where the flow can be scheduled (R_i essentially represents the location of the flow with respect to the coverage-regions of the BSes).

At $t = 0$, three flows arrive in the system $(0, 1, 3, \{1\}), (0, 1, 3, \{2\})$, and $(0, 2, 2, \{1, 2\})$. For this instance, the third flow must be scheduled immediately at $t = 0$. Since the problem (till now) is symmetric with respect to the BSes, we can assume without loss of generality that the online algorithm schedules the third flow on the first BS. Since the second BS is free, we schedule the second flow on the second BS at $t = 0$; this can not hurt the algorithm.

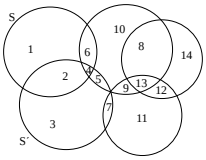


Figure 3.1: Subregions. Here, five BSes with circular coverage-regions give rise to 13 subregions.

Now, at $t = 1$, if the fourth flow $(1, 2, 3, \{1\})$ arrives, then either this or the third flow will never be completely served. But, this instance is certainly a completely-servable instance, since the first and fourth flows could have been scheduled on the first BS, and the second and third flows on the second BS. ■

In the following subsection, we design a semi-online algorithm that solves the OSCF optimally when aided with appropriate statistics on historical traffic pattern and slightly additional capacity. We also present a purely-online heuristic that is optimal for non-overlapping coverage regions.

3.5.1 Semi-Online and Online Algorithms

In this section, we start with designing a semi-online algorithm for the OSCF problem, which is aided by appropriate statistics on historical traffic patterns. We will show that our semi-online algorithm solves the OSCF problem optimally if it is allowed to use certain additional capacity (depending on the variations in traffic patterns) than that used by the optimal offline algorithm. We also design a purely online heuristic. We start with a few definitions.

Covering BS; Remaining Size s_{it}. For a flow i, a BS j is said to be its covering BS if j's coverage region contains l_i, the location of the flow i.

For a flow i, the remaining size at a time instant is denoted by s_{it} and is defined as the size of the remaining (i.e., not yet served) part of the flow i. More formally, in terms of notations of previous two sections, $s_{it} = s_i - \sum_j \sum_{t'<t} x_{ijt'}$.

Allowable Delay w_{it}. At a time instant t, the allowable delay w_{ij} of a flow i is the maximum amount of time by which the remaining part of i can be delayed, while being completely-served. More formally, $w_{it} = (d_i - t) - s_{it}/\alpha_{it}$, where d_i is the deadline, s_{it} is the remaining size at t, and $\alpha_{it} = \min_j \alpha_{ijt}$ (the minimum bit-rate across BSes).

Subregions r_m. Given a 2D network region, we define a subregion r_m as the set of points in the 2D plane that lie within the same set of coverage regions. Note that the set of subregions are disjoint. See Figure 3.1. For circular coverage-regions, it is easy to show that the total number of subregions is $O(n^2)$ where n is the number of BSes [98].

Traffic Indicators $f_p(r_m, t)$. Consider a cellular network and a set of completely-servable historical instances of flows $\{M_1, M_2, \ldots, \}$. Each M_p is essentially a set of flows with associated parameters, such that the flows can be completely-served by some schedule S_p. For a subregion r_m, time instant t, and an instance M_p, we define the traffic indicator $f_p(r_m, t)$ as the number of flows located in r_m that are being served by a BS at time t, in a schedule S_p of M_p that completely-serves all flows. Note that S_p can be computed in polynomial-time by Proposition 1.

Semi-Online Global (SOG) Algorithm. Given cellular network and a set of completely-servable historical instances $\{M_1, M_2, \ldots, \}$, let $f_p(r_m, t)$ be the traffic indicators as defined above. Let

$$g(r_m, t) = max_p \ \ f_p(r_m, t).$$

The Semi-Online Global (SOG) algorithm uses the above g values to schedule a given online instance of flows as follows.

At each time instant t, for each subregion r_m, pick $g(r_m, t)$ (or less, if not available) flows (with non-zero remaining size) located in r_m with least allowable delays. Let this set of flows for the entire network be S. Find the largest subset S' of S that can be scheduled onto BSes at time t; this can be done by finding the maximum-matching problem between S and "servers," where a BS of capacity k is represented by k servers. Finally, we can also add more flows (that are not in S) to schedule, if possible.

Performance of SOG Algorithm. We now show that the SOG algorithm solves the OSCF problem optimally when aided with slightly additional capacity, which depends upon the variation of f values across historical instances and the deviation of the input instance from the historical instances. We start with a definition.

Definition 4 (g-Capacities.) For a given cellular network and set of historical instances of flows $\{M_1, M_2, \ldots, \}$, we define the g-capacity for each BS as follows. On the given cellular network, consider the following instance of flows: For each subregion r_m and time instant t, we create $g(r_m, t)$ flows of size 1, arrival time t, and deadline $t + 1$. We solve this MTC problem using the LP-based algorithm, and call the resulting BS capacities as the g-capacities of the BSes. □

Theorem 12 *Given a cellular network and set of historical instances of flows $\{M_p\}$. The SOG algorithm would completely-serve all the flows of any instance M_p, if it uses the g-capacities for the BSes.*

PROOF: Consider an instance M_p. Now, by definition of f values, there exists a schedule S_p that schedules $f_p(r_m, t)$ flows (onto some BSes) from each subregion

r_m at each instant t. Using an exchange argument, we can assume that the $f_p(r_m, t)$ flows being scheduled are the ones with the least allowable delay, for each r_m and t. Since SOG uses the g-capacities, it has sufficient capacity to schedule $g(r_m, t)$ flows for each r_m and t. Since $g(r_m, t) \geq f(r_m, t)$, the theorem follows. ■

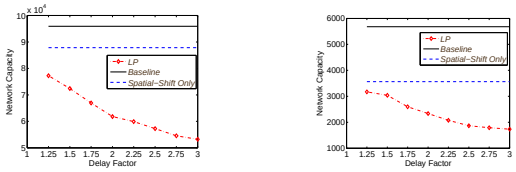
By the above theorem, note that if we had only one historical instance M_1, then the SOG algorithm can completely-serve all the flows in M_1 using at most $n + O$ total network capacity, where O is the optimal capacity needed and n is the number of BSes in the network. This is because $g(r_m, t) = f_1(r_m, t)$ for all r_m, t, and hence, the sum of g-capacities is at most $n + O$ by Theorem 9. In general, if (i) there is minimal deviation in $f_p(r_m, t)$ values across the given historical instances, and (ii) the given input to SOG is "similar-enough" to the historical instances, then SOG will completely-serve all the flows of the given input with minimal additional capacity compared to the optimal required.

Semi-Online Localized (SOL) Algorithm. The above SOG algorithm is not localized, since it needs to solve the matching problem at each time instant. We can also consider the simpler Semi-Online Localized (SOL) algorithm that instead computes a maximal matching greedily, which can be done in a localized manner. Note that any maximal matching is of size at least half of the maximum matching.

Purely-Online (PO) Heuristic. The above semi-online algorithms are aided by the g values computed from historical data, and hence, is not purely online algorithm. Below, we present a purely-online (PO) heuristic, which is motivated by the fact that it is optimal for networks with non-overlapping coverage-regions.

Heuristic Description. At a high-level, the PO heuristic does the following at each time instants of "interest." It orders the flows (with non-zero remaining size) in increasing order of their allowable delays at that instant. Then, it tries to "match" these flows with BSes greedily, as described in detail below. The time instants of interest (at which the above is done) are: (i) arrival of a new flow, (ii) completion of a flow, and (iii) passing of the deadline of a non-scheduled flow.

To match flows onto BSes, we consider the flows in the increasing order of their allowable delays, and try to schedule each flow i as follows. **If** there is a covering BS j of i that is free, then we schedule i on j. **Else**, we try free up some serving BS of i, by finding an "alternating path" and "shuffle" some flows along this path as follows. We find an alternating sequence of flows and BSes $(x_1, y_1, x_2, y_2, \ldots, x_m, y_m)$ such that (i) each x_q is a flow and each y_q is a BS, (ii) $x_1 = i$ and y_1 is a covering BS of x_1, (iii) for all $q \geq 1$, y_q is currently serving x_{q+1}, (iv) y_m is a covering BS of x_m, and (v) y_m is currently free (i.e., not serving any data request). If such an alternating path exists, then we stop serving all the flows $\{x_2, x_3, \ldots, x_m\}$, and reschedule each flow x_q to y_q. The above is thus able to schedule $x_1 = i$ on y_1, if an alternating path as described above exists. If there



(a) Uniform Tower Capacities. (b) Non-uniform Tower Capacities.

Figure 3.2: Total capacity requirements for varying delay factors, for various offline algorithms.

is no alternating path, then we do not schedule i at this point and consider the next flow in order.

As mentioned before, the motivation and intuition behind the above PO heuristic is that it can be shown to be optimal for the special case of the OSCF problem when the coverage regions of the BSes are disjoint. However, the non-optimality of the heuristic for the general case of arbitrarily overlapping coverage regions cannot be bounded.

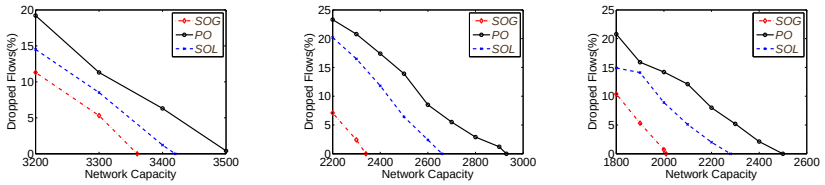
Theorem 13 *The above PO heuristic solves the OSCF problem optimally, if the coverage regions of the BSes are disjoint.*

PROOF: Note that due to disjoint coverage regions, it is sufficient to prove the theorem for the case of a single BS. For the single BS, the PO heuristic essentially schedules flows in the order of allowable delays. Now, consider the schedule of an offline algorithm that schedules all the flows within their deadlines; such a schedule exists by definition of the OSCF problem. In such a schedule, we can show that if there is a time instant wherein a flow with higher allowable-delay is scheduled before a flow with a lower allowable-delay, then "exchanging" them in the offline schedule still ensures that all flows are scheduled within their deadlines. ■

Using Non-Unit Capacity to Serve a Flow. We now discuss how our techniques of this section can be extended to allow non-unit capacities to serve a flow. If c is the bound on the number of units of capacities of a BS that can be used to serve a flow, then we make the following changes: (i) Extend the definition of allowable delay as follows: $w_{it} = (d_i - t) - s_{it}/(c\alpha_{it})$. (ii) Make $\lceil c \rceil$ copies of each flow in S, when solving the matching problem for SOG and SOL. (iii) Allow multiple units of flow to be scheduled simultaneously (up to the bound of using c units of BS capacity at a time) at each BS, in the PO Heuristic.

3.6 Simulations

In this section, we analyze the performance of our various algorithms on real cellular network data set collected at the core of a commercially operated 2G/3G net-



(a) Delay-Factor of 1.25. (b) Delay-Factor of 2.25. (c) Delay-Factor of 3.

Figure 3.3: Percentage of dropped flows for varying total network capacity, for various semi-online/online algorithms. Here, the starting network capacity value is the near-optimal capacity requirement as computed by the LP-based algorithm.

work. The data used in the evaluation consists of flow level (UDP or TCP flows) statistics collected for 101 base stations for a one week period in 2007. The region covered is about 80 square km spanning both dense urban and suburban areas. There are about 1 million flows in the data set considered.[6] In the data set, each flow has an arrival time, BS-id where it is served, and a flow size. We assume that the flows to be served at a constant rate equal to its size divided by its duration.[7]

Since obviously there is no delayed scheduling in this network, the flows are immediately served with no delay. Flows that are served are recorded in the data set. Flows that could not be served due to unavailability of network resources do not have any record.

Using a simulation model with this data set, we will show that (i) there is a considerable reduction in capacity requirements, when flows are allowed to be delayed even by a small factor, and (ii) with slightly additional capacity than the near-optimal network capacity (computed by the offline LP-based algorithms), our online semi-online algorithms are able to completely-serve all the flows.

Flow Locations, Coverage Regions, Delay-Factors. Since our data set neither includes the exact flow locations nor the BS coverage regions, we determine the set of covering BSes for a flow as follows: (i) We construct the Voronoi diagram over the BS locations; (ii) If a flow i arrives at a BS j, then we assume that it can be served by any BS j' whose Voronoi region is adjacent to that to j.[8] Essentially, a flow i

[6]For proprietary reason we are unable to provide further details about the network. We believe that the missing details will not hurt the readers' understanding of our work.

[7]Technically, we are given flow duration, how long within this duration the flow was inactive (i,e., no resource scheduled) and number of bytes served for this flow. However, it is impossible to decipher from this aggregated information: (i) when the flow was descheduled either for resource limitations or plain inactivity (in 3GPP standard the resources can be scheduled even when a flow is dormant, until an inactivity timer fires), (ii) what bit rates the flow was served with when it was indeed scheduled.

[8]In our data set, the distribution of BSes is roughly uniform: more than 70% of the geographically closest neighbors are within 1 to 2 miles. Thus, coverage assumption based on Voronoi is not too different from a coverage assumption based on the geographic distance.

arriving at BS j can be served by j and any of its "neighboring" BSes. Defining the serving BSes of a flow in the above manner precludes the need to artificially generate flow-locations (which are absent from the network trace).

Also, the flows in our data do not have deadlines associated with them. So, we run simulations for various "delay-factors" and define deadlines based on the delay-factor; in particular, for a delay-factor of c, the deadline of a flow i is computed to be $a_i + c\alpha_{ijt}s_i = a_i + cs_i$ where a_i and s_i are the given arrival time and size respectively and α_{ijt} is assumed to be 1 (for lack of record of low-level parameter values in our data).

LP-Based Offline Algorithms: Delay Factor vs. Capacity Requirements. We start with analyzing the effect of deferring flows on the capacity needs of the network. Thus, in Figure 3.2, for varying delay-factor, we plot the total network capacity required to completely-serve all the flows as computed by our LP-based algorithms. We consider both cases, viz., the uniform as well as non-uniform BS capacities, and use all 7 days of data.[9] In the graph, for comparison purposes, we also plot two values corresponding to the case of delay-factor of one (i.e, no-delays): (i) capacity requirements when each flow can be served only by the BS it arrived at as recoded in the data set, and (ii) capacity requirements when a flow can be served by multiple BSes (as determined by Voronoi tessellation described above). We refer to these values as `Baseline` and `Spatial-Shift Only` respectively.

As expected, the capacity requirements decrease with the increase in the delay-factor. The decrease is substantial even with minimal delays. For example, even with a delay factor of 1.25 (flows can be delayed only up to one-fourth of their size), the capacity requirements reduce by about 50% for the non-uniform case and about 20% for the uniform case. This changes to about factors of 3 and 2 respectively for a delay factor of 3 (i.e., flows can be delayed up to twice their size). Note that a sizable reduction comes from the ability of "load balancing" via scheduling on less-loaded neighboring BSes.

Semi-Online and Online Algorithms: Capacity Requirements vs. Percentage of Flows Dropped. In this set of plots, we run various semi-online/online algorithms, viz., (i) Semi-Online Global (SOG), (ii) Semi-Online Localized (SOL), and (iii) Purely-Online (PO). We run the above algorithms over Wednesday's data (the results were similar for other weekdays), while using the remaining four weekdays of data to compute the statistical g values used by the semi-online algorithms. For increasing total network capacity, we plot the number of flows that were not

[9]To solve the LP program over 1 million flows and about 100 BSes in a reasonable amount of time, we employed the following divide-and-conquer strategy: we divided the data into appropriate 3-4 hour durations, computed the LP solution for each input independently, and then "combined" the solutions to get a valid (but, perhaps, suboptimal) LP solution. Thus, the network capacity numbers for LP in Figure 3.2 and 3.4 are perhaps a slight overestimate of the best possible LP-based solution.

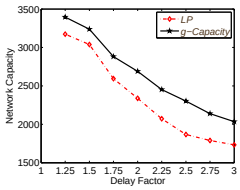


Figure 3.4: The g-capacity of the network, and the near-optimal offline capacity requirement, for varying delay-factors.

completely-served (i.e., dropped). See Figure 3.3. Here, we vary the network capacity by proportionally increasing all the BS capacities starting from (i) the value of the near-optimal LP-based solution (for the non-uniform case) for the given flows, till (ii) the algorithm is able to completely-serve all flows. We ran the algorithms for three different delay-factors. We observe that both the traffic indicators as well as the global-matching scheme have considerable impacts on reduction of capacity requirements. In particular, SOG uses very minimal additional capacity (about 5% more) over the offline capacity needs, to completely-serve all the flows. Even with the same capacity as the offline capacity, the drops are marginal (less than 10%).

g-capacities vs. Near-Optimal Capacities. In Figure 3.4, we show for various delay-factors: (i) the near-optimal (LP) network capacity needed to completely-serve all seven days of flows, and (ii) the g-capacity of the network, based on all seven days of data. We observe that the g-capacity of the network is only slightly higher (only about 5-15%) than the offline capacity, which suggests that SOG needs only slightly higher network capacity to completely-serve all the flows in any of the historical instances. In effect, this simulation result shows that the variation of $f_p(r_m, t)$ values across the historical instances is minimal enough that the inefficiency introduced by semi-online processing of flows as done in SOG is minimal compare to the near-optimal offline processing.

Chapter 4

Capacity Optimization of Femtocell Networks

4.1 Introduction

Mobile data traffic is predicted to grow exponentially in the next few years [99], and thus, cellular networks are in increasing demand for more capacity. Since spectrum bandwidth is limited, spatial reuse is the most feasible way to accommodate such increasing demand. Femtocell technology has been emerging as a solution to the increase of both capacity and coverage, while reducing both the capital expenditures and operating expenses of cellular networks. Essentially, femtocells are small cellular base stations, typically designed for use in homes or small businesses; they extend coverage indoors, by connecting to the service provider's network via broadband.

In this chapter, we consider the problem of assigning channels and powers to the femtocells in order to maximize either the total network capacity or the minimum capacity available to any femtocell. The resource allocation problem in femtocells is unique and challenging for various reasons. Firstly, the femtocell deployments are unplanned, and significantly more dense compared to the planned deployments of macrocells. Hence, while interference in macrocell networks may be localized at macrocell edges, it is more pervasive across femtocells. Thus, techniques such as fractional frequency reuse that are effective for macrocells are inadequate for femtocell networks [100]. Secondly, unlike WiFi networks, femtocells operate synchronously on licensed spectrum bands [100].

To the best of our knowledge, the only work on femtocells' capacity maximization is [101]; however, they provide a non-polynomial algorithm which is impractical for large networks. Since interference is the major factor that reduces capacity, intuitively minimizing interference implies higher capacity. A recent work [100] addresses the problem of interference mitigation in femtocell networks through careful resource management; however, their technique is based on a simplistic (pairwise) interference model. In our work, we model capacity based on Shannon's law and physical interference model [102]. To the best of our knowledge, our paper is the first to provide an approximation algorithm for power and channel assignment to maximize the total network capacity in femtocell networks as defined by Shannon's law.

Our Contributions. In this study, we consider the problem of power and channel assignment to maximize the total network capacity or the minimum capacity at a femtocell in a femtocell network. In the context of the above problem, our contributions in this work are:

- We show that the addressed problems are NP-hard (see Section 5.2).
- For the problem of maximizing the total network capacity, we provide a constant-factor approximation algorithm for large uniform networks, for fem-

tocells with arbitrary coverage regions, and generalize it to an efficient heuristic for general networks (see Section 4.3).

- For the problem of maximizing the minimum capacity at a femtocell, we provide an algorithm with appropriate performance guarantees for general networks wherein there is a lower bound on the minimum distance between femtocells. Our algorithm is based on our-designed constant-approximation algorithm for the related problem of minimizing the number of channels used in a femtocell network (see Section 4.4).

- Finally, we demonstrate the efficiency of our algorithms through extensive simulations, wherein we show that our algorithms deliver solutions with objective value close to an upper bound on the optimal value, and much better than that delivered by a naive divide-and-conquer algorithm (see Section 4.5).

4.2 Model and Problem Definition

In this section, we describe our model, formulate the problems addressed, and prove their NP-hardness.

Femtocells and Network Architecture. Femtocells are short-range, low-power base stations installed by customers to provide increased coverage and capacity in a small area, such as a home or a small office. Femtocells communicate with the service providers directly through a wired link (e.g., broadband in the house). In our assumed architecture, each femtocell is assigned a channel (part of the available spectrum) and a power by a central entity (called the spectrum broker) for communication with devices in its region. The assignment of powers and channels should be ideally done in such a way that: (i) each femtocell is able to provide coverage in its required "coverage region" (i.e., say the home it is deployed in), and (ii) some notion of "aggregate" resulting capacity across the entire network region is maximized.

To facilitate the above, each femtocell communicates with the spectrum broker through the wired link; in particular, each femtocell relays appropriate parameters (e.g., ambient noise) to the spectrum broker, and the spectrum broker periodically determines (in a global manner) the power and channel to be assigned to the femtocell.

A femtocell communicates with its "users/devices" (which are in its coverage region) as follows. It divides the assigned channel into "sub-channels" (also called Physical Resource Blocks or PRBs), and uses one or more sub-channels to communicate with each user. The femtocell uses orthogonal sub-channels for different users, and hence, there is no interference among the users within a femtocell. Thus,

in this study, we focus on mitigating the interference at users across different femtocells – which we implicitly model by considering the worst case scenario, i.e., each femtocell is using all the sub-channels (i.e., the entire assigned channel) at all times.

Below, we formally describe our model of a femtocell.

Definition 5 (Femtocell) A femtocell f is a small device associated with a fixed (unless otherwise specified) location, a coverage region, and a maximum transmission power denoted by $\overline{P}$. The coverage region can be of arbitrary shape, and let r be the distance from the femtocell's center to the farthest point in any coverage region.

Also, the purpose of deploying a femtocell at its location is to provide cellular coverage to any device(s)[1] within its coverage region. The femtocell communicates with device(s) in its coverage region using an assigned channel and a transmission power that is less than $\overline{P}$, the maximum allowed. □

The maximum transmission power $\overline{P}$ allowed for a femtocell takes into account both the technical limits of a femtocell as well as additional power constraints required to limit signal leakage outside the femtocell's coverage region.

We now define the capacity available in a femtocell coverage region, for a given assignment of channels and powers to femtocells in the given network.

Capacity in a Coverage Region. Consider a set of femtocells, each associated with its location and coverage region. Let $l(f)$ denote the location of a femtocell f. Also, let each femtocell f be assigned a channel $c(f)$ of bandwidth B and a transmission power $P(f)$. Let s be the number of sub-channels. Then, the sub-channel's capacity $C_s(f,p)$ available due to f at a point p within its coverage region is given by Shannon's law as follows.

$$C_s(f,p) = \frac{1}{s}B\log_2\left(1+\frac{\frac{1}{s}P(f)d_{l(f),p}^{-\alpha}}{\sum_{f'|f'\neq f,c(f')=c(f)}\frac{1}{s}P(f')d_{l(f'),p}^{-\alpha}+\frac{1}{s}N}\right) = \frac{1}{s}B\log_2\left(1+\frac{P(f)d_{l(f),p}^{-\alpha}}{\sum_{f'|f'\neq f,c(f')=c(f)}P(f')d_{l(f'),p}^{-\alpha}+N}\right),$$

where α is the path-loss exponent, $d_{a,b}$ is the distance between two locations a and b, and N is the ambient noise. Thus, the total capacity $C(f,p)$ yielded by a femtocell at a point p is

[1]Our model allows both open and closed access. In case of closed access, a femtocell ignores non-registered devices even if they are in its coverage region.

$$C(f,p) = B\log_2\left(1 + \frac{P(f)d_{l(f),p}^{-\alpha}}{\sum_{f'|f'\neq f, c(f')=c(f)} P(f')d_{l(f'),p}^{-\alpha} + N}\right),$$

Since our purpose is to guarantee a certain capacity at every point in a femtocell's coverage region, we define the capacity yielded by a femtocell in its coverage region $A(f)$ as:

$$C(f) = \min_{p\in A(f)} C(f,p). \tag{4.1}$$

Problems Addressed. We address the problems of assignment of channels and powers to the given femtocells so as to maximize two different notions of "aggregate" capacity of the femtocell network; in particular, we aim to maximize the total or minimum capacity.

Maximize-Total-Capacity (MTC) Problem. Given a set F of femtocells, k homogeneous channels, the MTC problem is to assign a channel $c(f)$ and transmission power $P(f)$ to each femtocell $f \in F$, such that (i) $P(f) \leq \overline{P}$, the maximum transmission power, and (ii) the total capacity $\sum_{f\in F} C(f)$ is maximized. □

Maximize-Minimum-Capacity (MMC) Problem. Given a set F of femtocells, k homogeneous channels, the MMC problem is to assign a channel $c(f)$ and transmission power $P(f)$ to each femtocell $f \in F$, such that (i) $P(f) \leq \overline{P}$, the maximum transmission power, and (ii) the minimum capacity $\min_{f\in F} C(f)$ is maximized. □

In the problems above, each femtocell receives one channel. We are going to extend these problems to multiple channels in the following sections.

We start with stating the NP-hardness of both of the above problems; the proof of the above theorem is deferred to Appendix B.1. In the following sections, we will circumvent this intractability by designing appropriate approximation algorithms.

Theorem 14 *MTC and MMC problems are NP-hard.* ■

Sub-channel Assignment to Users. Each femtocell divides the assigned channel into sub-channels (PRBs), and partitions the sub-channels into disjoint sets (to ensure no interference) across its users. If we assume that each femtocell uses all of its sub-channels (i.e., the entire spectrum of the channel assigned), then, for downlink communication, the capacity received by a user is independent of the location and sub-channels assigned to the users of other femtocells. Thus, the assignment of sub-channels (to their users) by the femtocells can be done independently of each other, without any impact on performance.

4.2.1 Related Work

To the best of our knowledge, the only work on femtocells' capacity maximization is [101]. However they provide an exponential algorithm which cannot be applied to

realistically large networks. Under appropriate assumptions, maximizing the capacity is the dual problem of minimizing the interference. A recent work [100] gives a method for assigning channels in order to minimize the interference. However they use a pairwise interference model, which has the limitation that interference is determined statically, and does not take into account that changing channel assignment and transmission power can change the interference patterns. On the other hand, our work is based on the physical (SINR) model [102], which is a more realistic interference model. Another recent work, [103], focuses on minimizing the interference between femtocells and macrocells using a distributed algorithm. However, they only prove that femtocells reach a Nash equilibrium, while our method aims at maximizing the capacity. The authors of [104] show how to maximize the capacity by time isolation by scheduling femtocells' transmissions in time slots. However, their method requires synchronization between femtocells, while we do not assume any synchronization. Heuristic methods for fractional frequency reuse are proposed in [105] and [106]. While assigning orthogonal channel guarantees better capacity for certain femtocells, it does not necessarily optimize the network capacity, which is our main objective. Many recent works [103, 107–110] have studied the problem of the signal leaking outside the femtocell areas. In this study, we handle the leaking problem by imposing an upper bound on the transmission power. This bound can be estimated with appropriate sensors or it can be determined using a method similar to the one of [103]. Other works [111–113] focus on determining the optimal power assignment to femtocells, without addressing the channel assignment problem. There are methods [114–116] to optimize the coverage of a femtocell network, while our objective is to optimize capacity in given coverage regions.

Channel assignment in wireless networks has been discussed in many papers (see [117] for a survey). The major difference respect to these works is that our model does not explicitly consider individual users, but each femtocell is responsible for coverage inside an entire area. Fallgren [118] considered the problem of jointly assigning users to base station, assigning powers and channels for general wireless networks. He showed that maximizing the total Shannon capacity is NP-hard to approximate within any constant factor. However, his result does not apply to our problem because femtocells are only required to provide coverage inside their respective coverage regions. The problem of jointly assigning channels and powers has also been considered in the context of power minimization, [119, 120]. Even though they consider the problem of assigning and power and channels to the base stations (without any predetermined user assignment), therein the challenge is to select minimum number of base stations to keep active, while we focus on maximizing the capacity with all femtocells.

4.3 The Maximizing-Total-Capacity (MTC) Problem

In this section, we address the objective of maximizing the total capacity in a femtocell network. In particular, we design an algorithm that provably returns a constant-factor approximate solution for large uniform networks. We start by giving a brief description of the algorithm.

Basic Idea. The basic idea of the algorithm is to assign channels to femtocells (nodes) in such a way that the femtocells assigned any particular channel are "spread out" as much as possible across the network. To facilitate the above, we compute a certain distance R for the given network, and assign channels to nodes such that any pair of nodes assigned the same channel are at least R distance away from each other. For a given R, assignment of channels to nodes under the above constraint is done by extracting a large "k-dependent subgraph" from the R-disk graph over the given nodes, and then, k-coloring the nodes in the extracted subgraph. For uniformly random networks, R can be computed directly from the uniform density of the network, while for general networks, the "best" R can be searched in an efficient manner.

Algorithm for Uniform Networks. Now, we give a formal description of our algorithm to solve the MTC problem. We start by describing our algorithm for uniform networks. Before giving a formal description of our algorithm, we give a formal definition of the k-dependent set and uniform networks.

Definition 6 (k-dependent Set.) For a graph $G(V, E)$ and a positive number k, a set of vertices $V' \subset V$ is called a k-dependent set of G, if V' induces a subgraph in G where each node degree is at most k. □

There is a PTAS (polynomial-time approx. scheme) [121] to compute the maximum k-dependent set in a graph.

Definition 7 (Uniform Random Network.) A femtocell network is said to be uniformly random if it can be generated by a Poisson point process. In a uniformly random network, the expected number of femtocells (nodes) in any constant-size disk is a constant. □

Proof of Approximation. We now show that the above MTC-Algorithm is a constant-factor approximation algorithm for the MTC problem, for large uniform networks. The proof is mainly based upon the claim that in large uniform networks with unit-disk coverage regions, the following assignment of powers and channels yields a constant-factor approximate network capacity: (i) assign the power uniformly, and (ii) assign the available channels uniformly with the same density. By observing that our algorithm assigns power and channels in the similar manner as

above, we can deduce that our algorithm is also near-optimal (within a constant factor) for unit-disk graphs. We then generalize our approximation result to general coverage regions.

Theorem 15 *The MTC-Algorithm delivers a constant-factor approximate solution for the MTC problem, for large uniform networks.*

PROOF: The proof is built over a lemma which is proved later, for sake of clarity of presentation. The proof of the theorem can be presented in the following sequence of steps.

1. First,

   (a) We note that the in MTC-Algorithm the density of distribution of different channels is same. This is the direct result of two facts. The graph being colored is a unit disk graph of a large size constructed from uniformly distributed node in the plain and our coloring algorithm pick a color for every node with uniform random probability from list of available colors.

   (b) We show that for any particular channel, the distribution of the nodes with the said channel is uniform random. Assume a very large circle C of radius xR such that xR is very large but much smaller than size of the network. The expected number of colored nodes in xR with is at most x^2K. The total number of nodes for of each particular channel is at most $(x+1)^2$. Since we have k channels, the expected number of nodes of the same channel is x^2 and the ratio of the expected number of nodes versus the maximum is $\frac{x^2}{(x+1)^2}$ which is almost equal to 1 for large x. Hence, the expected number of nodes of any channel in any circle of size xR imposed on the network is almost constant which means that the distribution of the nodes of all channels are uniform constant distribution in the plain.

2. First, we note that the MTC-Algorithm yields a solution wherein (i) for any particular channel, the set of nodes assigned that channel are uniformly distributed across the network, and (ii) the density of distribution of different channels is same. To see the above, observe that the value of R ensures that in any disk of radius R, the expected number of femtocells is k. Also, in G, these k nodes are connected to each other, and each is assigned a different channel. Thus, for any channel c, the expected number of nodes assigned the channel c is exactly 1 in any disk of radius R. This implies the above two claimed properties of MTC-Algorithm's solution.

3. Second, as shown in the following lemma, for unit-disk coverage regions (i.e., for $(r, 1)$ pseudo-disks), there is a constant-factor approximate solution for the MTC problem that (i) assigns uniform power to all nodes, and (ii) satisfies both the above given properties (that are satisfied by MTC-Algorithm's solution). See Lemma 10.

4. Third, for large uniform networks of size n, the expected number of nodes in G that have degree more than $(k-1)$ can be shown[2] to be at most $n/2$. Recall, that G is the R-disk graph over the set of femtocells, where R is as defined in Step 2 of the algorithm.

5. From the above three points, it is easy to see that MTC-Algorithm would yield a constant-factor approximate MTC solution for unit-disk coverage regions; here, the constant of approximation is twice of the constant in the second claim above.

6. The above approximation proof can be extended to general coverage regions as follows. Since each femtocell has a point that is at a distance of r from the center, the capacity $C(f)$ of a femtocell with a general coverage region is the same as the capacity of a femtocell with a coverage region of a disk of radius r. Thus, the above result holds.

■

Now, to complete the proof, we only need to prove the claim used in the second point of the above theorem. We do so in the following lemma.

Lemma 10 *For an MTC problem on large uniform random networks with unit-disk coverage regions, there is a solution within a constant-factor of the optimal that (a) assigns uniform power to all nodes, and (b) assigns channels to nodes, such that (i) for any particular channel, the set of nodes assigned that channel are uniformly distributed across the network, and (ii) the density of distribution of different channels is same.*

PROOF: The proof of the lemma follows from the following three main claims.

1. Consider a set of femtocells with unit-disk coverage regions and undetermined locations, and a network region A. Suppose there is only one available channel. For the problem of placement (determining locations) and assignment of powers to the femtocells to maximize total network capacity, it can be shown that the solution, wherein the femtocells are distributed

[2]For a Poisson distribution of density λ, the probability that there are more than λ nodes is given by $\sum_{k=\lambda}^{\infty} \lambda^k/(k!e^{\lambda}) \leq 1/2$.

uniformly in A and assigned uniform power, achieves near-optimal (within a constant factor) network capacity. The above claim follows directly from a prior single-channel result [122] on placement and power-assignment of sender-nodes to maximize aggregate throughput over a set of links. Here, each sender is transmitting to a single receiver that is within a constant distance of the sender.

2. Now, consider a large network of uniformly distributed femtocells, and two available channels h_1 and h_2. Each femtocell is assigned uniform power P and one of the channels, and let S_1 and S_2 be the sets of femtocells that are assigned h_1 and h_2 respectively. Assume, S_1 and S_2 are uniformly distributed. The total capacity of the femtocell network is maximized when the density of distribution of the nodes in S_1 and S_2 is the same.

   The proof of the above claim is rather non-trivial, and is given in the next paragraph.

3. Now, based on the above two points, the lemma can be proved as follows. Start with an arbitrary optimal MTC solution O. Since the total network capacity is the sum of the network capacities for each channel, by the first claim above, the solution O's total capacity cannot reduce if it is changed as follows: (i) assign uniform power $\overline{P}$, the maximum transmission power, to all femtocells, and (ii) reassign channels, s.t., for any particular channel, the set of nodes assigned that channel is uniformly distributed. Note that at this point the solution may still have different distribution densities for different channels. Now, by the second claim above, the solution O's capacity cannot reduce if the channels are reassigned to make the distribution densities for different channels the same; this can be done, by considering two channels at a time. This proves the lemma.

To complete the proof of the lemma, we now prove the second claim above.

Proof of Second Claim Above. Let us consider a femtocell f assigned the channel h_1 and the uniform power P. Let p be a point in f's coverage region. Let r_p be the distance of p from f's location, Now, we can compute the capacity $C(f,p)$ at p due to f as:

$$C(f,p) = B \log \left(\frac{P r_p^{-\alpha}}{\sum_i P r_i^{-\alpha} + N} + 1 \right)$$

where r_i is the distance from p to a femtocell i, and i ranges over the set $S_1 - \{f\}$; recall, that S_1 is the set of femtocells using the first channel h_1. For infinitely large networks, the above can be written in terms of the probability distribution of the

femtocells to get the expected capacity at p due to f.

$$C(f,p) = B\log\left(\frac{Pr_p^{-\alpha}}{P\delta_1\int_y^{\infty}(2\pi r)r^{-\alpha}dr + N} + 1\right)$$

where δ_1 is the density of distribution of femtocells in S_1, and $y > 0$ (since all other femtocells are a non-zero distance away). By replacing $\int_y^{\infty}(2\pi r)r^{-\alpha}dr$ with an appropriate constant[3] c_1 we get:

$$C(f,p) = B\log\left(\frac{Pr_p^{-\alpha}}{Pc_1\delta_1 + N} + 1\right)$$

Now, the above quantity is minimum when r_p is the maximum possible, i.e., 1, since we are considering unit-disk coverage regions. Thus, by Equation (4.1), we get:

$$C(f) = \min_p C(f,p) = B\log\left(\frac{P}{Pc_1\delta_1 + N} + 1\right)$$

Note that the above value of $C(f)$ applies to any femtocell f in S_1 (since S_1 is a uniform distribution in a large (infinite) network of femtocells). Similarly, we can compute the expected capacity of a femtocell in S_2, i.e., a femtocell assigned the second channel h_2. If δ_2 is the density distribution of femtocells in S_2, then the total network capacity of the femtocells in $(S_1 \cup S_2)$ per unit area of the network is

$$C = B\delta_1\log\left(\frac{P}{Pc\delta_1 + N} + 1\right) + B\delta_2\log\left(\frac{P}{Pc\delta_2 + N} + 1\right)$$

For appropriate constants c_1 and c_2, the above can be written as

$$C = B\left(\delta_1\log\left(\frac{c_1}{\delta_1 + c_2} + 1\right) + \delta_2\log\left(\frac{c_1}{\delta_2 + c_2} + 1\right)\right) \tag{4.2}$$

For a constant $\delta_1 + \delta_2$, we show below that the above equation is maximized when $\delta_1 = \delta_2$, which proves the theorem.

Maximizing Equation (4.2). Let $\delta_1 + \delta_2 = \delta$, where δ is a constant. We show that for any value of δ_1 and δ_2 other that $\delta/2$, Equation (4.2) has a lower than maximum value. We start by substituting δ_1 and δ_2 with $\delta/2 + x$ and $\delta/2 - x$ respectively, which makes the body of the logarithm as:

$$\left(\frac{c_1}{\frac{\delta}{2} + x + c_2} + 1\right)^{\frac{\delta}{2}+x}\left(\frac{c_1}{\frac{\delta}{2} - x + c_2} + 1\right)^{\frac{\delta}{2}-x}$$

[3]This integral converges since $\alpha > 2$.

Let $f_1(x) = c_1/(\frac{\delta}{2} + x + c_2) + 1$ and $f_2(x) = c_1/(\frac{\delta}{2} - x + c_2) + 1$. We are going to show that (i) $(f_1(x))^x (f_2(x))^{-x}$ maximizes for $x = 0$, and that (ii) $f_1(x) f_2(x)$ also maximizes at $x = 0$. This should suffice to prove that (4.2) maximizes at $x = 0$.

(i) We show that any value $x > 0$ gives the expression a smaller (or equal) value than at $x = 0$. Thus, we need to show that:

$$\left(\frac{c_1}{\frac{\delta}{2} + x + c_2} + 1\right)^x \left(\frac{c_1}{\frac{\delta}{2} - x + c_2} + 1\right)^{-x} \leq 1$$

where the right hand side is the value of the expression for $x = 0$. Reorganizing the terms and using the fact that $x > 0$ we get

$$\frac{\frac{\delta}{2} + c_1 + c_2 + x}{\frac{\delta}{2} + c_2 + x} \frac{\frac{\delta}{2} + c_2 - x}{\frac{\delta}{2} + c_1 + c_2 - x} \leq 1$$

which can be easily verified.

(ii) To show that $f_1(x) f_2(x)$ maximizes at $x = 0$, we show that any value $x > 0$ gives the expression gives a smaller (or equal) value than $x = 0$. Thus, we need to show that:

$$\left(\frac{c_1}{\frac{\delta}{2} + x + c_2} + 1\right)\left(\frac{c_1}{\frac{\delta}{2} - x + c_2} + 1\right) \leq \left(\frac{c_1}{\frac{\delta}{2} + c_2} + 1\right)^2$$

where the right hand side is the value of the expression for $x = 0$. We observe that the right hand side is bigger than 1, while the left hand side is less than 1 (as shown in (i) above). ■

Heuristic for General Networks. We now discuss how the MTC-Algorithm can be efficiently adapted for general (i.e., non-uniform) networks. Note that for uniform networks, R depends upon the uniform density, which is not well-defined for arbitrary networks. Thus, to adapt MTC-Algorithm for arbitrary networks, essentially, we need to find an appropriate R. The purpose of R is to ensure that pairs of nodes assigned the same channel are at least a certain distance away. Thus, a high value of R is desirable. On the other hand, a very high value of R will result in a small size of $(k-1)$-dependent set. Thus, ideally, we desire a large R as well as a large $(k-1)$-dependent set. Note that an increase in R results in a decrease in size of the maximum $(k-1)$-dependent set. Based on the above observations, we adapt the MTC-Algorithm as follows, for general networks.

- For efficiency, we use the known PTAS [121] for computing a near-optimal $(k-1)$-dependent set (instead of the original simple approach, which sufficed for the case of uniform networks).

- We try to find the highest value of R that results in a "large enough" $(k-1)$-dependent set. This can be achieved by starting with a small R, iteratively doubling it, and computing the $(k-1)$-dependent set for each R until the dependent set becomes smaller than desired.

Assigning Multiple Channels to Femtocells. In our original problem formulation, we assigned a single channel to each femtocell. However, in certain context/protocols, e.g., Long Term Evolution (LTE) [123], each femtocell can be assigned multiple channels. In particular, in LTE, each femtocell can be assigned up to four contiguous channels. Below we generalize our algorithm and results for the case where each femtocell can be assigned up to t contiguous channels. The algorithm is the following.

- For j from 1 to t:
  - Group channels into "super-channels", each containing j contiguous channels.
  - Execute the MTC-Algorithm to assign the super-channels.
- Keep the solution with largest total capacity among those obtained above.

It is easy to show the following approximation factor.

Theorem 16 *The above described algorithm delivers a t-factor approximate solution for the MTC problem, for large uniform networks, when each femtocell may be assigned up to t contiguous channels.* ■

4.4 The Maximizing-Minimum-Capacity (MMC) Problem

In this section, we address the MMC problem, wherein the objective is to maximize the minimum capacity. Our algorithm for the MMC problem is based on the solution of another complementary problem, viz., the Minimum-Channels (MC) problem which aims to minimize the number of channels used while ensuring that a certain minimum capacity is guaranteed within each femtocell's coverage region. We design an approximation algorithm for the MC problem, and use it to design an algorithm for our MMC problem. We start by defining the Minimum-Channels problem.

Min-Channels (MC) Problem. Given a set of femtocells F (with associated locations and coverage regions) and a minimum capacity requirement C_{min}, the MC

problem is to assign channels and powers to the given femtocells in order to ensure that the capacity $C(f)$ at each femtocell's region is at least C_{min}. The objective of the problem is to use the minimum number of channels.

Basic Idea for Algorithm. The algorithm for the MC problem is based on the simple intuition that to maximize capacity, we should minimize interference, and thus, assign any particular channel only to sufficiently far away nodes. For a given set of femtocells and C_{min}, the below lemma computes a value R such that if nodes that are assigned the same channel are at least R distance from each other, then $C(f) \geq C_{min}$ for each femtocell, when femtocells are assigned uniform power. We will use this result to design an algorithm and show that it delivers an approximate solution to the MC problem.

Lemma 11 *Consider a femtocell network consisting of a set F of femtocells with associated locations and coverage regions. Let C_{min} be the given capacity requirement for each femtocell. Let us define the value R to be such that:*

$$R \geq \frac{2}{\sqrt{3}} \left(r + \sqrt[\alpha]{6\zeta_{\alpha-1} \Big/ \left(\frac{r^{-\alpha}}{\beta} - \frac{N}{P} \right)} \right),$$

where N is the ambient noise, α is the path-loss exponent, $\beta = 2^{C_{min}/B} - 1$ for a given bandwidth B, $\zeta_{\alpha-1}$ is the Riemann zeta function for $(\alpha - 1)$, r is from Definition 5, and P is some power value (used below).

We claim that if each femtocell is assigned the uniform power P, and channels are assigned to femtocells in such a manner that any two femtocells assigned the same channel are at least R distance away, then the capacity $C(f)$ for each femtocell f is at least C_{min}.

PROOF: As defined in the lemma, let $\beta = 2^{C_{min}/B} - 1$ where B is the bandwidth. For the power and channel assignment suggested in the lemma, let us compute the minimum SINR at any point p in a femtocell's coverage region. The SINR at p in a coverage region of a femtocell f is at least

$$\frac{P\, r^{-\alpha}}{I_{max} + N},$$

where I_{max} is the maximum interference possible at p due to other femtocells assigned the same channel as f. The value of I_{max} can be bounded because each pair of femtocells assigned the same channel is at least a distance R away. In fact, it is easy to see that the interference at the location of f is maximized when all femtocells assigned the same channel as f are placed on a triangular lattice [124, 125] as shown in Fig. 4.1. In such a scenario, there are 6 femtocells on the first hexagon of

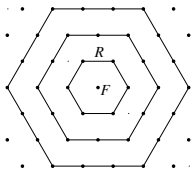


Figure 4.1: Highest density placement of femtocells.

edge-length R, 12 femtocells on the second hexagon of edge-length $2R$, and so on. In general, there are $6i$ femtocells on the i^{th} hexagon of edge-length iR. It is easy to see that a femtocell on the i^{th} hexagon is at a distance of at least $\frac{\sqrt{3}}{2}iR$ from the center (f's location). Thus, the total interference at f's location is at most

$$\sum_{i=1}^{\infty} 6Pi \left(\frac{\sqrt{3}}{2}iR\right)^{-\alpha}.$$

To bound the total interference at a point p in f's coverage region, note that p is at most at distance r from f's location. Thus, the interference I_{max} at a point p in f's coverage region can be bounded as:

$$\begin{aligned} I_{max} &\leq \sum_{i=1}^{\infty} 6Pi \left(\frac{\sqrt{3}}{2}iR - r\right)^{-\alpha} \leq \sum_{i=1}^{\infty} 6Pi \left(\frac{\sqrt{3}}{2}iR - ir\right)^{-\alpha} \\ &= \sum_{i=1}^{\infty} 6Pi^{1-\alpha} \left(\frac{\sqrt{3}}{2}R - r\right)^{-\alpha} = 6P \left(\frac{\sqrt{3}}{2}R - r\right)^{-\alpha} \sum_{i=1}^{\infty} i^{1-\alpha} \end{aligned}$$

The series $\sum_{i=1}^{\infty} i^{1-\alpha}$ converges to the Riemann zeta function $\zeta_{\alpha-1}$ for $\alpha > 2$. For example, if $\alpha = 2.5$ then $\zeta_{1.5} \simeq 2.612$. Plugging this into the above bound for I_{max}, we get that the minimum SINR at p is at least

$$\frac{P\, r^{-\alpha}}{6P\zeta_{\alpha-1}\left(\frac{\sqrt{3}}{2}R - r\right)^{-\alpha} + N}.$$

Now, it is easy to verify that the above quantity is greater than β for the given value of R, and thus, the capacity at the point p due to f is at least C_{min}. ■

Approximation Algorithm for the MC Problem. Based on the above Lemma, we propose the following algorithm for the MC problem.

Assign channels to the femtocells greedily, while ensuring that each pair of femtocells within a distance of R are assigned different channels. Set all femtocells' powers to $\overline{P}$.

Theorem 17 *If the minimum distance between any pair of femtocells is bounded below by a constant, then the MC-Algorithm is a constant-factor approximation algorithm for the MC problem.*

PROOF: If the distance between any pair of femtocells is bounded below by r', then any disk of radius R can contain at most R^2/r'^2 femtocells. Thus, Algorithm 4.4 uses at most R^2/r'^2 channels, where R is from Lemma 11. Since the optimal MC solution uses at least one channel, and R^2/r'^2 is a constant (as each of the parameters in R is a constant), the theorem follows. ■

Note that it is reasonable to assume a lower bound between any pair of femtocells, since deployment of femtocells is expected to be spread out — generally, one per building/room, which can be assumed to be of a minimum size.

Algorithm for the MMC Problem. Based on the above approximation algorithm for the MC problem, we can now design an efficient algorithm for the MMC problem. Recall that the MMC problem is to maximize the minimum capacity across the set of given femtocells with associated locations and coverage regions, using the given number of available channels.

Starting with a "small-enough" C_{min}, perform a binary search on C_{min}, to find the largest C_{min} such that the MC-Algorithm uses no more than the allowed number of channels.

Theorem 18 *In the above algorithm for the MMC problem, if we were to use a polynomial-time optimal algorithm for MC problems (in lieu of the constant-factor approximation algorithm MC-Algorithm), then MMC-Algorithm becomes a PTAS for the MMC problem.*

PROOF: The proof of the following theorem follows from the fact that the number of channels used by an optimal MC algorithm is monotonically non-decreasing with increase in the input parameter C_{min}. ■

Even though, the above theorem falls short of showing that our proposed algorithm for the MMC problem (MMC-Algorithm) is an approximation algorithm, it nevertheless provides strong theoretical evidence of the competitiveness of the proposed MMC-Algorithm. We corroborate the above theoretical evidence further through extensive simulations.

Assigning Multiple Channels to Femtocells. The MMC problem has been formulated to assign a single channel to each femtocell. However, in certain context/protocols, e.g., Long Term Evolution (LTE) [123], each femtocell can be assigned multiple channels. We generalize our algorithm to assign up to t contiguous channels as follows.

- For j from 1 to t:
  - Group channels into "super-channels", each containing j contiguous channels.
  - Execute the MMC-Algorithm to assign the super-channels.
- Keep the solution with largest minimum capacity among those obtained above.

4.5 Simulations

In this section, we present the results of our simulations. In particular, we compare the objective value of our algorithms' solutions with (i) an appropriately derived upper bound on the optimal value, and (ii) the objective value of a simple divide and conquer algorithm's solution. We consider three different network settings:

- `Uniform`: femtocells' locations generated at random, with uniform density, inside circular regions of increasing radii.
- `Urban`: femtocells placed in houses selected at random from a dense urban area. We used Google Earth to determine the house locations (the locations of the houses we are choosing from are drawn in Fig. 4.2(a)).
- `Rural`: femtocells placed in houses selected at random from a residential rural area. Again, we used Google Earth to determine the house locations (the locations of the houses we are choosing from are drawn in Fig. 4.2(b)).

We start by giving the details of how we obtained the upper bound on the optimal objective value, and then, describe the simple divide and conquer algorithm. We then present various parameters of our simulations, followed by our simulation results.

Upper Bound on Optimal Value. The upper bound on the optimum value is obtained by tessellating the network, and solving a mixed integer quadratic program (MIQP) optimally in each "tile." This MIQP is similar to the one used in [101], but with our desired objective function (for more details, please see Appendix B.2). The size of each tile is chosen so that the corresponding MIQP can be solved optimally within reasonable time; we chose the size of each tile to contain approximately 20-25 femtocells. Each tile is then padded with a boundary of width 25 m, with all the femtocells in this boundary-region considered in the computation of the interference. Essentially, we compute the capacity of each femtocell in the tile excluding the ones in the above defined boundary-region (as they are considered within their own tile), but for the sake of computing interference we include the femtocells in

(a) (b)

Figure 4.2: Locations of houses extracted from Google Earth: (a) from a dense urban area, and (b) from a residential rural area.

the boundary-region also. However, since for each tile, we ignore the interference due to far away femtocells (which are neither in the tile nor in the boundary-region), the computed capacity values represent an upper bound on the optimal values. The above method works for both objectives, viz., maximum total capacity and maximum minimum capacity.

Divide and Conquer Algorithm. The divide-and-conquer algorithm starts with the entire network region, and recursively splits it into two halves of equal size. The lowest level of the recursion tree represents subregions where the number of femtocells is at most the number of available channels; here, in these subregions, each femtocell is assigned a different channel. Thus, the above divide-and-conquer algorithm assigns different channels to femtocells in the same subregion (lowest level of recursion), while ignoring the interplay between femtocells across subregions.

Parameter Values. In all simulations, we use the following parameter values. We assume 8 available channels. The path-loss exponent is set to $\alpha = 2.5$ for communications within the coverage region, since it is indoor, and for the outdoor communications it is set to $\alpha = 3.5$ for the `Uniform` and `Urban` environments, and to $\alpha = 3.0$ for the `Rural` one [126]. The bandwidth of each channel is set to $B = 5$ MHz, the noise is set to $N = 10^{-10}$ W in the `Uniform` and `Rural` settings, and to $N = 10^{-8}$ W in the `Urban` setting. The maximum power for a femtocell is set to 1 W in all settings and simulations [123]. The coverage regions of femtocells are disks of radius 10 m for all settings. We ran two sets of experiments:

- Increasing regions: Network regions are circular regions of radius ranging from 100 m to 1 km. In the `Random` setting, the network density is kept constant to $1/(400\pi)$, while in the `Urban` and `Rural` settings, femtocells are randomly placed in 25% of the houses selected from the locations shown in Fig. 4.2.

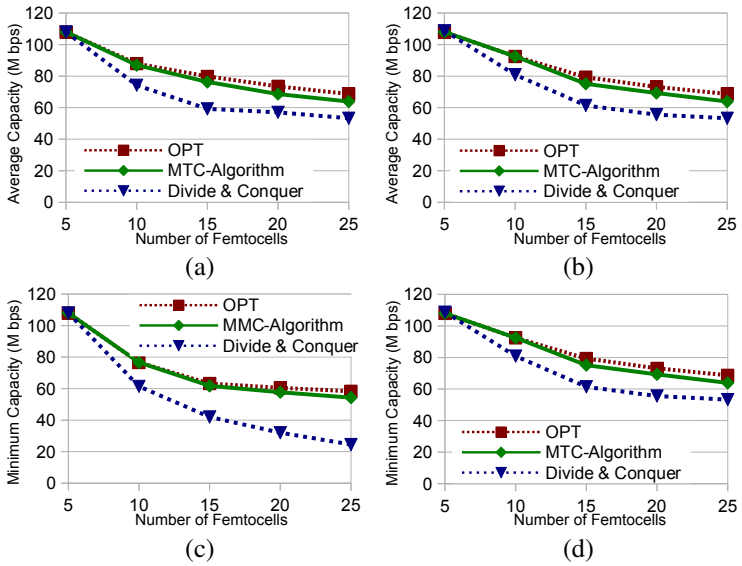


Figure 4.3: Results for small size networks in which the optimum can be computed exactly. Plots (a) and (b) are for our MTC-Algorithm, and plots (c) and (d) are for our MMC-Algorithm. The figures correspond to: (a) and (c) networks with constant density of femtocells in circular regions of increasing sizes, and (b) and (d) networks with increasing density of femtocells in circular regions radius 100 m.

- Increasing density: Network regions are circular regions of radius 1 km. In the `Random` setting, the network density ranges from $1/(40000\pi)$ to $1/(400\pi)$, while in the `Urban` and `Rural` settings, the number of selected houses ranges from 5 % to 50% (candidate house locations are shown in Fig. 4.2).

Simulation Results for the MTC problem. Simulations results for small networks of constant density of femtocells in circular regions of increasing sizes are shown in Fig. 4.3(a), and for small networks of increasing density in circular regions of radius 100 m are shown in Fig. 4.3(b). In these cases, the optimum can be computed exactly. For comparison purpose, we plot the average network capacity instead of the total network capacity. As we can see, both our MTC-algorithm and the Divide and Conquer Algorithm give optimal solutions for networks with only 5 femtocells as this is smaller than the number of channels. However, for networks with more femtocells than channels, our algorithm remains very close to the optimum, while the solution of the Divide and Conquer method gradually degrades as the network size increases. The results for the MTC problem for larger networks with constant density of femtocells in regions of increasing size and for for larger networks with increasing density of femtocells in a circular region of radius 1 km are shown in Fig. 4.4(a),(b),(c) and 4.4(d),(e),(f) respectively. Also in this case, we

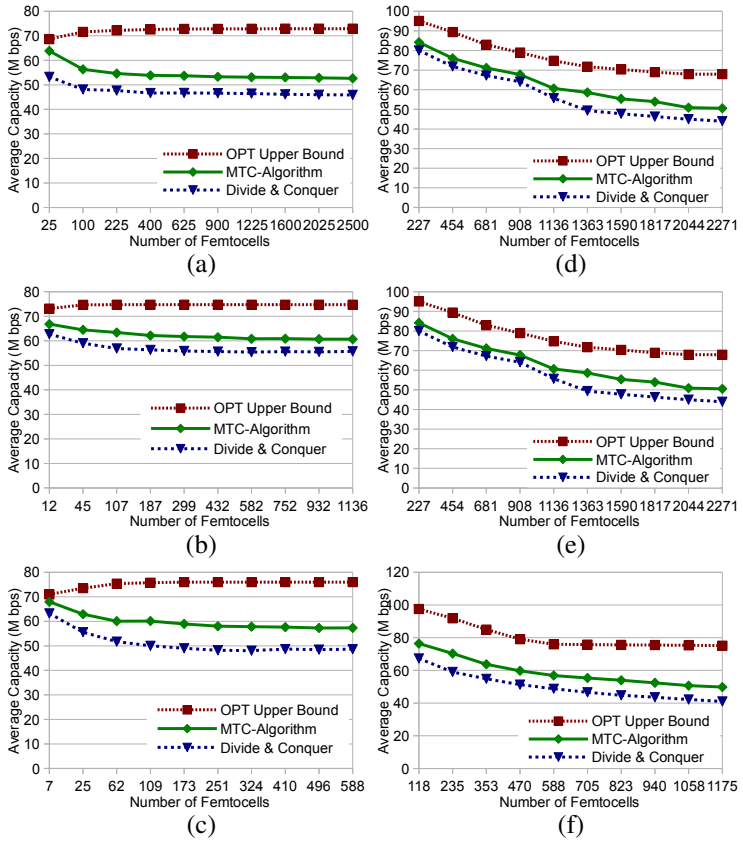


Figure 4.4: Maximum average capacity for our MTC-Algorithm for: (a),(b),(c) networks with constant density of femtocells in regions of increasing size, and (d),(e),(f) networks with increasing density of femtocells in a circular region of radius 1 km. The scenarios are: (a),(d) uniform, and non-uniform in (b),(e) urban and (c),(f) rural settings, respectively.

observe that the solution of our algorithm is better than that of the naive divide-and-conquer approach. More importantly, for the MTC problem, we see that our heuristic for general networks, when run on rural and urban scenarios, gives similar performance as the approximation algorithm for uniform networks. For networks of size larger than 25, the the optimal networks was derived with the method described above (by combining optimal solutions of subnetworks of 20-25 femtocells each). For this reason, in Fig. 4.4(a),(b),(c), the value of the optimum is lower for 25-size networks. Our MTC algorithm performs better for smaller networks, and the capacity slightly decreases as the network size increases. This can be justified by the fact that the overall interference increases as the network's size grows, so

small errors are amplified in larger networks. This is even more pronounced for the Divide and Conquer algorithm.

Simulation Results for the MMC problem. Simulation results with small networks with constant density of femtocells in circular regions of increasing sizes, and with small networks of increasing density in circular regions of radius 100 m are shown in Fig. 4.3(c) and 4.3(d) respectively. In these cases, the optimum can be computed directly. Similarly to what happened for the MTC problem, both our MCC-algorithm and the Divide and Conquer Algorithm give optimal solutions for networks with only 5 femtocells, as this is smaller than the number of channels. However, for networks with more femtocells than channels, our algorithms remain very close to the optimum, while the solution of the Divide and Conquer method gradually degrades as the network size increases. The results for the MMC problem for larger networks with constant density of femtocells in regions of increasing size are shown in Fig. 4.5(a),(b),(c), and for larger networks with increasing density of femtocells in a circular region of radius 1 km is shown in Fig. 4.5(d),(e),(f). The naive Divide-and-Conquer algorithm performs particularly bad for the MMC problem, in comparison with our MMC-Algorithm for the MMC problem which is based on the approximation-algorithm for the MC problem. We can observe that the rate at which the optimal solution decreases in Fig. 4.5(d),(e),(f) is smaller for larger networks. This is due to the method we used to derive the upper bound optimum by combining optimal solutions of subnetworks of 20-25 femtocells each.

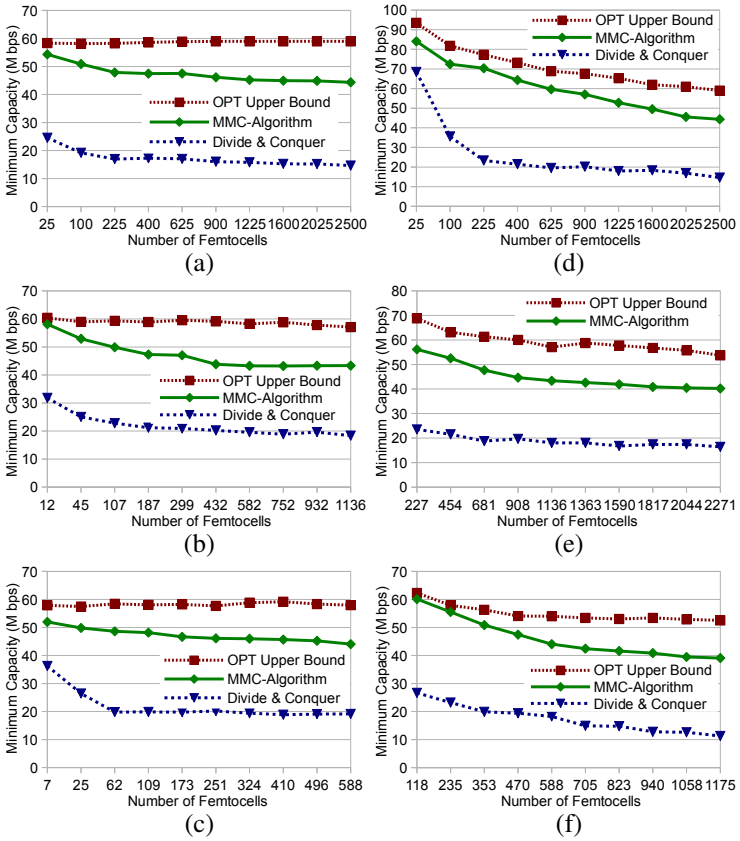


Figure 4.5: Maximum of minimum capacity for our MMC-Algorithm for: (a),(b),(c) networks with constant density of femtocells in regions of increasing size, and (d),(e),(f) networks with increasing density of femtocells in a circular region of radius 1 km. The scenarios are: (a),(d) uniform, and non-uniform in (b),(e) urban and (c),(f) rural settings, respectively.

Chapter 5

Optimal Spectrum Management in Two User Interference Channels

5.1 Introduction

This chapter addresses the problem of maximizing sum of user capacities in multiuser communication systems in a common frequency band. We consider a continuous frequency domain. For frequency-selective channels, we prove that in an optimal solution, each user must use the maximum power available to it. This maximum-power result also holds in the case wherein the objective is to maximize the product of user capacities; this objective is generally used to achieve proportional fairness. For the special case of two users in flat channels, we present an optimal spectrum management solution.

In a multiuser communication system, users either have to partition the available frequency (FDMA), or use frequency sharing (i.e., each user uses the entire spectrum), or a combination of the two (i.e., use partially-overlapping spectrums). Intuitively, FDMA is the optimal answer in the case of strong cross coupling (also referred to as strong interference scenario), and frequency sharing is optimal when the cross coupling is very weak. In the intermediate case, the optimal solution may be a combination of the two strategies [127] (i.e., users may use partially-overlapping spectrums).

There exist an extensive literature on the effect of cross coupling on choosing between FDMA and frequency sharing. The works in [128] and [129] provide sufficient conditions under which FDMA is guaranteed to be optimal; these conditions are group-wise conditions, i.e., each pair of users need to satisfy the condition. Recently, Zhao and Pottie [127] derived a tight condition which when satisfied by a pair of users guarantees that the given pair uses orthogonal frequencies (i.e, FDMA for the pair). Their result holds for any pareto optimal solution.

In the general interference scenarios in multiuser systems, the weighted sum-rate maximization problem is a non-convex optimization problem, and is generally hard to solve [130]. However, two general approaches have been proposed: (i) One approach considers the Lagrangian dual problem decomposed in frequency after first descretizing the spectrum [131]; the resulting Lagrangian dual problem is convex and potentially easier to solve [132, 133]. More importantly, [133] proves that the duality gap goes to zero when the number of "sub-channels" goes to infinity. However, the time-complexity of their method is a high-degree polynomial in the number of sub-channels (thus, becoming prohibitively expensive for the continuous frequency domain problem). (ii) The second approach changes the formulation of the problem to get an equivalent primal domain convex maximization problem [127]. Eventhough, the above approaches almost reduce the spectrum management problem to a convex optimization problem, they fall short of designing an optimal or approximation algorithm with bounded convergence.

The recent works in [127, 134] find the optimal solution for the special case of two "symmetric" users; their result is very specific, and doesn't generalize to non-

symmetric links. In another insightful work, [135] gives a characterization of the optimal solution for the two-user case which essentially yields a four to six variable equation. Our work essentially improves on these results and solves the problem for the general case of two users, using an entirely different technique.

Discrete Frequency Spectrum Management. In other related works, [133] and [136] consider the spectrum management problem in discrete frequency domain, wherein the available spectrum is already divided into given orthogonal channels and user power spectral densities are constant in each channel. Their motivation for considering the discrete version is to facilitate a numerical solution [133]. The discrete version is shown to be NP-hard (even for two users), and in [129] the authors give a sufficient condition for the optimal to be an FDMA solution. Even when restricted to FDMA solutions, they observe that the discrete version remains inapproximable, but provide a PTAS [136] for the continuous version (when restricted to FDMA solutions). Note that, for two users, the discrete version remains NP-hard [129], while the continuous version has been solved optimally (Section 5.4). Thus, discretizing the spectrum seems to make the spectrum allocation problem only harder, contrary to the motivation in [133]. Moreover, discretization of a given spectrum can actually reduce achievable capacity.

Our Results. In this chapter, we address the following spectrum management problem: Given a spectrum band of width W and a set of n users each with a maximum transmit power, the SAPD (spectrum allocation and power distribution) problem is to determine power spectrum densities of the users in the continuous frequency domain to maximize the sum of user capacities (as computed by the generalized Shannon-Hartley theorem). For the above SAPD problem, we present the following results.

- For frequency-selective channels, we show that in an optimal SAPD solution, each user must use the maximum transmit power. We extend the result to the case wherein the objective is to maximize the product of user capacities.
- For the special case of two users in flat channels, we design an optimal solution for the SAPD problem. This is a direct improvement of the recent recent in [127] which solves the problem optimally for the special case of two users with symmetric (equal channel gains and noise) and flat channels.

5.2 Problem Formulation, and Notations

Model, Terms, and Notations. We are given a set of users i (formed by a transmitter s_i and a receiver r_i) and a frequency spectrum $[0, W]$. The background noise at the receiver of user i is assumed to be white, i.e., constant across the spectrum, and

has a constant value of N_i (Watts/Hz) at each frequency. We use $h_{ij}(x)$ to denote channel gain between the sender of user i and the receiver of user j at frequency x.

Power Spectrum Density (PSD) $p_i(x)$; Total Power. For a user i, the power spectral density (PSD) is a function $p_i : [0, W] \mapsto \mathbb{R}_{\leq 0}$ that gives the power at each frequency of the signal used by the transmitter s_i to communicate with its receiver r_i. Thus, $p_i(x)$ is the power of s_i's signal at frequency x. We allow arbitrary PSD functions. The total power used by a user i is given by $\int_0^W p_i(x)dx$.

Maximum Total Power. Each user i is associated with a maximum total power P_i, which is the bound on the total power used by its transmitter s_i. That is, each PSD function $p_i(x)$ must satisfy the below condition:

$$\int_0^W p_i(x)dx \leq P_i. \tag{5.1}$$

Spectrum Used. Given a PSD function $p_i(x)$ for a user i, the spectrum used by user i is defined as $\{x|p_i(x) > 0\}$, i.e., the set of frequencies wherein the power is non-zero. Thus, **disjoint** spectrums are orthogonal.

User Capacity. Given PSD functions $\{p_i(x)\}$ for a set of users in a communication system, the (maximum achievable rate) capacity C_i of a user i can be determined using the generalized Shannon-Hartly theorem as below. Here, we assume that the signals to be Gaussian processes, and treat interference as noise, as in prior works [128, 129, 133, 136].

$$C_i = \int_0^W \log\left(1 + \frac{p_i(x)h_{ii}(x)}{I_i(x) + N_i}\right) dx. \tag{5.2}$$

Above, h_{ii} is the channel gain, and $I_i(x)$ is the total interference on frequency x at the receiver r_i due to other users. The interference $I_i(x)$ is computed as follows.

$$I_i(x) = \sum_{j \neq i} p_j(x)h_{ji}(x).$$

Spectrum Allocation and Power Distribution (SAPD) Problem. Given a set of users $\{1, 2, \ldots, n\}$, maximum total power values P_i for each user i, noise N_i at each receiver r_i, and an available frequency spectrum $[0, W]$, the Spectrum Allocation and Power Distribution (SAPD) problem is to determine the PSD functions $\{p_i(x)\}$ for the given users such that the total (system) capacity $\sum_i C_i$ is maximized, under the constraint of Equation 5.1 (i.e., the total power used by each user i is at most

P_i). Note that determination of PSD functions also gives the allocation of spectrum across users (i.e., spectrums used by each user).

5.3 Optimal SAPD Solution Uses Maximum Power

In this section, we prove that in an optimal SAPD solution, each user uses maximum total power. We note that our result does not contradict the prior results of [137, 138] which consider a different and very restricted model. In particular, [137, 138] consider a model wherein each user uses a constant PSD across the available spectrum (i.e., each user either uses the entire spectrum with a constant PSD or remains silent). For this model, they show that to achieve maximum sum of user rates either (i) each user uses maximum power, or (ii) one of the users is silent (with the other user using maximum power). In contrast, in our model (wherein each user can use an arbitrary PSD function, and thus, an arbitrary subset of the spectrum), we show that each user must use maximum power to achieve maximum sum of user capacities. In fact, it is easy to see from our Lemma 13 that, in our model, the sum of rates achieved when one user is silent is always sub-optimal.

Theorem 19 *For frequency-selective channels, in an optimal SAPD solution, each user uses maximum power, i.e., for each user i, $\int_0^W p_i(x)dx = P_i$.*

PROOF: Let n be the number of users. Consider an optimal solution $\{p_i(x)\}$, where $p_i(x)$ is the PSD of the i^{th} user. Assume that the claim of the theorem doesn't hold, i.e., there is a user k such that

$$p' = P_k - \int_0^W p_k(x)dx > 0.$$

Below, we use p' to improve on the given solution, which will contradict our assumption that the given solution is optimal and thus, proving the theorem.

Now, for an appropriate constant ϵ (as determined later), we change the given optimal solution as follows.

- First, in the spectrum $[0, \epsilon]$, we power-off all the users, i.e., for all i, we set $p_i(x) = 0$ for $x \in [0, \epsilon]$.
- Second, we uniformly add the power p' to k's PSD in the spectrum $[0, \epsilon]$, i.e., we set $p_k(x)$ to p'/ϵ for $x \in [0, \epsilon]$.

The first change causes a decrease in the capacity of every user (including k), while the second change results in some new capacity for k. We can compute these amounts as follows.

- The decrease $\bigtriangledown_i$ in capacity of each user i (including k) due to the changes can be computed as:

$$\begin{aligned} \bigtriangledown_i = & \int_0^{\epsilon} \log\left(1 + \frac{p_i(x)h_{ii}(x)}{I_i(x) + N_i}\right) dx \\ \leq & \epsilon \log(1 + \frac{p_{max}h_{max}}{N_{min}}) \end{aligned} \tag{5.3}$$

  Above, $N_{min} = \min_i N_i$, $p_{max} = \max_{i,x} p_i(x)$, and $h_{max} = \max_{i,x} h_{ii}(x)$, where $x \in [0, \epsilon]$ and i varies over all users.

- The new capacity C'_k of user k in $[0, \epsilon]$ after the second change is:

$$\begin{aligned} C'_k &= \int_0^{\epsilon} \log\left(1 + \frac{(p'/\epsilon)h_{kk}(x)}{N_k}\right) dx \\ &\geq \epsilon \log(1 + \frac{p'h_{min}}{N_k\epsilon}) \end{aligned} \tag{5.4}$$

  Above, we have used $h_{min} = \min_x h_{kk}(x)$.

Now, the overall increase in the sum of capacities of all the users is

$$C'_k - \sum_i \bigtriangledown_i.$$

Below, we pick an ϵ that will ascertain $C'_k > \sum_i \bigtriangledown_i$. Such an ϵ will imply that the above suggested changes result in an increase in the sum of user capacities, and thus, proving the theorem. In particular, using Equation 5.3 and 5.4, we pick an ϵ such that:

$$\begin{aligned} \epsilon \log(1 + \frac{p'h_{min}}{N_k\epsilon}) &> \epsilon \sum_i \log(1 + \frac{p_{max}h_{max}}{N_{min}}) \\ \log(1 + \frac{p'h_{min}}{N_k\epsilon}) &> n \log(1 + \frac{p_{max}h_{max}}{N_{min}}) \\ 1 + \frac{p'h_{min}}{N_k\epsilon} &> (1 + \frac{p_{max}h_{max}}{N_{min}})^n \\ \epsilon &< \frac{p'h_{min}}{N_k((1 + \frac{p_{max}h_{max}}{N_{min}})^n - 1)}. \end{aligned}$$

Since the above expression is positive, there exists an ϵ for which the above suggested changes result in an increase in the sum of user capacities. This contradicts

the assumption that the original solution is optimal, and thus, proving the theorem. ■

Theorem 19 can be easily generalized to the case wherein the objective is to maximize the product of user capacities, i.e., to achieve proportional fairness. We defer the proof to Appendix C.1.

Theorem 20 *For the SAPD problem wherein the objective is to maximize the product of user capacities, the optimal solution uses maximum power for each user.* ■

5.4 Optimal SAPD Solution for Two Users in Flat Channels

In this section, we present an optimal solution for the SAPD problem for the special case of two users in flat channels. In this section, we use h_{ij} to denote the channel gain, i.e., $h_{ij}(x) = h_{ij}$ for all x. We start with an important lemma. The lemma's proof is very tedious (see Appendix C.2).

Lemma 12 *For a two user SAPD problem in flat channels, there exists an optimal solution wherein the PSD of each user is constant in the spectrum shared by the users. More formally, there exists an optimal solution such that if S_1 and S_2 are the spectrums used by the respective users, then for $x \in (S_1 \cap S_2)$, $p_i(x) = c_i$ for some constants c_i (i = 1,2).* ■

A somewhat related result from [128] states that any SAPD solution for n users can be expressed using piecewise-constant PSD's over appropriate $2n$ pieces of the available spectrum; this result requires 4 pieces for $n = 2$ users. In contrast, our above lemma implies a stronger result for an SAPD solution for two users, and is essential to our result.

Optimal SAPD Solution for Two Users. Consider a system with two users and an available spectrum $[0.W]$. The optimal SAPD solution can take three possible forms, viz., (i) the users use disjoint subspectrums, (ii) both users use the same subspectrum, (iii) the users use partially-overlapping (i.e., non-disjoint and non-equal) subspectrums. We can solve the first and the second cases optimally by using the below Lemmas 13 and 14 respectively. We defer the proofs to Appendix C.3, but Lemma 13 is a slight generalization of a result from [139] while Lemma 14 follows easily from Equation 5.2 and Lemma 12.

Lemma 13 *Consider a system of two users* $\{1, 2\}$*, and an available spectrum* $[0, W]$*. If the spectrums used by the two users are disjoint, then the maximum system capacity is*

$$W \log(1 + \frac{P_1 h_{11}}{W N_1} + \frac{P_2 h_{22}}{W N_2}),$$

and is achieved by dividing the spectrum in the ratio $N_2 P_1 h_{11} : N_1 P_2 h_{22}$*.* ■

It is easy to see from the above lemma that the system capacity obtained when one of the users is silent is always less than that obtained by the partitioning the spectrum as suggested in the lemma.

Lemma 14 *Consider a system with two users, and an available spectrum* $[0, W]$*. If the spectrums used by the two users is equal, then the maximum system capacity possible is:*

$$W \log(1 + \frac{P_1 h_{11}}{P_2 h_{21} + W N_1}) + W \log(1 + \frac{P_2 h_{22}}{P_1 h_{12} + W N_2}).$$

■

In the following paragraph, we show how to compute an optimal solution for the remaining third case, viz., wherein users use partially-overlapping subspectrums. The overall optimal SAPD solution can be then computed by taking the best of the optimal solutions for the above three cases.

Optimal Partially-Overlapping SAPD Solution. Consider an SAPD solution that is optimal among all partially-overlapping SAPD solutions. In such a solution, the available spectrum can be divided into three subspectrums S_1, S_2, and S_{12}, where S_1 and S_2 are used exclusively by user 1 and 2 respectively and S_{12} is used by both the users. We assume S_1 and S_2 to be non-zero; the cases wherein one of them is zero are easier (see Appendix C.4. Now, since the noise is white, we can assume without loss of generality, that these three subspectrums are contiguous. It is easy to see that each user 1 must use a constant PSD in S_1, and user 2 must use a constant PSD in S_2. Also, by Lemma 12, we know that each user must use a constant PSD in S_{12}, and each of the three subspectrums. Finally, by Lemma 18 (see Appendix C.3), the PSD of user 1 in S_1 must be greater than its PSD in S_{12}; similarly, the PSD of user 2 in S_2 must be greater than its PSD in S_{12}. Now, let σ_1 and σ_2 be the PSD's in S_{12} of user 1 and 2 respectively, $\sigma_1 + c_1$ be the PSD of user 1 in S_1, and $\sigma_2 + c_2$ be the PSD of user 2 in S_2. See Figure 5.1. The total system capacity can now be written as follows.

$$\begin{aligned} B \;=\; & S_1 \log(1 + \tfrac{(\sigma_1 + c1) h_{11}}{N_1}) + S_2 \log(1 + \tfrac{(\sigma_2 + c_2) h_{22}}{N_2}) + \\ & S_{12}(\log(1 + \tfrac{\sigma_1 h_{11}}{\sigma_2 + N_1}) + \log(1 + \tfrac{\sigma_2 h_{22}}{\sigma_1 + N_2})) \end{aligned}$$

To find the optimal SAPD solution of the above form, we need to essentially find values of the seven variables $S_1, S_2, S_{12}, \sigma_1, c_1, \sigma_2$ and c_2 such that the above B is maximized. We do so by determining six independent equations that must hold true for an optimal B. These six equations will help us eliminate all but one of the seven variables in B, yielding a formulation of B in terms of a single variable. We can then differentiate B with respect to the remaining variable, find the root of the differential equation equated to zero, and thus, determine the value of all the seven variables. Below, we derive the six equations (Equations 5.5 to 5.10) that relate the above seven variables. Below, S_1, S_2 and S_{12} refer to the sizes of the corresponding spectrums.

- Since W is the size of the total available spectrum, we have (by a simple application of Lemma 13):

$$W = S_1 + S_2 + S_{12} \tag{5.5}$$

- Since P_1 and P_2 are the maximum total power of users 1 and 2 respectively, by Theorem 19, we have:

$$P_1 = S_1(\sigma_1 + c_1) + S_{12}\sigma_1 \tag{5.6}$$
$$P_2 = S_2(\sigma_2 + c_2) + S_{12}\sigma_2 \tag{5.7}$$

- Note that the PSD's of the users 1 and 2 in S_1 and S_2 respectively should satisfy the values computed in Lemma 13, else the solution can be improved. Thus, we have:

$$\frac{S_1}{S_2} = \frac{N_2 P_1 h_{11}}{N_1 P_2 h_{22}} \tag{5.8}$$

- Below, we show how to derive the remaining two equations, which require some tedious analysis.

Remaining Two Equations (Eqns 5.9-5.10). Let us now consider a small portion of the spectrum called S — taken partly from S_1 and S_{12}. In an optimal solution, redistribution of power within S should not lead to an improved total capacity. Without any loss of generality, let us assume S to be of size $(w + 1)$, with $w > 0$ in the exclusive part (S_1) and 1 in the shared part (S_3). See Figure 5.1. Thus, the total power used by the first user in S is $w(c_1 + \sigma_1) + \sigma_1$. Let the optimal distribution of this total power for user 1 within S be in the ratio of $k : (1 - k)$ $(0 \leq k \leq 1)$ between the exclusive and shared parts of S. Now, the total capacity of both users

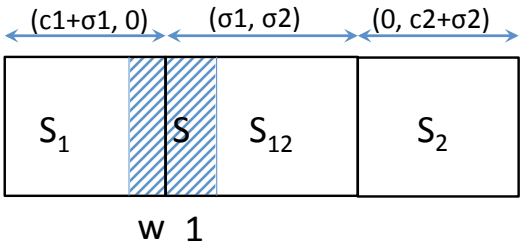


Figure 5.1: S_1 and S_2 are subspectrums used exclusively by users 1 and 2 respectively, and S_{12} is the subspectrum used by both the users. The shaded part of the spectrum is S (used to derive the final two equations) and is composed of two subspectrums of width 1 and w respectively. The top of the figures denotes the PSDs used by the users, e.g., $(c1 + \sigma_1, 0)$ signifies that the PSD values of the two users is $c1 + \sigma_1$ and 0 respectively in S_1.

in S for the above power distribution is given by:

$$\begin{aligned} C(k) = \; & w \log(1 + \tfrac{k(w(c_1+\sigma_1)+\sigma_1)h_{11}}{wN_1}) + \\ & \log(1 + \tfrac{(1-k)(w(c_1+\sigma_1)+\sigma_1)h_{11}}{\sigma_2 h_{21}+N_1}) + \\ & \log(1 + \tfrac{\sigma_2 h_{22}}{h_{12}(1-k)(w(c_1+\sigma_1)+\sigma_1)+N_2}) \end{aligned}$$

Since $C(k)$ is connected and derivable for $0 \leq k \leq 1$, $C(k)$ can be optimal only at k = 0, 1, or when $\frac{dC}{dk} = 0$. Having k = 0 or 1 will contradict our choice of S; thus, $\frac{dC(k)}{dk}$ must be zero at optimal $C(k)$. Since we started with an optimal SAPD solution, where the capacity $C(k)$ must also be optimal, the value of $\frac{dC}{dk}$ must be zero for the $k = \frac{w(c_1+\sigma_1)}{w(c_1+\sigma_1)+\sigma_1}$ (based on the distribution of power in the original solution), and this must be true for any w in $(0, x]$ where x is the size of S_1 (the exclusive part of the spectrum).

Analyzing $dC(k)/dk$. We computed $\frac{dC(k)}{dk}$ at $k = \frac{w(c_1+\sigma_1)}{w(c_1+\sigma_1)+\sigma_1}$. After simplification, the numerator in the resulting expression can be written as $w(\sigma_1+c_1)\Gamma_1+\sigma_1\Gamma_1$, where

$$\begin{aligned} \Gamma_1 = \; & h_{22}N_1^2\sigma_2 + 2h_{22}h_{11}N_1\sigma_1\sigma_2 + h_{22}h_{11}c_1N_1\sigma_2 \\ & + h_{22}N_1\sigma_2^2 - c_1N_2^2{h_{11}}^2 + c_1h_{11}h_{22}\sigma_2^2 \\ & - c_1h_{22}N_2{h_{11}}^2\sigma_2 + 2N_2h_{11}\sigma_1\sigma_2 + h_{22}N_2h_{11}\sigma_2^2 \\ & + h_{22}{h_{11}}^2\sigma_1^2\sigma_2 - c_1{h_{11}}^2\sigma_1^2 + h_{11}\sigma_1^2\sigma_2 \\ & + 2h_{22}h_{11}\sigma_1\sigma_2^2 + N_2^2h_{11}\sigma_2 - 2c_1N_2{h_{11}}^2\sigma_1 \end{aligned}$$

Since the numerator of $\frac{dC(k)}{dk}$ should be zero regardless of w's value in $(0, x]$, we must have that Γ_1 is zero. Similarly, for user 2, we must have $\Gamma_2 = 0$, where Γ_2 is

similarly defined as Γ_1. Thus, we get the fifth and sixth equations as:

$$\Gamma_1 = 0 \tag{5.9}$$
$$\Gamma_2 = 0 \tag{5.10}$$

Eliminations of Variables. It is easy to verify that the derived six equations are independent, and hence, are sufficient to eliminate six (out of the total seven) variables as desired. However, the order of elimination needs to be chosen carefully chosen to avoid getting into a unsolvable polynomial of high degree. We choose the following order of elimination. From Equation 5.5, we get:

$$S_{12} = W - S_1 - S_2$$

Substituting the above in Equation 5.6 and 5.7, and solving the resulting two equations for S_1 and S_2, we get

$$S_1 = \frac{-W\sigma_1^2 + P_2\sigma_1 + P_1c_2 - Wc_2\sigma_2}{c_1c_2 - \sigma_1\sigma_2}$$
$$S_2 = \frac{-W\sigma_2^2 + P_1\sigma_2 + P_2c_1 - Wc_1\sigma_1}{c_1c_2 - \sigma_1\sigma_2}$$

We can now write Equation 5.8 as follows.

$$\frac{\frac{-W\sigma_1^2+P_2\sigma_1+P_1c_2-Wc_2\sigma_2}{c_1c_2-\sigma_1\sigma_2}}{\frac{-W\sigma_2^2+P_1\sigma_2+P_2c_1-Wc_1\sigma_1}{c_1c_2-\sigma_1\sigma_2}} = \frac{N_2P_1h_{11}}{N_1P_2h_{22}}$$

In the above equation, we substitute c_1 and c_2 by the expressions derived from Equations 5.9 and 5.10 respectively. Note that Equations 5.9 and 5.10 are linear in c_1 and c_2 respectively, and hence, facilitating the above substitutions. After the above substitutions and tedious simplications, we actually get a fourth-degree equation in σ_1 (in terms of σ_2). Since four-degree equations have closed-form solutions, we solve the resulting equation to express σ_1 in terms of σ_2. The resulting expressions are extremely long and tedious, and hence omitted here (see [140] for details). The above allows us to express B solely in terms of σ_2. Thus, the single-variable equation $dB/d(\sigma_2) = 0$ can be solved efficiently using well-known numerical methods, since $dB/d(\sigma_2)$ is connected and derivable in σ_2 with bounded derivatives, and σ_2 has a bounded range (see Appendix C.5). Finally, as B is continuous and bounded, we can then use the roots of $dB/d(\sigma_2) = 0$ to compute the optimal B.

Note on Multiple Roots. Note that some of the intermediate equations in the above described process may not be linear, and hence may yield multiple roots. That only results in multiple expressions for B (in terms of σ_2), and hence, multiple possible

sets (but, at most 16 sets) of parameter values. We compute the total system capacity B for each of these set of values, and pick the one that yields the largest value of B.

Bibliography

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. In ACM SIGCOMM, 2008.

[2] Albert Greenberg et al. Vl2: A scalable and flexible data center network. In SIGCOMM, 2009.

[3] Lucian Popa, Sylvia Ratnasamy, Gianluca Iannaccone, Arvind Krishnamurthy, and Ion Stoica. A cost comparison of datacenter network architectures. In CoNEXT, 2010.

[4] Kai Chen et al. OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility. In Proc. NSDI, 2012.

[5] Xia Zhou et al. Mirror Mirror on the Ceiling: Flexible Wireless Links for Data Centers. In SIGCOMM, 2012.

[6] Daniel Halperin et al. Augmenting data center networks with multi-gigabit wireless links. In SIGCOMM, 2011.

[7] Guohui Wang et al. c-through: Part-time optics in data centers. In SIGCOMM, 2010.

[8] Andrew Curtis, Srinivasan Keshav, and Alejandro Lopez-Ortiz. LEGUP: Using Heterogeneity to Reduce the Cost of Data Center Network Upgrades. In CoNEXT, 2010.

[9] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking data centers randomly. In NSDI, 2012.

[10] Jayaram Mudigonda et al. Taming the Flying Cable Monster: A Topology Design and Optimization Framework for Data-Center Networks. In Proc. USENIX ATC, 2011.

[11] Charles Clos. A study of non-blocking switching networks. Bell System Technical Journal, 32, 1953.

[12] A Simpler Data Center Fabric Emerges . http://tinyurl.com/kaxpotw, .

[13] Brandon Heller et al. ElasticTree: Saving Energy in Data Center Networks. In NSDI, 2010.

[14] D. Kedar and S. Arnon. Urban Optical Wireless Communication Networks: The Main Challenges and Possible Solutions. IEEE Communications Magazine, 2004.

[15] Howard Lee Davidson et al. Data center with free-space optical communications. US Patent 8,301,028, 2012.

[16] Navid Hamedazimi, Himanshu Gupta, Vyas Sekar, and Samir Das. Patch Panels in the Sky: A Case for Free-Space Optics in Data Centers. In Proc. ACM HotNets, 2013.

[17] Kent optronics, inc. http://kentoptronics.com/switchable.html.

[18] Galvo mirrors. http://www.thorlabs.us/NewGroupPage9.cfm?ObjectGroup_ID=3770.

[19] htsim simulator. http://nrg.cs.ucl.ac.uk/mptcp/implementation.html.

[20] Mininet. http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet.

[21] Nathan Farrington et al. Helios: a hybrid electrical/optical switch architecture for modular data centers. In SIGCOMM, 2010.

[22] J. Shin et al. On the Feasibility of Completely Wireless Datacenters. In Proc. ANCS, 2012.

[23] SONAbeam FSOs. http://www.fsona.com/product.php?sec=2500e.

[24] Luka Mustafa and Benn Thomsen. Reintroducing free-space optical technology to community wireless networks. In Proc. AMICS, 2013.

[25] OpenGear out of band management. http://tinyurl.com/n773hg3.

[26] N. McKeown and others. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008.

[27] A. Curtis et al. DevoFlow: Scaling Flow Management for High-Performance Networks. In SIGCOMM, 2011.

[28] M. Al-Fares et al. Hedera: Dynamic Flow Scheduling for Data Center Networks. In NSDI, 2010.

[29] Lightpointe flightstrata g optical gigabit link. http://tinyurl.com/k86o2vh.

[30] K. Yoshida et al. Assisted focus adjustment for free space optics system coupling single-mode optical fibers. Industrial Electronics, IEEE Transactions on, 60(11), 2013.

[31] O. Svelto. Principles of Lasers. Plenum Press, New York, fourth edition, 1998.

[32] Single-fiber sfp. http://www.championone.net/products/transceivers/sfp/single-fiber-single-wavelength/.

[33] Julian Turner. Effects of data center vibration on compute system performance. In Proc. SustainIT, 2010.

[34] Paul F. McManamon et al. A review of phased array steering for narrow-band electrooptical systems. Proceedings of the IEEE, 2009.

[35] Le Li. CEO, KentOptronics. Personal Communication.

[36] Xinyu laser products. http://www.xinyulaser.com/index.asp.

[37] Mems scanning mirror. http://www.lemoptix.com/technology/mems-scanning-mirrors.

[38] B.W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. The Bell Systems Technical Journal, 49(2), 1970.

[39] Anupam Gupta and Jochen Konemannb. Approximation algorithms for network design: A survey. Surveys in Operations Research and Management Science, 16, 2011.

[40] Burkhard Monien and Robert Preis. Upper bounds on the bisection width of 3- and 4-regular graphs. Journal of Discrete Algorithms, 4, 2006.

[41] J. Friedman. On the second eigenvalue and random walks in random d-regular graphs. Combinatorica, 11(4), 1991.

[42] Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker. Abstractions for network update. In Proc. SIGCOMM, 2012.

[43] R. Mahajan and R. Wattenhofer. On consistent updates in software defined networks (extended version). In ACM HotNets, 2013.

[44] R. Wang, D. Butnariu, and J. Rexford. Openflow-based server load balancing gone wild. In Proc. Hot-ICE, 2011.

[45] `http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/COM_WP_10GBASE_T_Ecosystem_US4.pdf,.`

[46] C. Guo et al. Bcube: A high performance, server-centric network architecture for modular data centers. In ACM SIGCOMM, 2009.

[47] G. Porter et al. Integrating microsecond circuit switching into the data center. In ACM SIGCOMM, 2013.

[48] C.Y. Hong et al. Achieving high utilization with software-driven WAN. In ACM SIGCOMM, 2013.

[49] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". In MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pages 271–278. ACM, 1999. URL `citeseer.nj.nec.com/kahn99next.html`.

[50] Alexandros G. Dimakis, Vinod Prabhakaran, and Kannan Ramchandran. Decentralized erasure codes for distributed networked storage. IEEE/ACM Trans. Netw., 14(SI):2809–2816, 2006. ISSN 1063-6692.

[51] Julio C. Navas and Tomasz Imielinski. Geocast—geographic addressing and routing. In MobiCom '97: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking, pages 66–76, New York, NY, USA, 1997. ACM. ISBN 0-89791-988-2.

[52] Alexandros G. Dimakis and Kannan Ramchandran. Network Coding for Distributed Storage in Wireless Networks. Springer, 2008.

[53] Bla Bollobs. Random graphs. Cambridge studies in advanced mathematics 73. Cambridge University Press, Cambridge ; New York, 2nd edition, 2001.

[54] Bla Bollobs. Modern graph theory. Graduate texts in mathematics 184. Springer, New York, 1998.

[55] Szymon Acedanski, Supratim Deb, Muriel Mdard, and Ralf Koetter. How good is random linear coding based distributed networked storage. In In NetCod, 2005.

[56] Alexandros G. Dimakis, Vinod Prabhakaran, and Kannan Ramchandran. Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes. In IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks, page 15, Piscataway, NJ, USA, 2005. IEEE Press. ISBN 0-7803-9202-7.

[57] Yunfeng Lin, Ben Liang, and Baochun Li. Data persistence in large-scale sensor networks with decentralized fountain codes. In Proc. INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pages 1658–1666, 6–12 May 2007. doi: 10.1109/INFCOM.2007.194.

[58] Yunfeng Lin, Ben Liang, and Baochun Li. Geometric random linear codes in sensor networks. In Proc. IEEE International Conference on Communications ICC '08, pages 2298–2303, 19–23 May 2008. doi: 10.1109/ICC.2008.438.

[59] S. A. Aly, Zhenning Kong, and E. Soljanin. Fountain codes based distributed storage algorithms for large-scale wireless sensor networks. In Proc. International Conference on Information Processing in Sensor Networks IPSN '08, pages 171–182, 22–24 April 2008. doi: 10.1109/IPSN.2008.64.

[60] Growth codes: maximizing sensor network data persistence, volume 36, New York, NY, USA, 2006. ACM. doi: http://doi.acm.org/10.1145/1151659.1159943.

[61] L. Lovasz. Matching Theory (North-Holland mathematics studies). Elsevier Science Ltd, 1986.

[62] Mark De Berg. Computational Geometry: Algorithms and Applications. Springer-Verlag, New York, NY, USA, 2008. ISBN 9783540779735.

[63] Fundamentals of applied probability and random processes. Oliver Chukwudi Ibe. Academic Press, 2005. ISBN 0120885085.

[64] U. S. R. Murty John Adrian Bondy. Graph theory. Springer, 2007. ISBN 1846289696.

[65] Michael R. Garey and David S. Johnson. Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA, 1990. ISBN 0716710455.

[66] Cisco visual networking index: Global mobile data traffic forecast update, 2009-2014. tinyurl.com/d887fkswww.

[67] The impact of mobile computers and smartphones on cdma 2000 networks. www.signalsreseaerch.com, Jan 2011.

[68] H. Claussen, L. T. W. Ho, and L. G. Samuel. An overview of the femtocell concept. Bell Lab. Tech. J., 13, 2008.

[69] Kyunghan Lee et al. Mobile data offloading: how much can WiFi deliver? In ACM SIGCOMM 2010, 2010.

[70] M. M. Buddhikot and K. Ryan. Spectrum management in coordinated dynamic spectrum access based cellular networks. In DySPAN, 2005.

[71] Verizon to introduce tiered pricing for iPhone.

[72] NY Times. Limiting the unlimited. Issue of March 2, 2012.

[73] Utpal Paul et al. Understanding traffic dynamics in cellular data networks. In INFOCOM, april 2011. doi: 10.1109/INFCOM.2011.5935313.

[74] Chunyi Peng et al. Traffic-driven power saving in operational 3g cellular networks. In Proc. ACM Mobicom, 2011.

[75] G. Fusco et al. Finding green spots and turning the spectrum dial: Novel techniques for green mobile wireless networks. In IEEE DySPAN, 2011.

[76] N.D. Tripathi, J.H. Reed, and H.F. VanLandinoham. Handoff in cellular systems. IEEE Personal Communications, 5(6):26 –37, 1998.

[77] Nasif Ekiz et al. An overview of handoff techniques in cellular networks. World Academy of Sci., Eng. and Technology, 2005.

[78] J. Leung. Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, 2004.

[79] M. L. Pinedo. Scheduling: Theory, Algorithms, and Systems. Springer Publishing Company, Incorporated, 3rd edition, 2008. ISBN 0387789340, 9780387789347.

[80] W. Zhao, K. Ramamritham, and J. A. Stankovic. Preemptive scheduling under time and resource constraints. IEEE Trans. on Computers, 1987.

[81] Z. Liu and E. Sanlaville. Preemptive scheduling with variable profile, precedence constraints and due dates. Discrete Applied Math., 1995.

[82] T. Gonzalez and S. Sartaj. Open shop scheduling to minimize finish time. J. ACM, 1976.

[83] E. Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Annals of Operations Research, 26, 1990.

[84] J. Labetoulle et al. Preemptive scheduling of uniform machines subject to release dates. Progress in combinatorial optimization, 1984.

[85] K.R. Baker, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints. Operations Research, 1983.

[86] P. Baptiste. An $o(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Operations Research Letters, 1999.

[87] Grsel A. Ser et al. Minimizing the number of tardy jobs in identical machine scheduling. Computers & Industrial Engineering, 25(1-4), 1993.

[88] Bo Chen et al. An optimal algorithm for preemptive on-line scheduling. Operations Research Letters, 18(3):127 – 131, 1995.

[89] B. DasGupta and M. Palis. Online real-time preemptive scheduling of jobs with deadlines. In LNCS. Springer, 2000.

[90] Sajal K. Das et al. A dynamic load balancing strategy for channel assignment using selective borrowing in cellular mobile environment. Wirel. Netw., 3, Oct. 1997. ISSN 1022-0038. doi: http://dx.doi.org/10.1023/A:1019181923135. URL http://dx.doi.org/10.1023/A:1019181923135.

[91] D. Everitt and D. Manfield. Performance analysis of cellular mobile communication systems with dynamic channel assignment. IEEE J. on Selected Areas in Communications, 1989.

[92] J. V. Leeuwaarden et al. Load balancing in cellular networks using first policy iteration. Technical report, Helsinki University of Technology, 2001.

[93] Lin Du et al. Intelligent cellular network load balancing using a cooperative negotiation approach. In IEEE WCNC, volume 3, 2003.

[94] Samir Das et al. Dynamic load balancing through coordinated scheduling in packet data systems. In INFOCOM, 2003.

[95] J.-W. Lee et al. Joint resource allocation and base-station assignment for the downlink in cdma networks. IEEE/ACM Transactions on Networking, 14 (1), feb. 2006. ISSN 1063-6692. doi: 10.1109/TNET.2005.863480.

[96] O.K. Tonguz and E. Yanmaz. The mathematical theory of dynamic load balancing in cellular networks. IEEE TMC, 2008.

[97] Paz Carmi, Matthew J. Katz, and Nissan Lev-Tov. Covering points by unit disks of fixed location. In Proceedings of the 18th international conference on Algorithms and computation, ISAAC'07, pages 644–655, Berlin, Heidelberg, 2007. Springer-Verlag.

[98] Himanshu Gupta et al. Connected sensor cover: self-organization of sensor networks for efficient query execution. IEEE/ACM Trans. Netw., 14, 2006.

[99] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2010–2015. 2011.

[100] Mustafa Y. Arslan, Jongwon Yoon, Karthikeyan Sundaresan, Srikanth V. Krishnamurthy, and Suman Banerjee. FERMI: A FEmtocell Resource Management system for Interference mitigation in ofdma networks. In MobiCom, 2011.

[101] John Paul M. Torregoza, Rentsen Enkhbat, and Won-Joo Hwang. Joint power control, base station assignment, and channel assignment in cognitive femtocell networks. EURASIP Journal on Wireless Communications and Networking, 2010(285714), 2010. doi: http://dx.doi.org/10.1155/2010/285714.

[102] Piyush Gupta and Panganamala Ramana Kumar. The capacity of wireless networks. IEEE Transactions on Information Theory, 46(2):388–404, 2000.

[103] Ji-Hoon Yun and Kang G. Shin. CTRL: A self-organizing femtocell management architecture for co-channel deployment. In MobiCom, 2010.

[104] Karthikeyan Sundaresan and Sampath Rangarajan. Efficient resource management in OFDMA femto cells. In ACM MobiHoc, pages 33–42, 2009. doi: http://dx.doi.org/10.1145/1530748.1530754.

[105] Heui-Chang Lee, Dong-Chan Oh, and Yong-Hwan Lee. Mitigation of inter-femtocell interference with adaptive fractional frequency reuse. In ICC, 2010. doi: http://dx.doi.org/10.1109/ICC.2010.5502298.

[106] Poongup Lee, Taeyoung Lee, Jangkeun Jeong, and Jitae Shin. Interference management in LTE femtocell systems using fractional frequency reuse. In ICACT, 2010.

[107] Holger Claussen and Florian Pivit. Femtocell coverage optimization using switched multi-element antennas. In IEEE ICC, 2009.

[108] Han-Shin Jo, Cheol Mun, June Moon, and Jong-Gwan Yook. Self-optimized coverage coordination in femtocell networks. IEEE Transactions on Wireless Communications, 9(10):2977–2982, 2010.

[109] Han-Shin Jo, Cheol Mun, June Moon, and Jong-Gwan Yook. Interference mitigation using uplink power control for two-tier femtocell networks. IEEE Transactions on Wireless Communications, 8(10):4906–4910, 2009. doi: http://dx.doi.org/10.1109/TWC.2009.080457.

[110] Holger Claussen, Florian Pivit, and Lester T. W. Ho. Self-optimization of femtocell coverage to minimize the increase in core network mobility signalling. Bell Labs Technical Journal, 14(2):155–184, 2009.

[111] Vikram Chandrasekhar, Jeffrey G. Andrews, Tarik Muharemovic, Zukang Shen, and Alan Gatherer. Power control in two-tier femtocell networks. Transactions on Wireless Communications, 8(8):4316–4328, 2009.

[112] Chirag Patel, Lenny Grokop, Vinay Chande, Varun Khaitan, Mehmet Yavuz, and Sanjiv Nanda. Femtocell and beacon transmit power self-calibration. Technical report, Qualcomm, 2010.

[113] Mehmet Yavuz, Farhad Meshkati, Sanjiv Nanda, Akhilesh Pokhariyal, Nick Johnson, Balaji Raghothaman, and Andy Richardson. Interference management and performance analysis of UMTS/HSPA+ femtocells. IEEE Communications Magazine, 47(9):102–109, 2009.

[114] Holger Claussen, Lester T. W. Ho, and Louis G. Samuel. Self-optimization of coverage for femtocell deployments. In WTS, pages 278–285, 2008. doi: http://dx.doi.org/10.1109/WTS.2008.4547576.

[115] Imran Ashraf, Holger Claussen, and Lester T.W. Ho. Distributed radio coverage optimization in enterprise femtocell networks. In ICC, 2010. doi: http://dx.doi.org/10.1109/ICC.2010.5502072.

[116] Lester T.W. Ho, Imran Ashraf, and Holger Claussen. Evolving femtocell coverage optimization algorithms using genetic programming. In IEEE PIMRC, pages 2132–2136, 2009. doi: http://dx.doi.org/10.1109/PIMRC.2009.5450062.

[117] Goutam K. Audhya, Koushik Sinha, Sasthi C. Ghosh, and Bhabani P. Sinha. A survey on the channel assignment problem in wireless networks. Wireless Comm. and Mobile Computing, 11(5):583–609, 2011. doi: http://dx.doi.org/10.1002/wcm.898.

[118] Mikael Fallgren. On the complexity of maximizing the minimum shannon capacity in wireless networks by joint channel assignment and power allocation. In IWQoS, 2010. doi: http://dx.doi.org/10.1109/IWQoS.2010.5542766.

[119] Giordano Fusco, Milind Buddhikot, Himanshu Gupta, and Sivarama Venkatesan. Finding green spots and turning the spectrum dial: Novel techniques for green mobile wireless networks. In DySPAN, pages 613–617, 2011. doi: http://dx.doi.org/10.1109/DYSPAN.2011.5936255.

[120] Chunyi Peng, Suk-Bok Lee, Songwu Lu, Haiyun Luo, and Hewu Li. Traffic-driven power saving in operational 3G cellular networks. In Mobicom, 2011. doi: http://doi.acm.org/10.1145/2030613.2030628.

[121] Frédéric Havet, Ross J. Kang, and Jean-Sébastien Sereni. Improper colouring of unit disk graphs. Networks, 54(3):150–164, 2009. doi: http://dx.doi.org/10.1002/net.20318.

[122] Giusi Alfano, Michele Garetto, and Emilio Leonardi. Capacity scaling of wireless networks with inhomogeneous node density: upper bounds. IEEE J.Sel. A. Commun., 27:1147–1157, September 2009. ISSN 0733-8716. doi: 10.1109/JSAC.2009.090911.

[123] 3rd Generation Partnership Project. Technical specification group radio access networks; 3G home nodeb study item technical report (release 8). Technical Report 3GPP TR 25.820 V8.0.0 (2008-03), 3GPP, 2008.

[124] Carl Friedrich Gauss. Besprechung des Buchs von L. A. Seeber: Intersuchungen uber die eigenschaften der positiven ternaren quadratischen formen usw. Gottingsche Gelehrte Anzeigen (1831, July 9), 2:188–196, 1876.

[125] L. Fejes Tóth. Uber einen geometrischen satz. Math. Z., 46:78–83, 1940.

[126] T. S. Rappaport. Wireless Communications Principles and Practices. Prentice-Hall, 2002.

[127] Y. Zhao and G.J. Pottie. Optimal spectrum management in multiuser interference channels. In ISIT, 2009.

[128] Raúl H. Etkin, Abhay Parekh, and David Tse. Spectrum sharing for unlicensed bands. IEEE JSAC, 25(3), 2007.

[129] S. Hayashi and Z.-Q. Luo. Spectrum management for interference-limited multiuser communication systems. IEEE Trans. Inf. Theor., 2009.

[130] W. Yu, G. Ginis, and J.M. Cioffi. Distributed multiuser power control for digital subscriber lines. IEEE JSAC, 2002.

[131] W. Yu and R. Lui. Dual methods for nonconvex spectrum optimization of multicarrier systems. IEEE Trans. on Comm., 2006.

[132] R. Cendrillon, W. Yu, M. Moonen, J. Verlinden, and T. Bostoen. Optimal multiuser spectrum balancing for digital subscriber lines. Communications, IEEE Transactions on, 54(5):922–933, 2006.

[133] Z.-Q. Luo and S. Zhang. Dynamic spectrum management: Complexity and duality. IEEE J. of Selected Topics in Signal Processing, 2008.

[134] S. R. Bhaskaran et al. Maximizing the sum rate in symmetric networks of interfering links. IEEE Trans. on Information Theory, 2010.

[135] H. Shen et al. Optimal spectrum allocation in gaussian interference networks. In Asilomar Conf. on Signals, Systems and Computers, 2008.

[136] Z.-Q. Luo and S. Zhang. Duality gap estimation and polynomial time approximation for optimal spectrum management. ACM Trans. Sig. Proc., 2009.

[137] M. Charafeddine and A. Paulraj. Maximum sum rates via analysis of 2-user interference channel achievable rates region. In CISS. IEEE, 2009.

[138] A. Gjendemsjo et al. Optimal power allocation and scheduling for two-cell capacity maximization. In Intl. Symp. on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006.

[139] R. Gummadi et al. Interference avoidance and control. In ACM HotNets, 2008.

[140] Matlab source codes for omitted mathematical details. http://tinyurl.com/b6yhdal.

[141] V. G. Vizing. On an estimate of the chromatic class of a p-graph. Diskret. Analiz. (in Russian), 3:25–30, 1965.

[142] Scott Bloom, Eric Korevaar, John Schuster, and Heinz Willebrand. Understanding the performance of free-space optics [invited]. Journal of optical Networking, 2(6):178–200, 2003.

[143] Renka, single mode fiber specifications. http://www.renka.com/products/SmartLITE%20Cable/DEC00%20SM%20OFC%20RENKA.pdf.

[144] David P. Dailey. Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete. Discrete Mathematics, 30(3):289–293, 1980. doi: http://dx.doi.org/10.1016/0012-365X(80)90236-8.

Appendices

Appendix A

A.1 Evacuation Time of Flexible(f)

Here, we formally prove the claim mentioned in $S1.2.1$, with regards to the total evacuation time of the Flexible(f) architecture.

Theorem 21 *Consider an inter-rack traffic matrix D, wherein each entry D_{ij} represents the total traffic demand from source-rack i to the destination rack j. Assume that each unit of data takes a unit time to be evacuated over any link.*

The total evacuation time for the Flexible(f) architecture, when $f \leq l$ is at most

$$\frac{l}{f}M + \frac{1}{f}\max_{ij}(D_{ij} + D_{ji}) \approx \frac{l}{f}M$$

units, where M is the evacuation time for FBB (full-bisection network) and l is the number of machines per rack. For $f > l$, the evacuation time is same as that for $f = l$. Moreover, the number of reconfigurations needed to achieve the above evacuation time is less than the number of non-zero entries in the given traffic matrix.

PROOF: We start with considering the simple case of $f = 1$. Consider an undirected multigraph G over the set of racks, where the number of edges between a rack i and j is $D_{ij} + D_{ji}$. Each undirected edge in G represents a unit-demand between the pair of racks. Now, any matching in G represents a set of packets that can be evacuated in a unit time since: (i) a matching in G represents a valid "wiring" of the Flexible(f) architecture, and (ii) each link (i, j) can evacuate one unit of data from i to j or from j to i in a unit time.

Thus, the total time to evacuate the given traffic demand matrix in CorelessFlexible (1) is equal to the minimum number of matchings the given graph G's edges can be partitioned into (or equivalently, the minimum number to colors needed to color the edges of G such that no two edges incident on a node have the same color).

Now, using the classical result from Vizing [141], G can be decomposed into

$$\max_{ij}(\Delta_i + \mu_{ij})$$

matchings, where Δ_i is the degree of a node i and μ_{ij} is the number of edges between the nodes i and j in G. The extension to a general f is straightforward. Essentially, when $1 \leq f \leq l$, the Flexible(f) architecture can evacuate f matchings of G in one unit time. Thus, the evacuation time for a general $f \leq l$ is:

$$\frac{1}{f}\max_{ij}(\Delta_i + \mu_{ij}).$$

Now, note that the total evacuation time of FBB is exactly $\frac{1}{l}\max_i \Delta_i$, since it can send or receive l units of data from a rack in a unit time. Thus, the total evacuation time of $Flexible$(f) is

$$\frac{l}{f}M + \frac{1}{f}\max_{ij}(D_{ij} + D_{ji}).$$

We note that M is expected to be much larger than $\frac{1}{f}\max_{ij}(D_{ij} + dji)$ (especially, for a large number of racks) and thus, total evacuation time of Flexible(f) is $\approx M(l/f)$.

To bound the number of reconfiguration, we can show that the number of different matchings in the partitioning of G above can be bounded by the number of non-zero entries in the traffic matrix. We skip the details here. ■

A.2 FSO Link for Long Distances

Here, we briefly corroborate the claim that our FSO link design should work for distances up to 100m and more. Recall from Figure 1.3 the basic schematic of our FSO link design. Here, the light beam emanating from an optical fiber is first collimated at the transmitter using the collimation lens, and then focused into an optical fiber at the receiver using another lens. We argue below that, with our design, the loss of power due to beam divergence and air-attenuation is expected to be minimal for longer distances (say, up to 100m), and hence, the FSO link should work at such distances.

- We note that the collimation lens in our design has been chosen appropriately to ensure the beam remains collimated for a distance of up to 100m with a width of roughly 4mm. Since the diameter of our receiving lens is greater

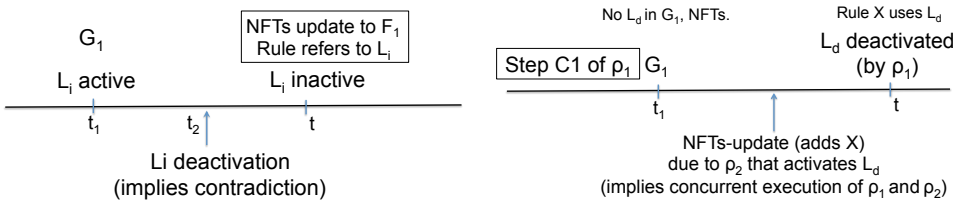


Figure A.1: Correctness proof for avoidance of black holes.

than 25mm, a 4mm wide beam is fully "gathered" by the receiving lens. Thus, we can easily ignore the power loss resulting from beam divergence.

- Now, we show that the signal attenuation due to scattering and reflection in the air is also negligible. Note that, while attenuation of an infra-red laser beam can be high in fog or rain, it is minimal in clear whether. In particular, [142] estimates the value of attenuation-coefficient for a beam of 1310nm wavelength that we use in our design to be 3.5dBm/km. This suggests a power loss of 0.35dBm over a 100m link, which is much lower than the power-losstolerance of the detectors at SFPs. For comparison, a typical commercial optical cable [143] has an attenuation of 0.38dBm/km which results in a power loss of 3.8dBm over 10km, the range of 10GBASE-LR SFP's used in our system. Further, we note that the above attenuation-coefficient is for outdoor environments, and the indoor attenuation-coefficient (which is more relevant in our context) is much lower.

A.3 Correctness of Data Translation Scheme

Here, we formally prove that our data translation scheme of $S1.6.3$ ensures the desired properties. We start with an observation.

Observation 1 *At any instant, the set of edges in $\mathcal{G}$is a subset of the actual set of active links in the networks.*[1] □

Theorem 22 *The overall scheme of $S1.6.3$ ensures that (a) there are no black holes (i.e., all rules in NFTs refer to active links), (b) the network remains connected, and*

[1]This is true because we only allow non-conflicting reconfigurations to execute concurrently; we skip the tedious details.

(c) the packet latency is bounded by $(2x + z)$, where x is the maximum packet latency in a fixed topology and z is the NFTs-update time. The above claim holds irrespective of how the NFTs are updated across the network; i.e., we do not require atomic updates.

PROOF: We start with stating a couple of assumptions we have implicitly made in our translation scheme: (i) In steps C1 and C3, the update to $\mathcal{G}$ occurs before the NFTs-updates, and (ii) the NFTs-updates (in C1 or C3, from different concurrent reconfigurations) finish in the same order as the corresponding updates to $\mathcal{G}$.[2]

Avoidance of Black Holes. Consider the first instant t when a black hole is created in the network. There are only two possible events that can induce a black hole, viz., (i) A new forwarding rule that refers to an inactive link is added, or (ii) A link that is being used in a forwarding rule is deactivated. We consider each of these cases below.

1. Lets say a rule that refers to an inactive link l_i is added at time t during an NFTs-update to $\mathcal{F}_1$. Let the corresponding update of $\mathcal{G}$ variable to $\mathcal{G}_1$ to have occurred at time t_1 ($< t$). Now, $\mathcal{G}_1$ must include the edge/link l_i (since $\mathcal{F}_1$, which is based on $\mathcal{G}_1$, has a rule that refers to it). By Observation 1, l_i is active at t_1. Thus, l_i must have been deactivated at time t_2 between t_1 and t (by a reconfiguration say ρ_2). See Figure A.1(a). This is impossible because it requires the corresponding step C1 of ρ_2 to have occurred before t_2, which yields one of the following two contradictions, viz., (1) If step C1 of ρ_2 occurs between t_1 and t_2, then updates to $\mathcal{G}$(from ρ_1 and ρ_2) occur in reverse order of corresponding NFTs-updates, or (2) If any part of step C1 ρ_2 (i.e, update to $\mathcal{G}$) occurs before t_1, then two conflicting reconfigurations (i.e., ρ_2 and another that activates l_i before t_1) have executed concurrently.

2. Lets say, at time t, a link l_d, that is being used in an existing forwarding-rule X, is deactivated. Let the deactivation be due to a reconfiguration ρ_1, and the corresponding step C1 to have finished at time t_1 ($< t$) and have updated the variable $\mathcal{G}$ to $\mathcal{G}_1$ and the NFTs to $\mathcal{F}_1$. Thus, $\mathcal{G}_1$ must not contain l_d and $\mathcal{F}_1$ must not refer to l_d. Thus, there must have been an NFTs-update t_1 and t that added the rule X, and this must be due to a reconfiguration ρ_2 that activates l_d. See Figure A.1(b). Now, ρ_1 and ρ_2 are obviously conflicting, but have executed concurrently. This is contradictory to our scheme.

We note that the above argument does not assume a minimum time-interval between updates, and hence, avoidance of black holes is guaranteed even if NFTs-updates are allowed to execute concurrently.

[2]This is due to NFTs-changes being sent to the network switches in the form of "deltas," and hence are received (and thus, finished) at each switch in a deterministic order.

Network Connectivity. Note that $\mathcal{G}$ is guaranteed to be connected at all times. Thus, by Observation 1, the underlying network topology must be connected at all times.

Bounded Packet Latency. The minimum time-interval of x between two updates, where x is the maximum packet latency in a fixed topology, ensures that each packet encounters at most two NFTs during its flight. Thus, the packet latency is bounded by $2x + z$. ■

Appendix B

B.1 NP-Hardness Proofs

In this section, we prove Theorem 14 which states that both the MTC and MMC problems are NP-hard.

PROOF: We start by defining the decision version of both problems.

`Decision-MTC`: Given a set of n femtocells (each associated with a location, coverage region, and a maximum transmission power), a constant C, and a number of available channels, is there an assignment of channels and powers to the femtocells so that the sum of all capacities is nC?

`Decision-MMC`: Given a set of femtocells (each associated with a location, coverage region, and a maximum transmission power), a minimum capacity requirement C_{min}, and a number of available channels, is there an assignment of channels and powers to the femtocells so that each femtocell f's capacity $C(f)$ is at least C_{min}?

We reduce from the problem of 3-coloring a planar graph of degree 4, which is proved to be NP-complete in [144]:

`3-Coloring`: given a planar graph of degree 4, is it possible to color all nodes with 3 colors such that no two adjacent nodes have the same color?

The construction is as follows. We are going to model nodes and edges with a combination of femtocells. In particular, each node of the graph is represented with

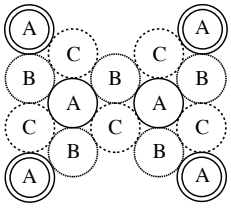


Figure B.1: Construction of femtocells corresponding to a node of the graph.

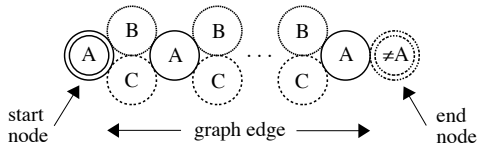


Figure B.2: Chain of femtocells representing edges of the graph.

femtocells placed as in Figure B.1, and each edge with a chain as in Figure B.2 (we assume that the graph is drown with enough space to accommodate these constructions). The four corners of the graph-node gadget are attached at one end of the edge gadget (femtocells with double border in Figures B.1 and B.2).

The parameters are chosen in such a way that, in an optimal solution, neighbor femtocells will receive different channels. A possible choice of parameters is the following. We assume there are exactly 3 channels. Femtocells' coverage regions are disks of radius $r = 1$. The bandwidth is set to $B = 1$ MHz, and the path loss parameter is set to $\alpha = 5$. The maximum power is set to $\overline{P} = 1$ W, and the noise is set to $N = 1/100$ W.

The following two lemmas can be proved by simply computing the capacity using Shannon's formula.

Lemma 15 *If all adjacent femtocells receive a different channel, then each femtocell has a capacity* $4 \leq C \leq 5.6$ *Mbps.*

Lemma 16 *If two adjacent femtocells receive the same channel, then both these femtocells have a capacity* $C \leq 1$ *Mbps.*

The purpose of our construction is to impose that if two graph nodes are adjacent in the input graph, then the corresponding femtocells receive different channels. In particular, the color of each graph node corresponds to the channel assigned to the femtocells marked with "A" in Figure B.1. The femtocells marked with "B" and "C" can assume any of the other 2 colors. By the way edges are constructed (see Figure B.2), the start and the end nodes get different channels, if all adjacent femtocells receive a different channel.

By the two lemmas above, a solution to the `Decision-MTC` with total capacity at least $4n$ guarantees that all adjacent femtocells receive different channels. In fact, the total capacity would decrease by at least 3 if two adjacent femtocells receive the same channel, or by at least 4 if one femtocell does not receive any channel. Also, note that if one femtocell uses less than the maximum power, its capacity is also decreased. Hence, if we can find a solution to `Decision-MTC` with $C = 4$ (i.e., total capacity $4n$), then we also get a solution to the `3-Coloring` problem.

A similar argument can be made for the `Decision-MMC` problem. If there is a solution to `Decision-MMC` with $C_{min} = 4$, then there is a solution to the `3-Coloring` problem. ■

B.2 MIQP

In our simulations, we computed an upper bound on the optimum using a Mixed Integer Quadratic Program (MIQP). In this section, we show how this MIQP is constructed.

Using Shannon's formula, the capacity of a femtocell f can be expressed as

$$C_f = \log_2(1 + SINR_f) = \log_2(1 + Pr^{-\alpha}/(I_f + N)),$$

where P is the power, r is the distance from the femtocell's center to the farthest point in any coverage region, α is the path loss exponent, I_f is the (maximum) interference perceived in any point of the coverage region of f, and N is the noise.

Using this formula, we can write the following non-linear mixed integer program for the MTC problem:

$$\begin{aligned}
\max \quad & C \\
\text{s.t.} \quad & C \leq \sum_f \log_2(1 + Pr^{-\alpha}/(I_f + N)) && \text{(B.1)} \\
& \sum_c a_{cf} = 1, \quad \forall f && \text{(B.2)} \\
& a_{cf} + a_{cg} - 1 \leq s_{cfg}, \quad \forall c, f, g && \text{(B.3)} \\
& s_{cfg} \leq a_{cf}, \quad \forall c, f, g && \text{(B.4)} \\
& s_{cfg} = s_{cgf}, \quad \forall c, f, g, \quad f < g && \text{(B.5)} \\
& I_f = \sum_{g \neq f} \sum_c s_{cfg} P(d_{fg} + r)^{-\alpha}, \quad \forall f && \text{(B.6)}
\end{aligned}$$

where $d_{f,g}$ = distance between femtocells f and g, and we used the following variables:

$C \geq 0$	is total capacity;
$a_{cf} \in \{0, 1\}$	represents whether channel c is assigned to femtocell f;
$s_{cfg} \in \{0, 1\}$	represents whether channel c is "shared" by femtocells f and g;
$I_f \geq 0$	is the interference of femtocell f.

The meaning of the constraints in the following:

(B.2) each femtocell has exactly one channel;

(B.3) if two femtocells have the same channel, then they "share" a channel;

(B.4) a femtocell can "share" a channel only if it has that channel;

(B.5) channels are symmetric;

(B.6) interference formula.

These problems are non-linear because of the logarithm in Equation B.1. The authors of [101] handle them by using a branch and bound technique in conjunction with their own solver. However, we use a slightly different technique based on Taylor series that allows us to use a standard solver. We now show the details of our substitution.

First, we observe that for femtocell f

$$\begin{aligned}
&\log_2(1 + SINR_f) = \\
&= \log\big(1 + Pr^{-\alpha}/(I_f + N)\big) \,/\, \log 2 = \\
&= \log\big((I_f + N + Pr^{-\alpha})/(I_f + N)\big) \,/\, \log 2 = \\
&= \log\big(\varepsilon(I_f + N + Pr^{-\alpha})/(\varepsilon(I_f + N))\big) \,/\, \log 2 = \\
&= \big(\log\big(\varepsilon(I_f + N + Pr^{-\alpha})\big) - \log(\varepsilon(I_f + N))\big) \,/\, \log 2,
\end{aligned}$$

where for the sake of applying Taylor expansion we multiplied both the numerator and the denominator for an opportunely chosen small constant ε.

Then, using the Taylor expansion of $\log(1 + x)$, we get

$$\log\big(\varepsilon(I_f + N + Pr^{-\alpha})\big) - \log(\varepsilon(I_f + N)) =$$

$$
\begin{aligned}
&= \big(\varepsilon(I_f + N + Pr^{-\alpha}) - 1\big) - \big(\varepsilon(I_f + N + Pr^{-\alpha}) - 1\big)^2/2 \\
&\quad + \big(\varepsilon(I_f + N + Pr^{-\alpha}) - 1\big)^3/3 - O\big((\varepsilon(I_f + N + Pr^{-\alpha}) - 1)^4\big) \\
&\quad - (\varepsilon(I_f + N) - 1) + (\varepsilon(I_f + N) - 1)^2/2 \\
&\quad - (\varepsilon(I_f + N) - 1)^3/3 + O\big((\varepsilon(I_f + N) - 1)^4\big) = \\
&= \varepsilon P r^{-\alpha} - \varepsilon^2 P^2 r^{-2\alpha}/2 + \varepsilon^3 P^3 r^{-3\alpha}/3 \\
&\quad + \varepsilon P r^{-\alpha}(\varepsilon(I_f + N) - 1)(\varepsilon(Pr^{-\alpha} + I_f + N) - 2) \\
&\quad + O\big((\varepsilon(I_f + N) - 1)^4 - (\varepsilon(I_f + N + Pr^{-\alpha}) - 1)^4\big) \\
&= \varepsilon^3 P r^{-\alpha} I_f^2 + \varepsilon^3 P r^{-\alpha}(Pr^{-\alpha} + 2N - 3/\varepsilon) I_f + \varphi(Pr^{-\alpha}, N, \varepsilon) \\
&\quad - O(K)
\end{aligned}
$$

where in the last step we assumed that P, r, α, and N are constants and they contribute to the constant term $\varphi()$, and we set $K = (\varepsilon(I_f + N) - 1)^4 - (\varepsilon(I_f + N + Pr^{-\alpha}) - 1)^4$.

Now, we observe that K is negative since its second term is bigger than the first one. Hence, if we do not include the error term $O(K)$ in the constraints of the MIQP, we obtain a solution that is larger than the true optimum (i.e., an upper bound on the optimum).

This allows to write the MIQP for MTC as follows

$$
\begin{aligned}
\max \quad & C \\
\text{s.t.} \quad & C \le \sum_f \big(Pr^{-\alpha} I_f^2 + Pr^{-\alpha}(Pr^{-\alpha} + 2N - 3/\varepsilon) I_f\big) \\
& \sum_c a_{cf} = 1, \quad \forall f \\
& a_{cf} + a_{cg} - 1 \le s_{cfg}, \quad \forall c, f, g \\
& s_{cfg} \le a_{cf}, \quad \forall c, f, g \\
& s_{cfg} = s_{cgf}, \quad \forall c, f, g, \quad f < g \\
& I_f = \sum_{g \ne f} \sum_c s_{cfg} (d_{fg} + r)^{-\alpha}, \quad \forall f
\end{aligned}
$$

The MIQP for MMC is analogous but the first constraint should replaced with

$$
C \le Pr^{-\alpha} I_f^2 + Pr^{-\alpha}(Pr^{-\alpha} + 2N - 3/\varepsilon) I_f, \quad \forall f.
$$

Appendix C

C.1 Proof of Theorem 20

Proof of Theorem 20. We make the same changes as suggested in Theorem 19's proof. The suggested changes will result in the objective value changing from

$$(\Pi_i C_i) \text{ to } (C_k + C'_k - \bigtriangledown_k)\Pi_{i \neq k}(C_i - \bigtriangledown_i),$$

where C_i is the total capacity of user i. Note that $(C_k - \bigtriangledown_k) \geq 0$. Let η' be the ratio of the above objective values (new to old value). Below, we show that there exists an ϵ that makes $\eta' > 1$. This would imply that the given optimal solution is suboptimal (a contradiction), and thus, proving the theorem.

Now, using Eqn 5.3 and 5.4, we get:

$$\begin{aligned}
\eta' &= \left(\prod_{ineqk} \frac{C_i - \bigtriangledown_i}{C_i}\right) \frac{C_k + (C'_k - \bigtriangledown_k)}{C_k} \\
&\geq \left(\prod_{i \neq k} \frac{C_{min} - \bigtriangledown_i}{C_{min}}\right) \frac{C_{max} + (C'_k - \bigtriangledown_k)}{C_{max}} \\
&\geq \left(\frac{C_{min} - \bigtriangledown_{max}}{C_{min}}\right)^{n-1} \frac{C_{max} + (C'_k - \bigtriangledown_k)}{C_{max}} \\
&\geq (1 - a_1\epsilon)^{n-1}(1 + a_2\epsilon \log(1 + \frac{a_3}{\epsilon}) - a_4\epsilon)
\end{aligned}$$

where a_1, a_2, a_3, a_4 are appropriate positive constants (independent of ϵ) and $\bigtriangledown_{max}$ is the expression in Equation 5.3. Let η denote the last expression above. We can

now state the following:

$$\begin{array}{lll} (i) & \lim_{\epsilon \to 0} \eta = 1. & \\ (ii) & \frac{d\eta}{d\epsilon} = & (1-a_1\epsilon)^{n-1} \times \\ & & \left(\frac{-a_1(n-1)(1+a_2\epsilon \log(1+a_3/\epsilon)-a_4\epsilon)}{1-a_1\epsilon} + \right. \\ & & \left. a_2 \log(1+a_3/\epsilon) - \frac{a_2\epsilon}{(1+a_3/\epsilon)\epsilon^2} - a_4 \right) \\ & = & (1-a_1\epsilon)^{n-1}.\xi \end{array}$$

Also, one can easily verify that $\lim_{\epsilon \to 0^+} \xi = +\infty$ and $(1-a_1\epsilon)^{n-1}$ is always positive. Thus, $\frac{d\eta}{d\epsilon}$ is positive when $\epsilon \to 0^+$, which implies (from (i) above) that there exists an $\epsilon > 0$ such that $\eta > 1$ and thus $\eta' > 1$. ■

C.2 Proof of Lemma 12

Proof of Lemma 12. Instead of directly proving Lemma 12, we prove the following lemma.

Lemma 17 *Consider two users 1 and 2, and an SAPD solution (not necessarily optimal) $\{p_1(x), p_2(x)\}$ where each user uses the entire available spectrum $[0, W]$. We claim that there always exists an SAPD solution $\{p'_1(x), p'_2(x)\}$ with equal or higher total capacity such that either (i) both the PSD functions $p'_i(x)$ are constant in $[0, W]$, or (ii) one of the users does not use the entire spectrum $[0, W]$.* ■

Lemma 12 can be easily inferred from Lemma 17 by using contradiction. Lets consider an SAPD problem instance for two users, which has no optimal solution wherein the PSDs of the two users is constant in the shared part of the spectrum. From the set of optimal solutions, lets pick the one with minimum size of the shared spectrum. According to lemma 17, we can find another solution with equal or higher capacity in which either the size of the shared spectrum is reduced or the users use constant PSD's in the shared spectrum. In either case, we get a contradiction. We now present the proof of Lemma 17.

Proof of Lemma 17. We start with defining a couple of notations.

k-rectangular SAPD Solution. An SAPD solution $\{p_1(x), p_2(x)\}$ is considered to be k-rectangular if there exists frequency values w_i, such that $0 = w_0 < w_1 < w_2 < \ldots < w_{k-1} < w_k = W$ such that for each j $(1 \leq j \leq k)$ and x $(w_{j-1} \leq x < w_j)$, we have $p_1(x) = c_{1j}$ and $p_2(x) = c_{2j}$ for some constants c_{1j} and c_{2j}.

2-rectangular SAPD Solution. First, we prove the lemma for the special case when the given SAPD solution $\{p_1(x), p_2(x)\}$ is 2-rectangular. Without loss of generality, let us assume that the given SAPD solution is the optimal 2-rectangular SAPD solution, under the given total powers (viz., $\int_0^W p_1(x)dx$ and $\int_0^W p_2(x)dx$ respectively). Now, we can write the given optimal 2-rectangular SAPD solution as follows.

- For $0 \leq x < w$, $p_1(x) = \sigma_1$, $p_2(x) = \sigma_2$.
- For $w \leq x < W$, $p_1(x) = \sigma_1 + \Delta_1$, $p_2(x) = \sigma_2 + \Delta_2$.

Above, $\sigma_i > 0$, $\Delta_i + \sigma_i > 0$, for each i. Let Ψ_1 and Ψ_2 be the aggregate (sum over two links) capacity per unit-bandwidth in the two sub-spectrums $[0, w]$ and $(w, W]$ respectively. Without loss of generality, let us assume $\Psi_1 \leq \Psi_2$. We consider the following four cases.

$\Psi_1 = \Psi_2 = \Psi$ and $\Delta_1\Delta_2 = 0$. In this case, the given solution can be easily converted to a 1-rectangular solution of equal or higher capacity.

$\Psi_1 = \Psi_2 = \Psi$ and $\Delta_1\Delta_2 > 0$. Without loss of generality, we assume $\Delta_2 \geq \Delta_1 > 0$.[1] Note that, in either sub-spectrum, if we "scale-up" the PSD value of each link, then the aggregate capacity (per unit-bandwidth) would increase. Thus, for any $a > 1$, the PSD value of $a.\sigma_1$ and $a.\sigma_2$ would result in a higher aggregate capacity than Ψ_1 (= Ψ_2). Now, since $\Delta_i > 0$, there exists $a > 1$ such that $a.\sigma_i < \sigma_i + \Delta_i$ for each i. For such an a, changing the PSD value in the second sub-spectrum from $\sigma_i + \Delta_i$ to $a\sigma_i$ results in an increase in the aggregate capacity (with lower total power). Thus, the given solution is not an optimal 2-rectangular solution. QED.

$\Psi_1 = \Psi_2 = \Psi$ and $\Delta_1\Delta_2 < 0$. Without loss of generality, we can assume $\Delta_1 > 0$ and $\Delta_2 < 0$. Now, if $W > 2w$, let $[g_1, g_2] = [0, 2w]$ otherwise let $[g_1, g_2] = [W - 2w, W]$. Let $X(b)$ and $Y(b)$ be such that $\log X(b)$ and $\log Y(b)$ are the capacities per unit-bandwidth of the first and second links when they use a constant PSD value of $\sigma 1 + b\Delta 1$ and $\sigma 2 + b\Delta 2$ respectively; here, $b \in [-\frac{\sigma 1}{\Delta_1}, -\frac{\sigma 2}{\Delta 2}] \supseteq [0, 1]$. Below, we show how to choose appropriate b values to create a better 2-rectangular solution, or an equal-capacity solution wherein one of the links does not use the entire spectrum.

Let X_{max} be the maximum value of $X(b)$ over the above range of b. Since the above function $X(b)$ is reversible, we can define the function $f = Y(X^{-1}) : [0, X_{max}] \mapsto \mathbb{R}_{\geq 0}$ such that $f(x)$ gives the capacity-per-bandwidth of the second link when the capacity/bandwidth of the first link is x due to constant PSD values of $\sigma 1 + b\Delta 1$ and $\sigma 2 + b\Delta 2$ respectively for some b; note that, b is unique for a given x. We can show (we omit the details here) that the second-derivative of the function $(d(df(x)/dx)/dx)$ cannot be zero in $[0, X_{max}]$. Thus, the function $f(x)$

[1] If both are negative, then we can reverse the role of the two sub-spectrums.

has no inflection point in the range $[0, X_{max}]$, and hence, we can plot the various possibilities for the $f(x)$ relative to $y = 2^{\Psi}/x$ as shown in Figure C.1. Note that $f(x)$ is maximum at $x = 1$, and is 1 at X_{max}, and intersects the $y = 2^{\Psi}/x$ plot at two x values corresponding to $b = 0$ and $b = 1$ (since $\Psi_1 = \Psi_2 = \Psi$). Moreover, since $X(b)$ is monotonically increasing in b, we get the values/ranges of b as depicted in the figure. Now, for each of the four possibilities of $f(x)$ depicted in the Figure C.1, we can prove the lemma as follows.

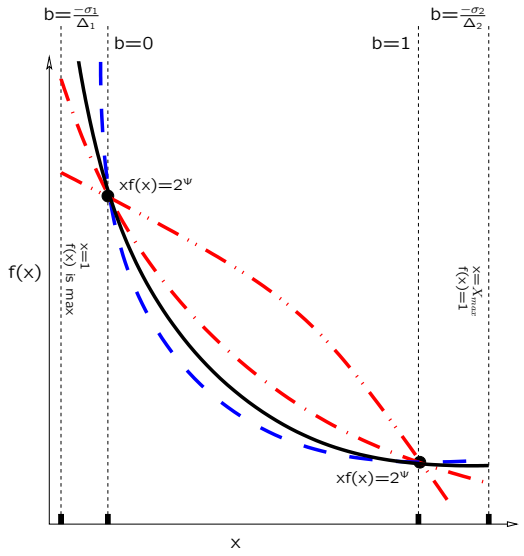


Figure C.1: Red and blue (dotted) curves are the possible shapes of $f(x)$; here, the black (solid) curve is $y = 2^{\Psi}/x$.

- If $f(x)$ is one of the two red plots, then we pick $b = 1/2$. For $b = 1/2$, we get $X(b)Y(b) > 2^{\Psi}$ and hence $\log X(b) + \log Y(b) > \Psi$. Now, if we can choose constant PSD values of $\sigma 1 + b\Delta 1$ and $\sigma 2 + b\Delta 2$ for the two links respectively in $[g_1, g_2]$, we get a 2-rectangular solution in $[0, W]$ of higher total capacity within the given power constraint. QED.

- If $f(x)$ is the blue or the black plot, then we choose two values of b, viz., b_l and b_r, so as to use PSD values of $\sigma_i + b_l\Delta_i$ in $[g_1, w]$ and $\sigma_i + b_r\Delta_i$ in $[w, g_2]$ for each link i. For our purposes, we need to choose b_l and b_r such that they satisfy the following three conditions: (i) $-\sigma_1/\Delta_1 \leq b_l \leq 0$, and $1 \leq b_r \leq -\sigma_2/\Delta_2$ (to ensure that b is in the valid range and the capacity/bandwidth is at least Ψ in each sub-spectrum), and (ii) $b_l + b_r = 1$ (to ensure that the total power used is at most the total power in the original

solution, for each link), and (iii) $\sigma_i + b_l\Delta_i$ or $\sigma_i + b_r\Delta_i$ is zero for some i (so that one of the links uses zero power in one of the sub-spectrums). To satisfy the above three conditions, we choose the pair (b_l, b_r) as $(-\sigma_1/\Delta_1, 1+\sigma_1/\Delta_1)$ if $1 + \sigma_1/\Delta_1 < -\sigma_2/\Delta_2$, or $(1 + \sigma_2/\Delta_2, -\sigma_2/\Delta_2)$ otherwise. The above yields an SAPD solution of higher capacity wherein one of the links doesn't use the entire spectrum. QED.

$\Psi_1 < \Psi_2$. In this case, we consider sub-spectrums $[g_1, w]$ and $[w, g_2]$ for some appropriate g_1 and g_2 (determined later), and increase the aggregate capacity within these sub-spectrums by appropriate redistribution of power.

Let $r = (g_2 - w)/(w - g_1)$, the ratio of the two sub-subspectrums, and $\tau = (g_2 - g_1)$. Let P_1' and P_2' be the total power used by link 1 and 2 in $[g_1, g_2]$, i.e., $P_i' = \tau(\sigma_i + \Delta_i r/(r + 1))$. Let $\Phi(r)$ be the aggregate capacity per bandwidth in $[g_1, g_2]$ when the PSD values are P_1'/τ and P_2'/τ respectively for the two links. We now show that a "large-enough" r will ensure that $(1 + r)\Psi(r) > \Psi_1 + r\Psi_2$, which will imply that in $[g_1, g_2]$ the 1-rectangular solution yields a higher total capacity than the given solution. Observe the following: (i) $\lim_{r\to 0^+} \Phi(r) = \Psi_1$, (ii) $\lim_{r\to\infty} \Phi(r) = \Psi_2$, and (iii) $\phi(r)$ is connected. Since $\lim_{r\to\infty^+} (1 + r)\Phi(r) = (1 + r)\Psi_2 > \Psi_1 + r\Psi_2$, there exists a large-enough r for which $(1 + r)\Psi(r) > \Psi_1 + r\Psi_2$. Once we find the appropriate r, we can determine g_1 and g_2 as follows: If $(r + 1)w < W$, then pick $[g_1, g_2] = [0, (r + 1)w]$, else pick $[g_1, g_2] = [w - (W - w)/r, W]$. Then, in $[g_1, g_2]$, we use power-signals of $\sigma_i + \Delta_i r/(r + 1)$ for link i, yielding a 2-rectangular solution with a higher-capacity than the given solution. QED.

k-rectangular Solution. This can be easily proven by induction on k, using the above result on $k = 2$ as the base case.

Arbitrary SAPD Solution. Let $p_1(x)$ and $p_2(x)$ be the power-distribution functions for the given solution, and let $P_i^* = \int_0^W p_i(x)dx$ be the total powers used by the links. Assume that there is no solution of equal or higher capacity, in which one of the link doesn't use the full spectrum. Let us construct an n-rectangular solution that "approximates" the given solution as follows: First, we divide the spectrum $[0, W]$ into n equi-sized sub-spectrums, and then, within each sub-spectrum we use a constant PSD value of minimum $p_i(x)$ in that sub-spectrum. Note that the total power used by the link i in the above n-rectangular solution is atmost P_i^*. Let F_n be the total capacity of the above n-rectangular solution, and let R be the total capacity of the 1-rectangular solution that uses a constant PSD of P_i/W for each link. Since the lemma holds for k-rectangular solutions, we get that $F_n \leq R$ for any n. Now, if C is the total capacity of the given solution, then by definition $C = \lim_{n\to+\infty} F_n$. Thus, we get $C \leq R$, which completes the proof. ∎

C.3 Proofs of Lemma 13 and 18

Proof of Lemma 13. First, it is easy to see that the union of the disjoint spectrums must be the entire available spectrum. Let the links use disjoint spectrums of size yW and $(1-y)W$ where $0 \leq y \leq 1$. Since both links should use maximum power for maximum capacity, we can compute the total capacity as follows.

$$C = yW\log(1+\frac{P_1h_{11}}{yWN_1}) + W(1-y)\log(1+\frac{P_2h_{22}}{(1-y)WN_2})$$

We can find the optimal value of y by solving for $dC/dy = 0$. We have:

$$\begin{aligned}\frac{dC(y)}{dy} = \; & W\big(\log(1+\tfrac{P_1h_{11}}{yWN_1}) - \log(1+\tfrac{P_2h_{22}}{(1-y)WN_2}) - \\ & \frac{P_1h_{11}}{yWN_1(1+\frac{P_1h_{11}}{yWN_1})} + \frac{P_2h_{22}}{(1-y)WN_2(1+\frac{P_2h_{22}}{(1-y)WN_2})}\big)\end{aligned}$$

The root of the equation $dC/dy = 0$ is:

$$y = \frac{N_2P_1h_{11}}{N_1P_2h_{22} + N_2P_1h_{11}}.$$

Hence, the PSD's of link 1 and 2 are $\frac{N_1P_2h_{22}+N_2P_1h_{11}}{WN_2h_{11}}$ and $\frac{N_1P_2h_{22}+N_2P_1h_{11}}{WN_1h_{22}}$ respectively and the optimal value of C is:

$$C = W\log(1+\frac{P_1h_{11}}{WN_1}+\frac{P_2h_{22}}{WN_2})$$

■

Lemma 18.

Lemma 18 *Consider a communication system with a single user 1, and an available spectrum $[0,W]$. Let the interference (from other users) in the sub-spectrums $[0,w]$ and $(w,W]$ be constant and equal to I and I' respectively. If $I > I'$, then to achieve maximum capacity for user 1, its PSD value in $[0,w]$ should be lower than in $(w,W]$.*

PROOF: It is easy to see that for optimal capacity: (i) the PSD should be constant in each of the sub-spectrums, and (ii) the link should use maximum power. Now, if we divide the total power of P_1 into the two sub-spectrums in the ratio of $k:(1-k)$, for some $0 \leq k \leq 1$, we get link capacity as:

$$C(k) = w\log(1+\frac{kP_1h_{11}}{w(I+N_1)}) + (W-w)\log(1+\frac{(1-k)P_1h_{11}}{(W-w)(I'+N_1)})$$

By solving $dC/dk = 0$, we get $k = \frac{w}{W} + \frac{w}{WP_1h_{11}}(I'-I)(W-w)$ which give us

the PSD values of $\frac{1}{W}(P_1 + (W-w)(I'-I)/h_{11})$ and $\frac{1}{W}(P_1 + w(I-I')/h_{11})$ in the two sub-spectrums. This proves the lemma, since the first PSD value is always greater than the second PSD value. ■

C.4 Cases for S_1 or S_2 = 0.

Case where S_1 or S_2 is of Zero Size. Let $S_1 = 0$ and $S_2 > 0$. In this case, Equations 5.8 and 5.9 are not valid. At the same time, the variables S_1 and c_1 are eliminated from the system, and hence, we have two fewer equations and variables which only simplifies the problem. We can use the exact same order of elimination and technique to yield an optimal solution for this case. This case of $S_2 = 0$ and $S_1 > 0$ is similarly handled, and the case of $S_2 = 0$ and $S_1 = 1$ is already handled by Lemma 14.

C.5 Upper Bound of σ_2

Upper bound of σ_2. Here, we show that there exists an upper bound for σ_2. Since the PSD's used by users 1 and 2 in S_1 and S_2 is $(c_1+\sigma_1)S_1$ and $(c_2+\sigma_2)S_2$ respectively, we have the following (by applying Lemma 13, and using the PSD values computed therein):

$$c_1 + \sigma_1 = \frac{N_2(c_1+\sigma_1)S_1h_{11} + N_1(c_2+\sigma_2)S_2h_{22}}{(S_1+S_2)N_2h_{11}},$$
$$c_2 + \sigma_2 = \frac{N_2(c_1+\sigma_1)S_1h_{11} + N_1(c_2+\sigma_2)S_2h_{22}}{(S_1+S_2)N_1h_{22}},$$

and $c_2+\sigma_2 = (c_1+\sigma_1)\frac{N_2h_{11}}{N_1h_{22}}$. Let $\varsigma = \frac{N_2h_{11}}{N_1h_{22}}$ and $\gamma = \max(\varsigma, 1)$. Let $P = P_1 + P_2$, and recall that c_1, c_2, σ_1, and σ_2 are positive numbers. Thus, we have:

$$\begin{aligned}
P &= (c_1+\sigma_1)S_1 + (c_2+\sigma_2)S_2 + (\sigma_1+\sigma_2)S_{12} \\
P &> \frac{1}{\varsigma}(c_2+\sigma_2)S_1 + \sigma_2S_2 + \sigma_2S_{12} \\
\gamma P &> (\gamma/\varsigma)\sigma_2S_1 + \gamma\sigma_2(S_2+S_{12}) \\
\gamma P &> \sigma_2(S_1+S_2+S_{12}) \quad (\text{as } \gamma \geq 1, \varsigma) \\
\gamma P/W &> \sigma_2
\end{aligned}$$

Thus, $\gamma P/W$ is an upper bound on σ_2, where $\gamma = \max(1, \frac{N_2h_{11}}{N_1h_{22}})$.