NETWORK-CENTRIC MECHANISMS FOR PERFORMANCE IMPROVEMENT IN DENSE WIRELESS NETWORKS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of the Ohio State University

By

Tarun Bansal, M.S.

Graduate Program in Department of Computer Science and Engineering

The Ohio State University

2014

Dissertation Committee:

Prasun Sinha, Advisor

Kannan Srinivasan

Anish Arora

ABSTRACT

In recent years, the number of wireless devices and the amount of data generated by these devices has seen an exponential growth. However, the number of channels available for data transmission has not increased significantly leading to the problem of "spectrum crunch". Our measurements show that the existing wireless deployments employ high density of wireless devices (access points, smartphones etc.). However, to prevent interference, the current wireless algorithms prohibit these neighboring wireless devices to operate simultaneously. The main focus of this thesis is on *improving the network–experience on the mobile devices by leveraging the high density of wireless devices.*

This thesis proposes four different solutions that are suited for different wireless topologies: *Symphony, RobinHood, Mozart* and *R2D2*. Symphony and RobinHood are best suited for Enterprise wireless networks where multiple access points are connected to each other and are willing to cooperate. Both Symphony and RobinHood use novel cooperative decoding techniques to enable multiple neighboring access points to simultaneously receive different packets on the same channel. Symphony is suitable for wireless networks that span large geographical areas while RobinHood is suitable for smaller deployments. Symphony and RobinHood leverage the high density of access points in Wi-Fi networks that otherwise remain unused in traditional wireless solutions.

Mozart is suitable for Wi-Fi networks where neighboring access points belong to different entities that may not be willing to cooperate. Mozart takes an unconventional approach

of letting all transmitters collide at the access point and then lets the access point decode the collided packets in the fewest number of slots.

Finally, R2D2 is designed for cellular networks and it enables neighboring devices to efficiently communicate with each other while reducing the dependence on the cellular base stations. R2D2 leverages the temporal-spatial asymmetry in the traffic patterns to efficiently allocate resources across cellular base stations as well as to efficiently schedule links at each of the base stations.

The thesis discusses all the solutions in detail, along with the techniques to address the challenges involved in their practical implementation.

Dedicated to my family.

ACKNOWLEDGMENTS

I would not have been able to finish this dissertation without the guidance, support, and encouragement I received from many great people in the past five years. I am especially grateful to my adviser, Prof. Prasun Sinha, from whom I have learned a lot. He gave me much freedom and independence in exploring many interesting research topics. I thank him for encouraging me all the time, teaching me how to overcome obstacles, giving me of his wisdom and long experience, and building my self-confidence in the times of disappointment and frustration.

I have also greatly benefited from close interactions with Professor Kannan Srinivasan. His unique perspectives on research problems have influenced and enriched my way of thinking. His inquisitive nature coupled with his desire to push helped me in making several improvements to my dissertation. I would also like to thank my thesis committee members Professor Anish Arora and Professor Luis Rademacher for their helpful comments and suggestions.

I appreciate the suggestions and the help offered by Dr. Karthik Sundaresan and Dr. Sampath Rangarajan during my summer internship at NEC Labs America. Their invaluable guidance and support greatly helped in shaping the last piece of my thesis, R2D2.

I also want to thank my colleagues at Ohio State University: Zizhan Zheng, Ren-Shiou Liu, Zhixue Lu, Shengbo Chen, Yousi Zheng, Dong Li, Wenjie Zhou, and Bo Chen. I will always remember some of the memorable experiences shared with this group such as performing localization experiments on the streets of Columbus.

I would not be able to complete this dissertation without the tremendous support, and countless encouragement from my father, Ravinder Bansal, my mother, Veena Bansal, and my brother, Rahul Bansal who have always stood besides me, believing in me, and giving me strength and comfort when I most needed them. I am very thankful and fortunate to have them on my side through all the ups and downs in my life.

Finally, I would like to thank my wife Yanyan, who has always been encouraging to me. She has gladly taken over most of our family responsibilities, allowing me more time for studies. I simply could not have reached this stage without her. I also would like to thank to my daughter, Aarini, who joined us when I was writing my dissertation, for giving me unlimited happiness and pleasure.

VITA

2006	B.S. Department of Computer Science and Engineering, Indian Institute of Technology (IIT), Roorkee, India
2009	M.S. Department of Computer Science and Engineering, University of Texas at Dallas, Richardson, TX
2013	M.S. Department of Computer Science and Engineering, Ohio State University, Columbus, OH
2009-Present	Ph.D. Department of Computer Science and Engineering, Ohio State University, Columbus, OH

PUBLICATIONS

Tarun Bansal, Karthikeyan Sundaresan, Sampath Rangarajan and Prasun Sinha, "R2D2: Embracing Device-to-Device Communication in Next Generation Cellular Networks," *in Proc. of IEEE International Conference on Computer Communications (INFOCOM), April 2014.*

Tarun Bansal, Bo Chen and Prasun Sinha, "FastProbe: Malicious User Detection in Cognitive Radio Networks Through Active Transmissions," *in Proc. of IEEE International Conference on Computer Communications (INFOCOM), April 2014.*

Dong Li, Zhixue Lu, Tarun Bansal, Erik Schilling and Prasun Sinha, "ForeSight: Mapping Vehicles in Visual Domain and Electronic Domain," *in Proc. of IEEE International Conference on Computer Communications (INFOCOM), April 2014.*

Tarun Bansal (Co-Primary), Wenjie Zhou (Co-Primary), Kannan Srinivasan and Prasun

Sinha, "RobinHood: Sharing the Happiness in a Wireless Jungle," *in Proc. of ACM HotMobile, February 2014.*

Tarun Bansal (Co-Primary), Bo Chen (Co-Primary), Prasun Sinha and Kannan Srinivasan, "Symphony: Cooperative Packet Recovery over the Wired Backbone in Enterprise WLANs," *in Proc. of ACM International Conference on Mobile Computing and Networking (MobiCom), September 2013.*

Tarun Bansal, Bo Chen and Prasun Sinha, "DISCERN: Cooperative Whitespace Scanning in Real Environments," *in Proc. of IEEE International Conference on Computer Communications (INFOCOM), April 2013.*

Dong Li (Co-Primary), Tarun Bansal (Co-Primary), Zhixue Lu (Co-Primary) and Prasun Sinha, "MARVEL: Multiple Antenna based Relative Vehicle Localizer," *in Proc. of ACM International Conference on Mobile Computing and Networking (MobiCom), August 2012.*

Shengbo Chen (Co-Primary), Tarun Bansal (Co-Primary), Yin Sun (Co-Primary), Prasun Sinha, and Ness Shroff, "Life-Add: Lifetime Adjustable Design for WiFi Networks with Heterogeneous Energy Supplies," *in Proc. of WiOPT, 2013.*

Tarun Bansal, Dong Li and Prasun Sinha, "Fairness by Sharing: Split Channel Allocation for Cognitive Radio Networks," *to appear in IEEE Transactions on Mobile Computing (TMC).*

Zhixue Lu, Tarun Bansal and Prasun Sinha, "Achieving User-Level Fairness in Open-Access Femtocell based Architecture," *IEEE Transactions on Mobile Computing (TMC), Oct 2013.*

FIELDS OF STUDY

Major Field: Computer Science and Engineering

Specialization: Networking

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

The number of mobile devices and the amount of data communicated by the applications on these devices has been growing at an exponential rate during the past few years. However, the number of channels available for data transmission has not increased significantly leading to the problem of "spectrum crunch". Network administrators have tried to tackle this problem by employing high density of access points. However, to prevent interference, the current wireless algorithms prohibit these neighboring access points to operate simultaneously. This thesis focuses on improving the network–experience on the mobile devices by leveraging the high density of wireless devices. This thesis proposes four different algorithms that are suited for different wireless topologies: *Symphony, RobinHood, Mozart* and *R2D2*.

Symphony, RobinHood, and Mozart focus on the Wi-Fi networks. In Wi-Fi networks, due to discrepancy between the interference at the transmitter and at the receiver, it becomes very difficult for the transmitters to precisely estimate the interference at receivers which leads to hidden and exposed terminal problems. Further, the IEEE 802.11 protocol and its derivatives (e.g., 802.11 with RTS- CTS, 802.11ec [55], ZigZag [29] etc.) require the channel to remain idle when the nodes are undergoing backoffs. In dense deployments, such idle listening leads to up to 30% loss of throughput [40]. The focus of Symphony, RobinHood, and Mozart is to eliminate these losses and propose novel physical and MAC layer techniques to enable multiple transmissions in the neighborhood.

On the other hand, the focus of R2D2 is on cellular networks where the high density of devices has forced service providers to densely deploy the base stations. However, within a single base station, the current algorithms allow at most one device to operate on a given frequency. In this thesis, we explore a recently proposed communication paradigm called Device to Device (D2D) communication, that allows neighboring devices to directly communicate with each other.

In Chapter 2, this thesis proposes Symphony[1] [11], a cooperative decoding approach that minimally uses the wired connection among APs to increase the throughput of wireless networks. Symphony encourages collision of packets among transmitters at APs. On receiving multiple packets, APs suppress a subset of colliding transmitters. This reduces the number of transmitters in each slot. Eventually, the number of transmitters reduces to a small enough value such that the APs can cooperatively decode all the received packets.

The working of Symphony is best explained through an example. Figure 1.1a shows a simple topology with two APs and four clients where the APs are connected to each other through a wired backbone. Assume that each of the clients wants to upload one packet to the wired network. For this topology, omniscient TDMA will take at least four slots to schedule the four transmissions (See Figure 1.2a). In Symphony, on the other hand, all the four clients transmit (See Figure 1.2b) in the first slot. At the end of the slot, Symphony suppresses Alice (A) and Bob(B). In the next slot, only Carol(C) and Don(D) transmit. At the end of the second slot, AP_1 decodes the packet D since AP_1 received it interference-free. After decoding, it sends the decoded bits of D to AP_2 over the wired backbone. From the received bits, after correcting for different offsets, AP_2 recreates $D's$ received samples and subtracts them from the samples received in slot 2. After subtraction, it is left with only the samples corresponding to C which AP_2 decodes. At the same time, AP_1 recreates the samples of D as received in slot 1 and subtracts those samples

[1] A joint work with Bo Chen

(a) Network topology. AP_1 is assumed to be outside the interference range of Bob and Carol.

	AP\ Slot	Slot 1	Slot 2	Slot 3	Slot 4
Omniscient TDMA	AP 1	(A)			(D)
	AP 2	A	(B)	(C)	D
Symphony	AP 1	(A) D	(D)		
	AP 2	A (B) C D	(C) D		

(b) Set of packets received by APs on air when using different algorithms. Circles indicates that the packet was decoded using samples received in that slot.

Figure 1.1: Network topology and set of packets received by APs.

from the samples received in slot 1. AP_1 then decodes the remaining samples to obtain A and sends it to AP_2. Finally, AP_1 recreates the received samples of A, C and D for slot 1. After subtracting these samples from the samples received in slot 1, AP_2 decodes B. Figure 1.1b shows the set of packets received by the two APs in different slots and samples from which slot are used to decode which packet. Using cooperation among APs and by using the backbone for exchanging decoded packets, Symphony enables the two APs to receive the four packets in two slots. Thus, by utilizing cooperation among APs to simultaneously decode multiple colliding transmissions, Symphony provides twice the throughput as compared to omniscient TDMA. Chapter 2 also shows that by encouraging collisions among transmissions, Symphony creates more opportunities of harnessing the power of Successive Interference Cancellation (SIC). This allows receivers in Symphony to decode multiple transmissions in the same slot. For the example network shown in Figure 1.1, when using SIC, Symphony can decode all the four packets in a single slot (See Figure 1.2c).

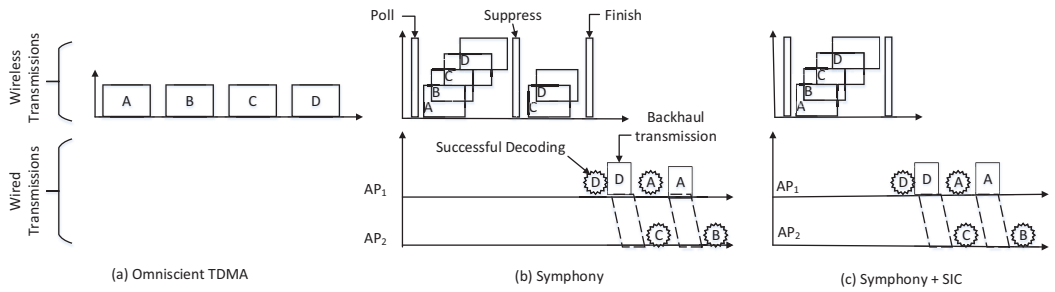
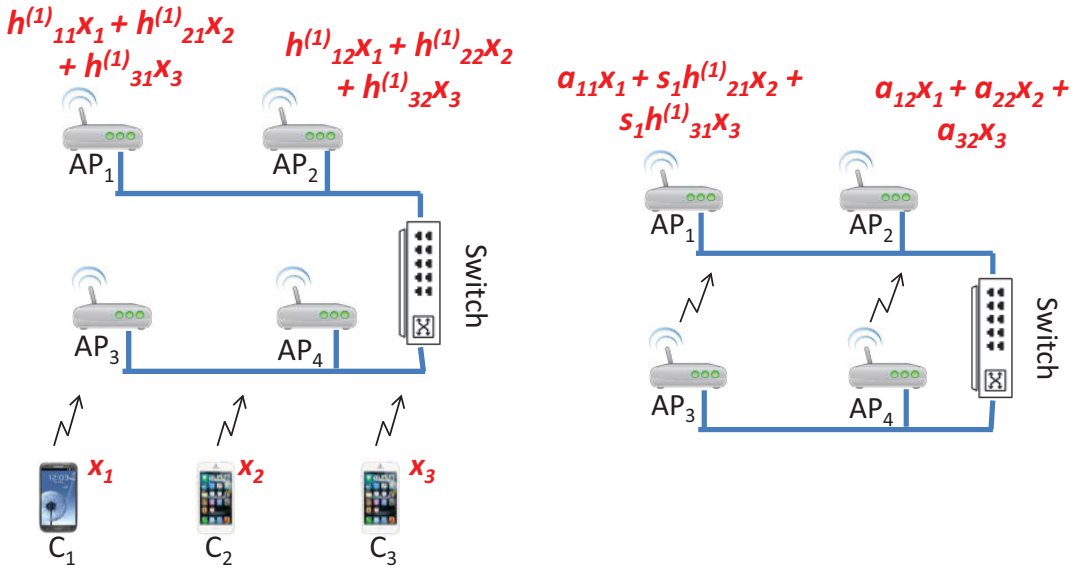


Figure 1.2: Omniscient TDMA takes a minimum of 4 slots. Symphony can receive four packets in two slots. By leveraging SIC, Symphony can receive all the four packets in a single slot. Symphony does decoding in the reverse chronological order by first decoding the packets received in the last slot.

In the next chapter, this thesis describes solution RobinHood[2] [13] that scales the uplink throughput with the number of clients in the network by enabling multiple nearby access points to concurrently receive uplink packets from multiple mobile clients, all within a single collision domain. RobinHood does not increase energy consumption on the clients and executes exactly over two time slots. RobinHood leverages three properties that are unique to Enterprise Wireless LANs (EWLANS): (i) *Dense deployment of APs (See Fig. 1.6 and [57])*; (ii) *Capability of these APs to exchange packets with each other over the underutilized wired backbone*; and, (iii) *Immobility of APs resulting in relatively stationary channels* (See Fig. 1.7).

Consider the example enterprise WLAN shown in Fig. 1.3a where all the APs and the three clients are in a single collision domain. Assume that the three users want to upload one packet each to the backbone. An omniscient TDMA scheduling algorithm with global knowledge would require three time slots to complete this upload. In RobinHood, in the first slot as shown in Fig. 1.3a, all users will transmit at the same time. All the 4 APs

[2]A joint work with Wenjie Zhou



(a) First slot. x_1, x_2 and x_3 are the three packets transmitted by C_1, C_2 and C_3, respectively. h^1_{ij} is the channel from client i to AP_j during time slot 1.

(b) Second slot. A subset of APs transmit in the second slot while the rest of the APs receive. a_{ij} are the final channel coefficients after the transmission of the second slot. s_i is the scaling coefficient at AP_i.

Figure 1.3: Illustration of RobinHood over a topology of 3 clients and 4 APs. All devices belong to the same collision domain and can hear each other.

will receive a combination of three transmitted packets. In the second slot, AP_3 and AP_4 will retransmit the received signals by first precoding [37] them such that the following condition is satisfied as shown in Fig. 1.3b: At AP_1, samples corresponding to x_2 and x_3 in the second slot align with the samples corresponding to x_2 and x_3 in the first slot. Decoding happens in multiple steps as follows:

- At the end of the second slot, AP_1 scales the samples received by AP_1 in the second slot and subtracts them from the samples received in the first slot. This scaling is done such that samples corresponding to x_2 and x_3 are *nulled*. Afterwards, it is left with only the samples corresponding to x_1. AP_1 decodes the samples to obtain the

packet transmitted by C_1. Next, it transmits the decoded packet over the backbone to AP_2.

- AP_2 recreates the samples corresponding to x_1 and subtracts them from the samples received in the first slot and the second slot.
- After subtraction, AP_2 is left with two equations (one from each slot), and two variables (x_2 and x_3). AP_2 solves the two equations to obtain x_2 and x_3.
- Afterwards, AP_1 and AP_2 forward x_1, x_2 and x_3 towards their destinations.

Symphony and RobinHood are suitable for networks where the neighboring APs cooperate with each other. For general wireless networks where the APs may not be willing to cooperate, Chapter 4 presents a new cross-layer solution called Mozart[3], that encourages collisions and hidden terminal transmissions in a planned way to enable fast recovery of colliding packets via a combination of Successive Interference Cancellation (SIC) and a new approach called *successive packet subtraction*. In Mozart, a receiver simultaneously receives multiple packets from different transmitters resulting in a collision. The receiver then smartly suppresses transmissions in subsequent slots based on its estimation of signal strengths from various senders so as to best apply a combination of mechanisms involving signal subtraction and SIC to recover all the colliding packets. Finally, the receiver decodes packets by using "successive packet subtraction". Consider the example shown in Figure 1.4 where in the first slot, the receiver receives four packets. However, at the end of every slot, the receiver *suppresses* one transmitter while other transmitters retransmit the same packet. Eventually, in the fourth slot, only one packet (P_4) is received which the receiver can simply decode. Next, it subtracts the samples of P_4 from slot 3 to decode P_1. This process is continued until the receiver decodes all the packets. Chapter 4 explains

[3]A joint work with Bo Chen

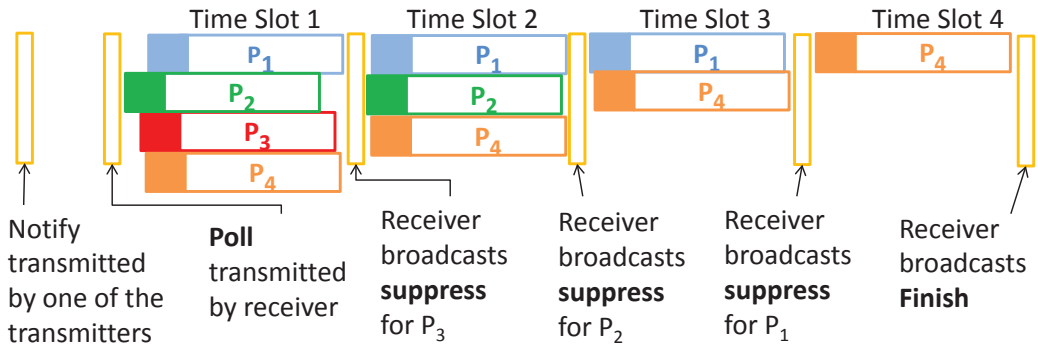


Figure 1.4: Collision Recovery Period. Through control messages, the receiver ensures that the number of transmitters is reduced by 1 from the previous slot. Data transmissions in the same slot may arrive at different times at the receiver due to propagation delays and radio's TX-RX turn-around time. At the end of the recovery period, the receiver will reconstruct samples for P_4 and subtract it from samples received in slot 3. The remaining samples are then decoded to obtain P_1. Similarly, P_2 and P_3 are decoded from the samples of slot 2 and slot 1, respectively.

how this approach allows Mozart to eliminate hidden terminal problem while harnessing Exposed terminals. Mozart also significantly reduces the backoff time, thereby improving the throughput of the nodes.

Device-to-device (D2D) communications is being pursued as an important feature [3] for the next generation cellular networks (LTE-advanced). The goal of D2D is to leverage the high density of communicating devices to improve the network utilization in two ways: (i) *offload:* a data session between two devices (D1, D2) in the same sector which conventionally incurs two hop transmissions in the cellular mode (D1→BS, BS→D2) now requires only a single hop transmission (Fig. 1.5) in D2D mode (D1→D2), and (ii) *reuse:* leveraging the physical proximity, the D2D communication can further operate on resources on which conventional cellular users are already scheduled. In Chapter 5, this

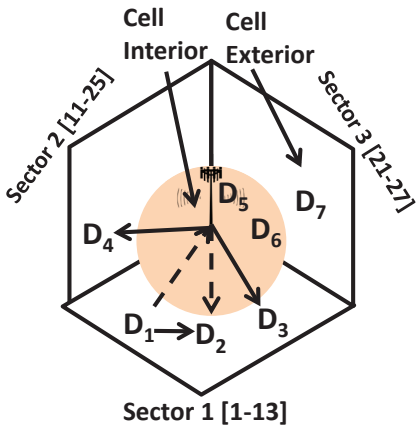


Figure 1.5: D2D usage

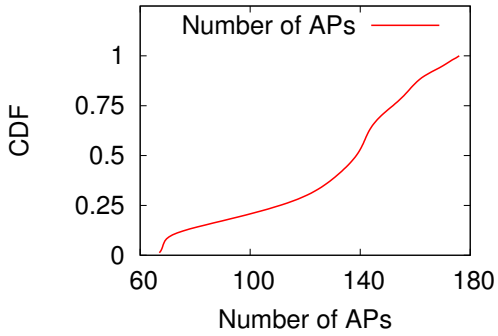
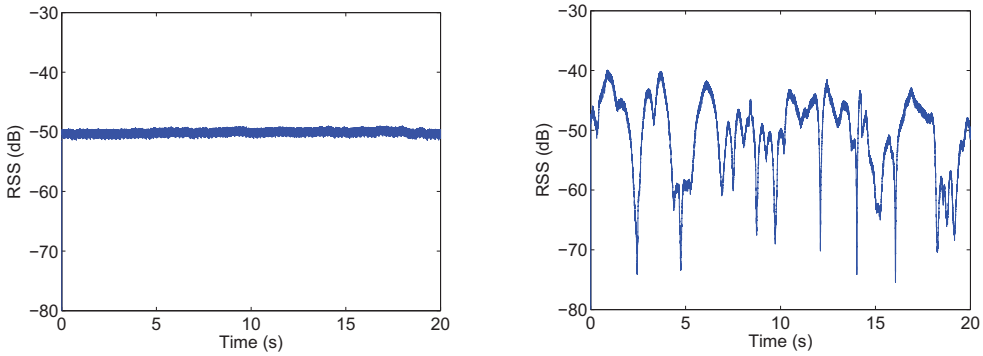


Figure 1.6: CDF of number of APs observed across different locations. The data was collected at multiple places including a hospital, a large university library and an apartment complex.

thesis proposes solution *R2D2* [12] that efficiently integrates D2D communications in the existing cellular networks.

This thesis shows that in multicell deployments with sectorized cells, the existing reuse provided by the cellular deployment leaves little room for D2D communication to provide additional reuse. Hence, most of D2D's gain is restricted to its ability to offload cellular



(a) Channel between a pair of APs

(b) Channel between a mobile client and an AP

Figure 1.7: Received Signal Strength (RSS) in an office environment. The channel between APs is relatively stationary compared to channel between AP and mobile client.

traffic. *R2D2* leverages the temporal variations in the cellular downlink and uplink traffic and places D2D traffic *flexibly* on the cellular resources that would have otherwise gone wasted.

Chapter 5 also proposes a two time scale solution: (i) At the *beginning of every epoch* (lasting several tens of frames), *R2D2* estimates the average traffic (resource) demand from cellular and D2D traffic in each sector and assigns cellular resources to each of the sectors; and, (ii) *In every frame*, for the set of sectors co-located at the same base station and instantaneous traffic demands, *R2D2* solves the coupled problem of D2D traffic placement and scheduling of cellular and D2D traffic jointly on both the DL and UL resources as well as across the sectors. Thus, while the coarse time-scale component in *R2D2* allocates resources and removes interference only between cross sectors (through dynamic FFR) for an entire epoch, the fine time-scale component is responsible for alleviating the interference between co-located sectors generated by D2D traffic (through joint sector scheduling) and maximizing the utilization of allocated resources in every frame in the epoch.

1.1 Contributions and Structure of this Thesis

The key contributions of this thesis are as follows:

- In our work on uplink traffic performance in Enterprise WLANs, we propose two different algorithms Symphony and RobinHood (Chapter 2 and Chapter 3, respectively). Symphony works across APs that belong to different collision domains while RobinHood enables multiple APs in the same collision domain to simultaneously receive data. This thesis also discusses the results from the deployment of both Symphony and RobinHood on wireless testbeds.

- Chapter 4 discusses Mozart that improves the wireless throughput in networks where APs may not be willing to cooperate with each other. Mozart implicitly handles the hidden terminal problem by requiring all neighboring clients to transmit simultaneously and then carefully decoding the collided packets one-by-one.

- Chapter 5 presents a two-time scale algorithm for efficiently integrating Device-to-Device communications in existing cellular networks. The proposed algorithm does resource block allocation across different base stations. Further, at a finer time-scale, it schedules uplink, downlink and D2D communications at each cellular base station.

The rest of the thesis is organized as follows. Chapter 2 presents our work on the Symphony algorithm. Chapter 3, we present our algorithm RobinHood that scales the uplink throughput in Wireless LANs. Chapter 4 presents Mozart, an algorithm for packet decoding. In Chapter 5, we present R2D2 that efficiently integrates D2D communications in existing cellular networks. Chapter 6 concludes this thesis.

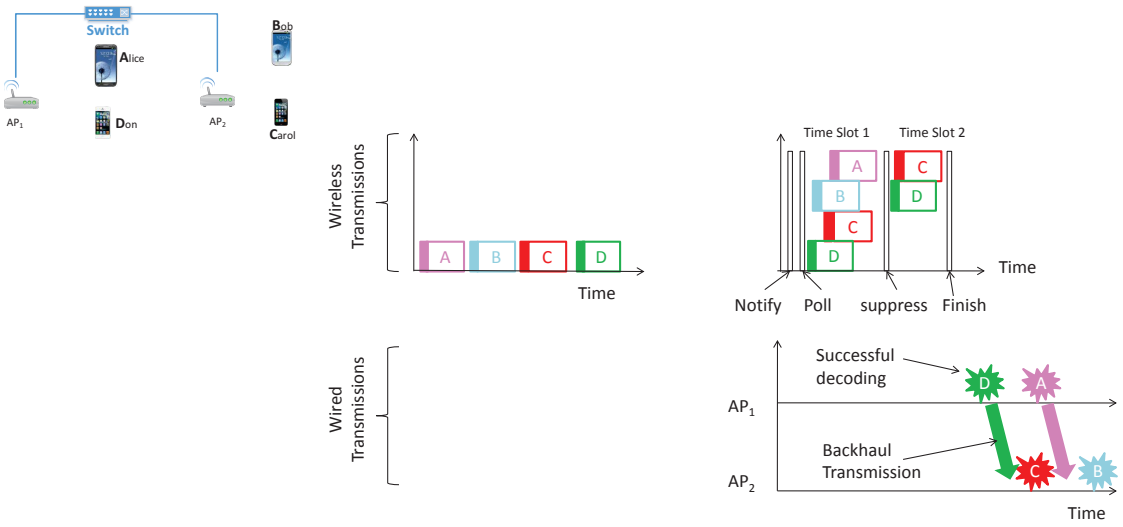
CHAPTER 2

SYMPHONY: COOPERATIVE PACKET RECOVERY OVER THE WIRED BACKBONE IN ENTERPRISE WLANS

2.1 Introduction

APs in Enterprise Wireless Local Area Network (EWLAN) are typically connected using a wired backbone. *In this chapter, we propose Symphony, a cooperative decoding approach that minimally uses the wired connection among APs to increase the throughput of wireless networks.* Symphony encourages collision of packets among transmitters at APs. On receiving multiple packets, APs suppress a subset of colliding transmitters. This reduces the number of transmitters in each slot. Eventually, the number of transmitters reduces to a small enough value such that the APs can cooperatively decode all the received packets.

Figure 2.1a shows a simple topology with two APs and four clients where the APs are connected to each other through a wired backbone. Assume that each of the clients needs to upload one packet to the backbone. For this topology, omniscient TDMA will take at least four slots to schedule the four transmissions (See Figure 2.1b). In Symphony, on the other hand, all the four clients transmit (See Figure 2.1c) in the first slot. At the end of the slot, Symphony suppresses Alice (A) and Bob(B). In the next slot, only Carol(C) and Don(D) transmit. At the end of the second slot, AP_1 decodes the packet D since AP_1 received it interference-free. After decoding, it sends the decoded bits of D to AP_2 over the



(a) Network Topology (b) Omniscient TDMA (c) Symphony

Figure 2.1: Network topology and set of packets received by APs. (a) Network topology. AP_1 is assumed to be outside the interference range of Bob and Carol. (b) Omniscient TDMA takes a minimum of 4 slots. (c) Symphony can receive four packets in two slots. In Symphony, APs cooperatively decode packets in the reverse chronological order by first decoding the packets received in the last slot.

wired backbone. From the received bits, after correcting for different offsets, AP_2 recreates samples received from D and subtracts them from the samples received in slot 2. After subtraction, it is left with only the samples corresponding to C which AP_2 decodes. At the same time, AP_1 recreates the samples of D as received in slot 1 and subtracts those samples from the samples received in slot 1. AP_1 then decodes the remaining samples to obtain A and sends it to AP_2. Finally, AP_2 recreates the received samples of A, C and D for slot 1. After subtracting these samples from the samples received in slot 1, AP_2 decodes B. Using cooperation among APs and by using the backbone for exchanging decoded packets, Symphony enables the two APs to receive the four packets in two slots. Thus, by utilizing

cooperation among APs to simultaneously decode multiple colliding transmissions, Symphony provides twice the throughput as compared to omniscient TDMA.

Cooperation among APs has been used before to increase wireless throughput. Authors in [57] focus on improving the client-AP association and AP channel assignment. Miu *et al.* [56], Woo *et al.* [79]and Gowda *et al.* [32] have respectively proposed techniques for bit-level, symbol-level and coarse symbol-level combining across different APs. In these solutions, a packet overheard by different APs is decoded cooperatively by multiple APs. However, in all these algorithms, all the APs cooperate to decode a single packet while in Symphony, APs cooperate to simultaneously decode multiple packets. Further, symbol-level combining [56] across different APs is prohibitive due to significantly higher bandwidth requirements [32,79] on the wired backbone. In Symphony, APs only exchange decoded packets which are typically much smaller in size compared to raw samples [32,79]. The idea of exchanging packets on the backbone was also studied in [30]. However, unlike Symphony, [30] provides throughput improvement only when all the devices have at least two antennas.

Several prior works [49, 65, 72] have focused on increasing network throughput for downlink traffic in EWLANs while neglecting the uplink traffic. On the other hand, Symphony primarily improves the network throughput on uplink while allowing efficient schemes such as conflict-free TDMA scheduling [49] to be utilized over the downlink. We believe that upstream traffic is increasing rapidly due to rise in use of a wide-range of new storage paradigms, computing paradigms and applications, such as cloud computing, cloud storage, P2P file access, video conferencing, online gaming, VoIP, and traffic generated from the mobile devices (e.g., location information or sensor readings). Using cooperation to improve uplink wireless throughput is particularly challenging, since APs are not aware of which client has data to transmit making it difficult to schedule transmissions. Collecting this information from different clients is also cost-prohibitive.

Although the idea behind Symphony is simple, there are multiple challenges to realize it in practice.

- **Set of nodes to suppress:** At the end of each slot, Symphony needs to determine the set of nodes to be suppressed. This set should be feasible (ensuring that the set of suppressed transmissions would be decoded in future) and large, so that all the collided transmissions can be decoded in a small number of slots.

- **Synchronization and variable non-zero latency:** Cooperation among APs requires that APs be synchronized with each other and that their mutual-latency be low and predictable. However, our measurements described in Section 2.6.2 show that these requirements do not hold true in large networks.

- **Absence of central controller:** In wireless networks, it may not be possible to set up a central controller that coordinates all the APs in the entire network due to the large size of the network. Thus, it is required that cooperative decoding should work in the absence of central controller.

- **Knowledge of topology:** To cooperatively suppress a subset of transmitters, the APs need to know the network topology. This information (particularly the interference among non-AP nodes) is difficult to acquire for large scale networks [30, 49] and could lead to significant overhead.

- **Difference in packet lengths:** It is possible that different transmitters may have different amount of pending data left. However, if the channel access time is fixed as shown in Figure 2.1c, then the users with less pending data may not fully use the access time leading to channel wastage.

The chapter is organized as follows. The next section describes a centralized algorithm that assumes the presence of a central server. In Section 2.3, we describe our distributed

protocol, Symphony that handles all the above challenges. In Section 2.4, we explain how Symphony harnesses SIC to increase wireless throughput. The following section discusses some issues that makes our solution more practical. Section 2.6 and Section 2.7 outline the results from our USRP testbed experiments and ns-3 based simulations, respectively. Section 2.8 presents the state of the art and finally, Section 2.9 concludes this chapter.

2.2 Cooperative Packet Subtraction

In this section, we present a simplified version of our algorithm, called Centralized Algorithm (CA) that works under a centralized framework. In Subsection 2.2.2, we explain the practical limitations of CA.

CA assumes the presence of a central server (CS) that coordinates different APs. CA also assumes that all the APs are synchronized with the CS and their mutual latency is zero. We relax these assumptions in Section 2.3 where we describe our distributed algorithm. CA works in multiple phases as described below (See Figure 2.1c):

- **Polling:** First, using the wired backbone, the CS asks all the APs to simultaneously transmit poll packets on the air. The poll packets are encoded using PN sequences [55]. Use of PN sequences, allows the clients to correlate them even if multiple poll packets collide.

- **Data Transmission:** Upon receiving the poll, the clients that have uplink data transmit their data packets. The clients also encode their IDs using their PN sequence and include it in the packet's header at the front. Due to simultaneous transmissions, multiple packets may collide at the APs. As explained in Section 4.3.1, each AP uses the PN sequence to determine the ID of as many colliding transmitters as possible, and then forwards this information to the CS using the wired backbone.

- **Computing the set of suppressed clients:** On receiving the *best-effort* information

about which AP received packets from which clients, the CS uses Algorithm 1 (explained in Section 2.2.1) to compute the subset of transmitters to be suppressed at the end of this slot. The algorithm described in Section 2.2.1 also determines which AP should suppress which client. All this information is conveyed by CS to the APs through the backbone.

- **Transmitting the suppress packets:** Upon receiving the information about which nodes to suppress, the APs transmit the *suppress* packet that includes the PN sequence of the client that is suppressed. Similar to poll packets, here also use of PN sequences allows the clients to correlate them even if multiple suppress packets collide.

- **Data retransmission:** Each client upon hearing the suppress determines if it is being suppressed: If so, it does not transmit (until it receives *finish*), otherwise, the client retransmits its data.

- **Finish:** As the nodes are suppressed, the number of concurrent transmissions received by APs decreases with each slot. Eventually, it becomes possible for the APs to use cooperation and decode all the transmissions received in the last slot. As described in Section 2.1, this decoding is done in the *reverse chronological order* until all the received packets are decoded. At that point, the CS directs all APs to broadcast *finish*. The reception of a finish packet at the clients indicates that the data transmitted by the nodes has been successfully received at APs. This allows the nodes to transmit the next MAC-layer data frame.

Another way of scheduling transmissions may be that in the first slot, using the poll, APs gather the PN sequences of all the clients that have uplink data and then schedule different transmitters in a TDMA fashion. However, this approach does not work well since APs may not detect PN sequences from transmitters that have low SINR (See 4).

Thus, transmitters with low SINR will not be scheduled and get starved. On the other hand, the PN sequences of such transmitters will be detected in CA in later slots when the number of colliding packets become small. So, with the reverse chronological decoding of Symphony, the transmitters with low SINR will also be suppressed and eventually their packets would also be decoded.

To prevent overwhelming [32, 79] the wired backbone, one of the requirements for cooperative decoding is that APs should not exchange sample level information on the wired backbone[1]. The set of transmissions suppressed during a slot and the samples received at all the APs during that slot form a set of linear equations. Observe that, just because the set of equations is solvable, it does not imply that all the suppressed transmissions can be decoded without exchanging sample level information. Consider the example network shown in Figure 2.1a. If at the end of slot 1, the CS suppresses A and D, then even though the set of linear equations has 2 variables (A and D) and 2 equations (samples received at AP_1 and AP_2), still it is not possible to decode both A and D *without the exchange of samples*. Thus, we need another mechanism to compute the set of suppressed transmissions.

2.2.1 Computing the set of suppressed transmissions

At the end of each slot, the CS computes the set of suppressed transmissions. *This set should satisfy the following four requirements*: (*i*) **Feasibility:** It should be a feasible set such that the APs can successfully decode all the suppressed packets without exchanging sample-level information; (*ii*) **Optimality:** It should maximize the number of transmissions suppressed in this slot; (*iii*) **Optimality in future slots:** It should maximize the applicability of cooperative decoding in future slots; and, (*iv*) **Low backbone overhead:**

[1]Although by exchanging coarse samples [32], it is possible to reduce the overhead in the backbone, it is not enough for our purpose since high fidelity representation of symbols is required to decode a packet in the presence of interference from other transmissions.

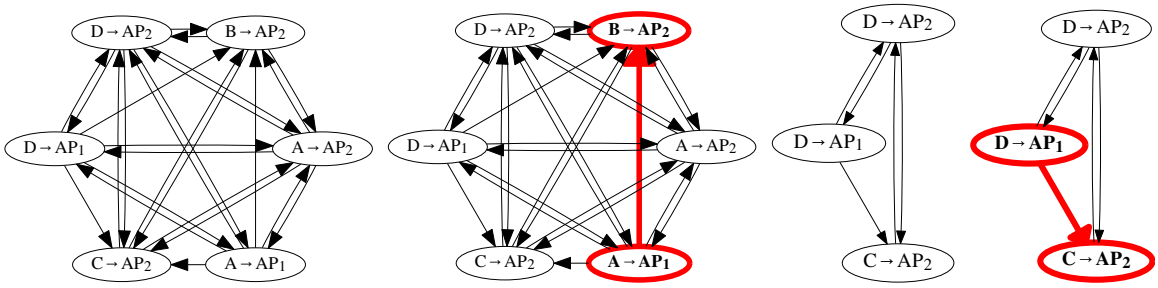
It should minimize the number of decoded packets that need to be exchanged among APs, thereby reducing the overhead on the wired network.

Computing the largest set of feasible suppressed transmissions (while taking into account only the first two requirements) is a NP-Hard problem (shown in Section A.2). To that end, we propose a two-phased greedy algorithm that computes the set of suppressed transmissions by first constructing a *dependence graph* and then picking its maximum acyclic induced subgraph (Both phases are explained below). This acyclic graph corresponds to the set of transmissions that can be successfully suppressed in this slot. In this chapter, we use P_i to denote packet received from client i. We overload this term and also use P_i to represent the client that transmitted the packet.

Phase 1: Constructing the dependence graph: When using cooperative decoding, decoding of one transmission may depend on successful decoding of another transmission. For the example network shown in Figure 2.1a, let's say that in the first slot, the CS suppresses A and B. However, since A's transmission interferes at AP_2, therefore, B's transmission at AP_2 can be decoded only after the former transmission has been decoded at AP_1. In general, for two transmissions $\{P_i, AP_j\}$ and $\{P_k, AP_l\}$, if P_k interferes at AP_j, then the transmission from P_i can be decoded at AP_j only after P_k's transmission has been decoded. Thus, we say that the decoding of P_i's transmission is *dependent* upon the decoding of P_k's transmission. In this phase, the CS constructs a *dependence graph* for the current slot (Lines 3-4 of Algorithm 1)):

- **Vertices:** Corresponding to each client-AP link in the topology, a vertex is created in the dependence graph. We use v_{ij} to denote the vertex that corresponds to the link $\{P_i, AP_j\}$. v_{ij} is created *iff* P_i transmitted at least one packet in this slot and AP_j is in the receiving range of P_i (i.e., AP_j can decode P_i in the absence of interference).

- **Edges:** Edges in the dependence graph capture the decoding dependence among different transmissions. A directed edge is added from vertex v_{ij} to a vertex v_{kl} *iff*



(a) G_d at the end of slot 1

(b) Two vertices and one induced edge (shown in bold) form the Acyclic subgraph (G_s) at the end of slot 1.

(c) G_d at the end of slot 2

(d) G_s (shown in bold) at the end of slot 2.

Figure 2.2: Dependence graph (G_d) and Induced Directed Acyclic Subgraph (IDAS denoted by G_s) at the end of the first and the second slot for the example network shown in Figure 2.1a.

(i) P_i interferes at AP_l; and, (ii) v_{ij} is not same as v_{kl}. The second condition forbids self-loops in the graph.

By creating vertices on the basis of transmission range and edges on the basis of the interference range, the algorithm is able to handle the differences between the transmission and interference range of practical radios. Figure 2.2a shows the dependence graph for the example network shown in Figure 2.1a at the end of slot 1. In the following discussion, we use G_d to denote the dependence graph.

Phase 2: Computing the maximum vertex-induced directed acyclic subgraph (IDAS): We prove (See Section A.2) that a set of client-AP transmissions can be decoded if and only if the vertex induced subgraph of the corresponding set of vertices in G_d is a DAG (Directed Acyclic Graph). For example, since transmissions $A \rightarrow AP_1$ and $B \rightarrow AP_2$ (See Figure 2.2a) are part of induced acyclic subgraph of the dependence graph, therefore both these transmissions can be suppressed at the end of slot 1. Thus, to compute the largest set of

suppressed nodes in this slot, the CS computes the maximum Induced Directed Acyclic Subgraph (IDAS) of G_d. Algorithms for computing the maximum IDAS have been proposed before [24]. Although, those algorithms take into account the first and the second requirements discussed in Section 2.2.1, but they do not take into account the other two requirements. To that end, we now explain the greedy algorithm used by CS to compute the IDAS (denoted by $G_s = (V_s, E_s)$) of G_d.

CS works by greedily adding vertices while ensuring that G_s stays acyclic. At any step, it may be possible to add any of the multiple vertices (each vertex corresponds to some client-AP pair in the dependence graph) available to G_s. However, different vertices may give different performance results. To that end, for each vertex that can be added to V_s, CS computes its *weight* and greedily adds that vertex that does not create a cycle in G_s and has the lowest weight (Lines 8-15 of Algorithm 1). This process is repeated until no more vertices can be added to G_s (Lines 6-17 of Algorithm 1).

Computing the weight: When adding a vertex (say v_{ij}) to G_s, the following equation is used to compute its weight:

$$\begin{aligned} & w_{ij} = w(P_i, AP_j) \\ = \; & |(v_{ij}, v_{kl}) : (v_{ij}, v_{kl}) \in E_d \wedge v_{kl} \in V_s| \\ & \times |(v_{kl}, v_{ij}) : (v_{kl}, v_{ij}) \in E_d \wedge v_{kl} \in V_s| \end{aligned} \tag{2.1}$$

If on adding v_{ij} to V_s, it has either fewer incoming edges or fewer outgoing edges, then this indicates that v_{ij} has low probability of being part of some cycle in the future. This increases the possibility of extending G_s to a larger acyclic induced subgraph in the following iterations. Further, fewer edges indicate that APs need to exchange fewer packets in the backbone. Thus, (2.1) assigns a lower weight to the vertices with either fewer incoming edges or fewer outgoing edges and thus, a higher priority. Therefore, by using (2.1),

Algorithm 1 tries to take all the requirements into account that were discussed in Section 2.2.1.

Figure 2.2b shows the computed IDAS for the dependence graph shown in Figure 2.2a. The subgraph shows that AP_1 suppresses A while AP_2 suppresses B. After the APs send the suppress, only C and D transmit in the second slot. Figure 2.2c shows the dependence graph created by CA at the end of the second slot. Figure 2.2d shows the computed IDAS which shows that C and D are suppressed in this slot. Since, no other transmitters are left for transmitting, so, the CS would ask APs to send *finish*. Observe that the edges in the G_s also reveal how the APs should exchange the decoded packets. For example, the directed edge from "$D \rightarrow AP_1$" to "$C \rightarrow AP_2$" indicates that decoding of C at AP_2 is dependent upon decoding of D at AP_1. Thus, AP_1 should send the decoded bits of D to AP_2. The CS transmits this information to the APs so that they only exchange packets that are required, thereby reducing the overhead on the backbone.

2.2.2 Practical Challenges

Although the algorithm proposed before is good in theory, it is difficult to implement due to various challenges:

- **Synchronization:** CA requires that all APs be time-synchronized to ensure correct correlation of PN sequences. As described in Section 4.3.1, receivers in CA use correlation of Gold codes to identify the set of transmitters. Gold codes guarantee that the circular cross-correlation of two instances of Gold code is bounded. For 127-symbol sequence Gold codes, the bound is 1/7 [28]. So, the correlation works better when Gold codes collide only with other Gold codes, and not with arbitrary samples from data packets. This implies that CA needs to ensure that when an AP receives a set of collided transmissions, then the PN sequences in them only collide with other PN sequences. One way to satisfy this requirement is to ensure that all

APs in the network poll and suppress in a synchronous fashion. Algorithms for synchronization in wired networks have been proposed before [41, 45], however, as the size of the network grows, ensuring synchronization across the entire network becomes difficult. Thus, a new method is needed to ensure that PN sequences collide only with other PN sequences so that they can be correlated with high accuracy.

- **Unpredictable delays over the backbone:** In CA, the central server computes the set of suppressed transmitters and distributes that result to the APs. However, due to unpredictable delay, it is possible that this result may not arrive in time and thus, some of the APs may not be able to proceed.

- **Non-uniform distribution of clients across APs:** The previous algorithm requires that all APs poll at the same time. In an enterprise network, it is possible that different APs have unequal number of neighboring clients with outstanding data. This may result in different APs finishing at different times, leading to lower throughput as all the finished APs in the network need to wait for the AP with the highest number of clients.

- **Absence of central controller:** As discussed in Section 2.1, a central controller may not be present to control all the APs in the network.

2.3 Symphony: Distributed algorithm

In this section, we explain the working of Symphony that takes into account the aforementioned challenges in Section 2.2.2.

2.3.1 System Structure

Using cooperative decoding requires multiple APs to collaborate, however, at the same time, it may not be possible for all the APs in the network to collaborate since the wired

backbone delays from APs to the central controller may be unbounded. Further, if the central controller is overseeing a lot of APs, it may also become a bottleneck. Symphony handles these contradicting requirements by dynamically placing different APs in different *groups*. One AP in each group serves as the Group Head (GH) and executes Symphony independently from other groups. Putting APs in different groups allows Symphony to achieve multiple objectives: (i) When dynamically creating groups, it is ensured that the average round trip delay on the wired backbone from the AP to the GH is no more than a threshold (ψ μs), thereby bounding the expected communication delay between the AP and the GH with high probability; (ii) If one AP has high number of clients, then only the APs in the same group are required to wait for this AP to finish; and, (iii) No central controller is required.

Implementing this dynamic group based structure leads to various **challenges**: (i) To avoid decoding failures due to conflicting *suppress* schedules, a client that is in the range of APs from different groups is not allowed to transmit (discussed below). Thus, it is important to dynamically change the composition of the groups so that all clients get an opportunity to transmit; and, (ii) The overhead of maintaining the groups should be minimized. The next subsection describes the group management algorithm used in Symphony.

2.3.2 Group Management

Group creation and joining an existing group: When a node (AP or client) turns *on* or whenever the node is not part of any group, it enters the *wait* stage where it waits to hear a *join* message on the wireless. This message includes the PN sequence of the GH that originated this message. In Symphony, the ID of a group is same as the ID of its GH. Therefore, the PN sequence included in *join* is also useful in determining which group a received message came from. If the node (say n_i) hears such a message, it joins the announced group if all of the following requirements are satisfied: (*i*) n_i is not in *recovery*;

(*ii*) n_i has no wireless neighbor n_j such that n_j is a member of some other group and is under recovery; and, (*iii*) If n_i is an AP, then the average round trip delay between n_i and the GH on the wired backbone is no more than a threshold ($\psi\mu$s). If n_i determines that it is eligible to join the group, then it immediately rebroadcasts the *join* message. Once n_i rebroadcasts the message, we say that n_i has joined the group and has entered recovery. On the other hand, if n_i is an AP and it does not hear any *join* message within a certain time (θ μs) that satisfies all the requirements, then it exits the wait stage. Next, if n_i checks if it has a wireless neighbor n_j such that n_j is a member of some other group and is under recovery. If n_i has no such neighbor, then n_i creates a new group and declares itself as the GH. Afterward, it broadcasts a *join* message.

Data transmission: A client broadcasts its data packet immediately after broadcasting the *join*. Once the APs receive the headers of the data packets transmitted by clients, the APs correlate the PN sequences in the headers to determine the IDs of clients that transmitted (See Section 4.3.1). This information is then sent by the APs to the GH that uses algorithm described in Section 2.2.1 to compute which AP will suppress which client. This result is then sent back by the GH to all the APs who use this result to suppress the clients at the end of the slot. However, due to unpredictable queuing delays on the backbone, it is possible that some APs do not receive the result back from the GH before the end of the slot. To handle such scenarios, the GH iteratively computes the set of nodes to suppress for the next k slots and sends that to all the APs (We explain in Section 2.6.2 how k is determined). Further, in the unlikely case when some AP does not hear back from GH for more than k slots, then at the end of the slot that AP does not suppress any client.

Finishing the recovery: As the APs suppress transmitters, eventually the number of packets transmitted will be small enough such that they can be decoded. This marks the end of recovery. At the end of recovery, the APs need to take a few steps to ensure: (i) GH is dynamically changed to ensure that all APs have equal chance of being the GH; (ii)

Members belonging to a small group have a non-zero probability of merging with a bigger group; (iii) Members belonging to a large group have a non-zero probability of splitting into multiple smaller groups; and, (iv) Next recovery starts as soon as possible. We now explain the steps taken by APs at the end of the recovery: (*i*) At the beginning of the last slot, the GH sends a message to the APs announcing that the recovery will finish at the end of this slot. This allows the APs to know when to enter the *wait* stage. (*ii*) Before the recovery is over, the GH randomly selects some other AP in the group to be the new GH. The current GH informs this new potential GH of this decision. This step ensures that all APs have equal chance of being the GH. (*iii*) Once the recovery is over, all APs enter the *wait* stage for a duration of θ μs as described in the beginning of Section 2.3. For all the APs, θ is inversely proportional to the number of APs in their group. This ensures that smaller groups wait for a longer time to receive a *join* message from other groups, and thus they have higher probability of merging into another group. Also, the value of θ is set to be low for the selected GH while it is higher for the other APs. This ensures that the selected GH has higher chances of becoming a GH. This also reduces the probability of multiple APs independently declaring themselves as GH. This allows Symphony to set θ to a low value, thereby reducing the overhead of group management.

2.3.3 Cyclic Padding

As discussed in Section 2.3.2, on receiving the *join* message, a client first rebroadcasts the message and then its uplink data. Consider the example network shown in Figure 2.3. Lets say Alice starts transmitting *join* message at t = Tμs. join message is a 127-symbol PN sequence, so, at 20Mbps, it will take 6.35μs [55] to transmit. Further, due to hardware constraints, it may take Alice up to 2μs more before it can transmit the data packet. Thus, Alice will start transmitting data packet somewhere between $t = T + 6.35\mu$s and $t = T + 8.35\mu$s (See Figure 2.3). AP_1 upon hearing the join message would immediately

join transmission start time	T	[T + 6.35, T + 9.35]	[T + 6.35 + 6.35, T + 9.35 + 9.35] = [T + 12.70, T + 18.70]
data transmission start time	[T + 6.35, T + 8.35]		[T + 12.70 + 6.35, T + 18.70 + 8.35] = [T + 19.05, T + 27.05]

Figure 2.3: Synchronization: The propagation delay is at most 1μs, the length of 127 symbol PN sequence transmitted at 20Mbps is 6.35μs while the radio tx-rx turn around time is at most 2μs for a total duration of 9.35μs [55]. For a given AP, the maximum difference between the start of the data transmission times of its neighboring clients is at most 20.70μs

rebroadcast it. However, due to propagation delay (at most 1 μs [55]) and radio tx-rx turn around (at most 2 μs [55]), it may happen as late as $t = T + 9.35\mu$s. Continuing this computation, we can see that all neighboring clients of AP_2 will start transmitting before $t = T + 27.05\mu$s. Thus, for a given AP, the maximum difference between the start of the data transmission times of its neighboring clients is at most 20.70μs (See Figure 2.3 where Alice can possibly start transmitting data as early as $t = T + 6.35\mu$s while Don may start as late as at $t = T + 27.05\mu$s). Thus, to ensure that the PN sequences in their data packet headers collide only with the PN sequences of other transmitters, the clients pad cyclic bits after their PN sequences. In Symphony, the length of the cyclic padding is set to 25μs.

By using circular padding and *join* messages broadcasted on wireless, Symphony ensures that it is possible to correlate the PN sequences without requiring global synchronization. By requiring GH to proactively compute the set of suppressed transmissions, Symphony is able to handle unpredictable and non-zero latency on the backbone.

2.4 Heterogeneous Data Rates and Packet Sizes

In Symphony, different transmitters may select different physical layer data rates due to differences in channel conditions. Since wireless channels are bidirectional in nature [30], the transmitters in Symphony use the received poll packet to estimate the channel to the receiver. This is similar to the Mozart's (See Chapter 4) approach of estimating the physical layer data rate. However, in Symphony, a client may receive poll from multiple APs. In that case, it estimates the channel with all the APs from whom it received the poll message and chooses the maximum possible data rate such that at least one AP can decode the transmitted packet. Estimating the channel requires the client to estimate the RSS of as many poll packets as possible. However, estimating the RSS of multiple colliding packets is not trivial due to high interference. For this, Symphony borrows the channel estimation technique proposed in Mozart for estimating the RSS of colliding packets (Section 4.3.1).

In Symphony, the channel access time is fixed for all transmitters. A transmitter with better channel may transmit more bits than a transmitter with worse channel. However, it is possible that some transmitter may have good channel but not enough pending data to make complete use of the channel access time. This may lead to wastage of the channel. To minimize the channel wastage, such a transmitter in Symphony will reduce its data rate such that its transmission still fits in the slot size. This reduced data rate sometimes allows the receiver to simultaneously decode two transmissions using Successive Interference Cancellation (SIC) [70]. When using SIC, APs in Symphony decode two packets in the same slot and can suppress up to two transmissions in a single slot.

To implement SIC, the receivers in Symphony need to estimate the RSS of each of the colliding transmitters. For this, we borrow the idea proposed in Mozart whereby PN sequences are used to estimate the RSS. Using SIC in Symphony requires making some changes to the way we construct the dependence graph:

- In the dependence graph, instead of a vertex being a duplet, it can now be a triplet such that vertex $v_{ijk} = \{P_i, P_j, AP_k\}$ implies that AP_k can decode packets from both P_i and P_j in a single slot using SIC where P_i is the packet with higher RSS at AP_k. Such a vertex is added only if on the basis of estimated RSS and the physical layer data rates used, AP_k determines that it can decode P_i and P_j simultaneously using SIC.

- A directed edge is drawn from vertex v_{ijk} to another vertex $v_{lmn} = \{P_l, P_m, AP_n\}$ *iff* either P_i interferes at AP_n or P_j interferes at AP_n. Similarly, a directed edge is drawn from v_{ijk} to vertex $v_{ln} = \{P_l, AP_n\}$ *iff* either P_i interferes at AP_n or P_j interferes at AP_n. Finally, a directed edge is drawn from v_{ln} to v_{ijk} *iff* P_l interferes at AP_k.

Figure 2.4 shows the dependence graph for the network topology shown in Figure 2.1a when using SIC. As before, to compute the set of suppressed transmissions, the GH picks an Induced Directed Acyclic Subgraph (IDAS) of the dependence graph. If the subgraph has vertices with three tuples, then the corresponding AP needs to suppress two transmitters. To maximize the probability of invoking SIC, Algorithm 1 is modified such that when greedily adding a vertex (Lines 8-15 of Algorithm 1), the GH first only considers the vertices with three tuples. If no such vertex is found, only then the GH adds the vertices with only two tuples.

2.5 Discussion

In this section, we discuss some issues that need deeper investigation to make our solution more practical.

- **Achieving gains on downstream traffic:** In EWLANs, it is generally easier to avoid collisions in downstream data as the APs are aware of which AP has data to

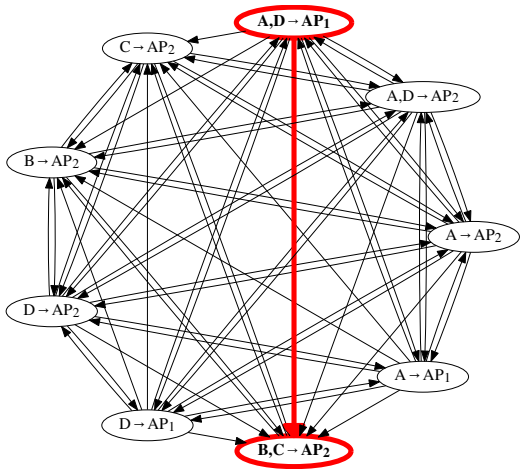


Figure 2.4: Dependence graph for network topology shown in Figure 2.1a when using SIC. The chosen acyclic subgraph is also shown in bold.

send to which client. Multiple algorithms have been proposed in the literature that use this information to schedule downlink transmissions [49, 65, 72]. Symphony, on the other hand, improves performance for upstream data while allowing coexistence of downstream traffic.

In Symphony, due to differences in the distribution of clients, it is possible that for some AP, all its clients are suppressed sooner compared to clients of other APs. Symphony allows these *free* APs to transmit downlink traffic to their clients while other APs keep receiving uplink traffic. For the example network shown in Figure 2.5, once AP_2, AP_3 and AP_4 become free, they can start transmitting downlink traffic to their neighboring clients. The downlink transmission from AP_2 to P_2 will be successful while the transmission from AP_3 to P_3 will not be successful since it is interfered by P_1's uplink traffic. The transmission from AP_4 to P_4 will also be successful. Although, this downlink transmission interferes with the uplink transmission at AP_1, it is still possible for AP_1 to decode P_1. For this, Symphony requires AP_4 to send its

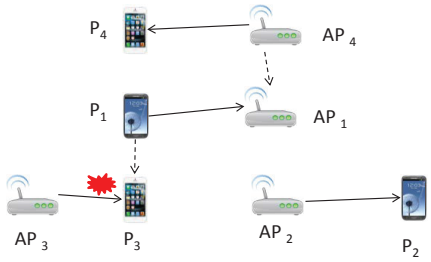


Figure 2.5: Downlink-uplink coexistence.

downlink packet to AP_1 on wired backbone. Using that, AP_1 recreates the samples of AP_4's downlink packet and subtracts them from the received samples. Thus, in Symphony, the downlink traffic can coexist with the uplink traffic without causing interference to it.

Once all the APs in a group become free (i.e., they have transmitted *finish*), then the GH schedules interference-free downlink transmissions as discussed before in [49]. Apart from [49], it is also possible to use other techniques proposed in the literature [65, 72] for achieving throughput gains on downlink traffic. Observe that, unlike other algorithms [49, 72], Symphony does not require any knowledge of the network topology. This reduces the overheads in Symphony compared to other algorithms.

- **Packet subtraction:** In order to decode collided packets, APs in Symphony perform packet subtraction by re-creating the samples. To increase the accuracy of subtraction, the APs need to correct for the phase, frequency and sampling offsets [29]. This problem of packet subtraction in presence of interference has been well studied [29]. In our implementation, we use the scheme proposed in ZigZag [29] as it gave the best results.

- **Sending ACK:** In Symphony, APs exchange the data packets among themselves on the backbone. However, this exchange may take some time due to the wired

latency. In order to avoid wasting the channel during the decoding process, the APs immediately send *finish* to the clients without actually decoding the data packets. This allows clients to transmit new data packets when they receive a poll. While the next recovery is going on, the APs exchange the data packets on the wired backbone. Upon successful decoding, the APs send the *acks* to the different transmitters at the end of the next recovery period (during the *wait* period).

- **Handling decoding errors:** After subtracting a packet, a small amount of *residual noise* (See Chapter 4) is left due to inaccuracies in correcting for the different offsets. If too many packets collide at the same AP, then as the AP cancels packets one-by-one, the residual noise builds up. This may result in an AP being unable to decode the packet of interest due to failed checksum. Further, this may also cause decoding error for the transmissions that are dependent on this transmission. In that case, the AP has two options: (*i*) Different APs may combine frames to correctly decode the received packet [56]. This is particularly useful in dense AP networks where multiple APs (including those who did not suppress any transmitter) may hear the transmission from the same client. (*ii*) If the previous option is not successful, then the APs send *ack* to only the transmitters that were decoded successfully.

- **Mobility and fading:** It is possible that during recovery, a client may move and its set of neighboring APs may change. Symphony can handle such scenarios as clients in Symphony do not associate with any specific AP. However, if during the recovery, the client moves out of the range of *all* the APs in the group, then this may cause decoding error. In that case, the APs will use the mechanisms discussed above to handle the decoding error. It is also possible that during the recovery, due to fast-fading, some packet (say P_i) is temporarily not received by some of its neighboring

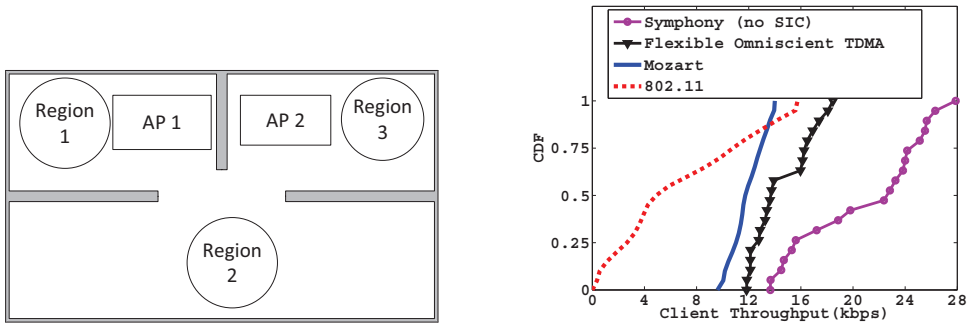
APs. This does not cause any decoding error since in Symphony, the set of suppressed transmissions is computed only on the basis of transmissions received in that slot. So, in future when the APs receive the packet again, then it will be suppressed.

2.6 Experiments

In this section, we describe the results from our experiments performed on the GNU radio platform and Universal Software Radio Peripheral (USRP) N210 version 4 radios with WBX daughterboards. In our experiments, we used BPSK with 1/2 convolutional code and the channel frequency was set to 1078 MHz. The phase offset, frequency offset and the sampling offset were canceled as described in Section 2.5 while RSS estimation and transmitter identification were done as described in Section 4.3.1 and Section 2.4.

Besides Symphony, we also implemented a version of the IEEE 802.11 protocol. One of the challenges in implementing 802.11 was that USRP has different hardware parameters compared to the commercial 802.11 cards. So, similar to Mozart, we re-measured the optimum values of all 802.11 parameters (SIFS, DIFS, slot size, ACK timeout) for the N210 hardware. The transmission data rate was kept low (62.5 Kbps) to ensure that the delay between the host computer and the radio was *small* compared to the packet length so that the relative delay values would approximately match the values from off-the-shelf wireless cards.

Apart from implementing our algorithm and IEEE 802.11, we also implemented two other algorithms: (i) **Mozart** (See Chapter 4); and, (ii) **Flex Omniscient TDMA**: Here, the clients do not associate with any particular AP but transmit data to and receive data from any of the neighboring APs. The schedule is greedily computed by a central omniscient scheduler. Two links are scheduled at different times *iff* transmitter of one of the link interferes at the receiver of the other link. The knowledge of which nodes have data to send



(a) USRP Testbed

(b) CDF plot across different topologies

Figure 2.6: Experiment testbed and results

to which other nodes, as well as the complete network topology is provided out-of-band (i.e., without any overhead) to the Flex-TDMA central scheduler and is 100% accurate.

2.6.1 Results

In our experiments, we placed two USRP APs in two different rooms (as shown in Figure 2.6a). Four USRP clients were placed at different locations in the three regions so as to create ten different topologies (hidden, non-hidden, mix etc.). In all the topologies, for 802.11 and Mozart, the clients associated to the AP with the strongest signal. Figure 2.6b shows the CDF of the throughput of clients for the four algorithms. Averaging over all topologies, Symphony (no SIC), Omniscient TDMA, Mozart (no SIC) and IEEE 802.11 provide a per node throughput (in Kbps) of 20.7, 14.4, 11.9 and 7.2, respectively. Thus, Symphony (no SIC) has an average throughput of 43%, 74% and 187% higher than Omniscient TDMA, Mozart (no SIC) and IEEE 802.11, respectively.

2.6.2 Measurements

Apart from implementing Symphony on the USRP testbed, we also made extensive measurements on the Ohio State University EWLAN. These measurements help us in better designing the algorithm and are also fed to the simulator (See Section 2.7). The measurements were conducted on a weekday between 10am-4pm (worst case analysis) with the routers and switches also serving the traffic from other desktops and APs on the EWLAN. All the switches and routers on the EWLAN had 1 Gbps interface speed. We measured the round trip time (RTT) between pairs of APs by transmitting and echoing back packets between multiple pairs of APs. The size of the packets was varied from 4-100 bytes to approximate the size of the control packets that would be transmitted from APs to the GH. Table 2.1 shows the result of the measurements. Using the three-sigma rule, we conclude that even with 5 hops of distance between a pair of APs, with 99.7% confidence, the RTT between the two APs will be less than 2914.93 μs (Mean + 3 times standard deviation).

So, the number of slots for which the GH proactively computes the set of suppressed transmissions (k as discussed in Section 2.3.2) can be set such that $k \times T \geq 2914.93\mu$s where T is the channel access time (maximum duration of a data packet as discussed in Section 2.5). In our experiments and simulations, we computed T as $\frac{1500\ bytes}{6\ Mbps} + \delta$ where δ is fixed time required for sending the data packet's header. Value of k was set to 3.

2.7 Simulations and Evaluations

In this section, we explain the results from our ns-3 based trace driven simulations.

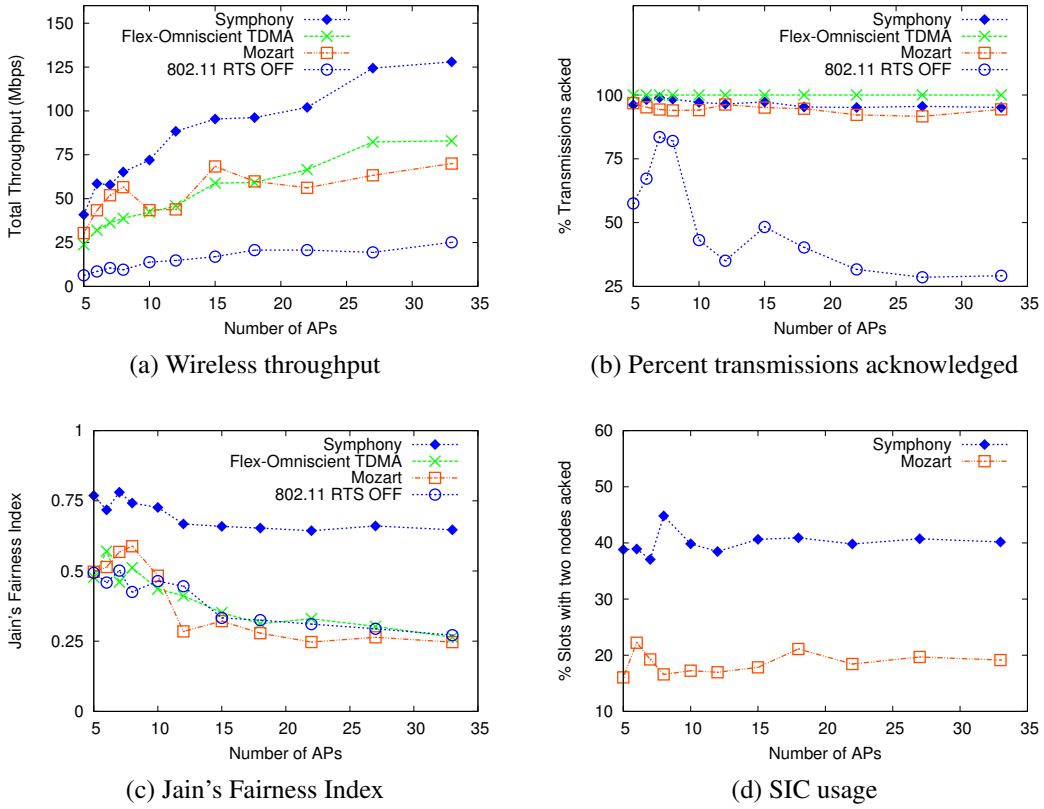
2.7.1 Setup

In our ns-3 based simulations, we randomly placed varying number of APs in a 750m $\times$ 750m field. The number of clients in the network were 5 times the number of APs. Each

client associated with the AP from which it received the strongest signal. To generate traffic in the simulator, TCP connections were established between each client and its AP. For this, from the previously collected traffic traces during SIGCOMM [67], we computed the pdf distribution of packet sizes and also the pdf distribution of packet inter-arrival time over all connections. These two pdf distributions were then used to generate both uplink and downlink traffic in the simulation. To create network saturation condition, the number of connections between each client and its associated AP was set to 20.

Further, the results from the experiments were fed into the simulator as follows: (i) **PN-Sequence detection and RSS estimation accuracy**: The accuracy of correlating a PN sequence and estimating the transmitter's RSS depends on its SINR (See Chapter 4). In the simulations, we used the values from Mozart to model the PN sequence detection accuracy and the error in RSS estimation; (ii) **Residual noise level**: The level of the noise left after packet subtraction depends on the SINR of the canceled packet. In simulations, we fed the power of the residual noise using the data shown in Chapter 4; and, (iii) **Delays on backbone:** In our simulation, the APs were connected with each other through switched wired Ethernet. To every switch, we connected three APs that were geographically close to each other. These switches were then connected to a router. The delay between two APs separated by k hops, was randomly generated using the collected data as discussed in Section 2.6.2.

To compare the performance of Symphony with state of the art, we implemented various other algorithms: (i) **Mozart**; (ii) **IEEE 802.11 without RTS-CTS**; and, (iii) **Flex Omniscient TDMA** as discussed in Section 2.6. The channel access time in both Mozart and Symphony was fixed to $\frac{1500\ bytes}{6\ Mbps} + \delta$ where δ is fixed time required for sending the data packet's header. In 802.11, the transmitters follow the data rate adaptation algorithm proposed in [48], while in Mozart and Symphony, the transmitters estimated the channel using the poll packet as described in Section 2.4.



(a) Wireless throughput

(b) Percent transmissions acknowledged

(c) Jain's Fairness Index

(d) SIC usage

Figure 2.7: Simulation results.

2.7.2 Results

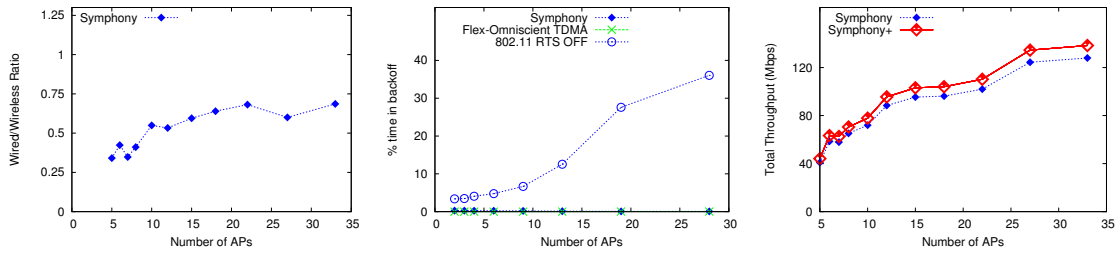
Throughput: Figure 2.7a compares the throughput of different algorithms. On an average, throughput of Symphony was observed to be 63% and 58% higher than TDMA and Mozart, respectively. Further, average throughput of Symphony was 5.6x compared to 802.11. Higher throughput of Symphony can be attributed to two factors: First, cooperative decoding as shown in Figure 2.1c allows APs in Symphony to simultaneously receive multiple packets. Secondly, when the transmitters have fewer data to send, then it results in throughput loss for TDMA that uses fixed slot length. Although, Symphony also uses fixed

slot length, however, by using SIC to decode multiple packets, a single AP in Symphony is able to decode multiple packets simultaneously.

To further explore the throughput increase, we also plotted the percentage of data transmissions that were acknowledged by the receiver (See Figure 2.7b). This number is high for Symphony while for IEEE 802.11, the percentage of packets acknowledged is very low due to hidden terminal problem. Similar to Symphony, the percentage packets acknowledged for TDMA and Mozart were also close to 100%. Symphony had a slightly higher packet acknowledgment rate compared to Mozart since in the case of decoding failure in Symphony, multiple APs exchange frames to cooperatively decode a packet (See Section 2.5). In 802.11, the packet acknowledgment rate peaks as the number of APs and clients increase and then gradually decreases. This is because with increasing density of APs, it becomes possible for the clients to have stronger channel with the APs, thereby mitigating the effect of hidden terminal (due to data rate adaption and closeness to the AP). However, with increasing density, the probability of two clients picking the same backoff value and the probability of a hidden terminal also increases which decrease the packet acknowledgment rate. Due to these contradicting factors, the packet acknowledgment rate peaks when number of APs is 7.

Figure 2.7c shows the Jain's fairness index for different algorithms. Symphony has higher fairness since it allows all clients to participate. In other algorithms (including Flex Omniscient TDMA), if a client is in the range of multiple APs, then it may not get a chance to transmit to its AP since the channel in its neighborhood is very likely to be busy.

Figure 2.7d compares the percentage slots where SIC was used. Symphony uses SIC in more slots than Mozart since in Symphony, the AP can use SIC to suppress any pair of transmitters in its neighborhood. This increases the probability of finding a pair of transmitters that are eligible to be decoded simultaneously. Further, APs in Symphony also



(a) Wired overhead for Symphony.

(b) Time spent in backoff

(c) Symphony+ has zero wired latency

Figure 2.8: Simulation results: Overheads.

cooperate to use SIC to simultaneously transmit two packets to a single client resulting in higher applicability of SIC [70].

Wired overhead: In Symphony, APs exchange packets over the wired backbone. Theoretically, the number of unicast packets exchanged on wired backbone can be as many as $\frac{N\times(N-1)}{2}$-times the number of packets decoded on wireless (where N is the number of APs in the network). However, Algorithm 1 minimizes this overhead by reducing the number of edges in the subgraph. Figure 2.8a shows the overhead as ratio of unicast data sent over wired over the wireless goodput. This ratio increases with increase in number of APs as expected but it plateaus as the number of APs go beyond a certain limit. This is because the broadcast nature of the wireless does not allow every AP in Symphony to suppress a packet. Thus, even with increase in APs, the number of APs that suppress at least one client do not increase, thereby also limiting the number of packets exchanged on the backbone. Further, at any time the ratio is less than 1 implying that for every packet decoded on wireless, at most 1 unicast transmission occurs on the wired backbone.

Backoff Overhead: APs in Symphony spend some time in backoff (or timeout) between every recovery. The amount of time spent is given by $\frac{100}{n}$ where n is the number of APs in the group (See Section 2.3). Figure 2.8b shows the percent time spend by APs in

backoff. APs in Symphony spend less than 1% time in backoff. In Symphony, the backoff overhead is amortized over the length of the recovery period. Further, unlike 802.11, the backoff in Symphony stays constant irrespective of the collision rate.

Effect of wired Ethernet latency: Due to the unpredictable latency on the wired Ethernet, the APs in Symphony proactively compute the schedule for the next 3 slots. To compute the effect of this latency on the optimality of the suppress schedule, we implemented a version of Symphony (called Symphony+) with latency on backbone set to zero. Figure 2.8c shows that throughput of Symphony+ is within 5% of Symphony indicating that the advance schedule computed by Symphony is close to optimal.

2.8 Related Work

Backbone usage: The idea of using the wired backbone to increase wireless throughput is not new. In IAC [30], every node is assumed to have at least two antennas. The APs direct the transmitters in the network to align their transmissions such that the APs can cooperatively decode multiple packets. However, unlike [30], Symphony does not assume the presence of multiple antennas. In MegaMIMO [65], multiple APs cooperatively precode the transmissions such that each client receives only the packets intended for it while the other transmissions cancel-out. However, MegaMIMO requires that transmitters exchange decoded packets among themselves and thus, it works only for the downlink transmissions. On the other hand, Symphony improves the throughput for the uplink traffic.

In Epicenter [32], authors propose that APs should exchange coarse representation of symbols to decode corrupted bits. Similarly, authors in [56] also propose that APs exchange bits for decoding corrupted packets while Woo *et al.* [79] propose that APs should exchange raw samples. In all these algorithms, the APs cooperate to decode the same packet whereas in Symphony, APs encourage transmitters to collide and then cooperate to decode multiple packets simultaneously without exchanging the raw samples.

Other MAC algorithms: Algorithms proposed in [15, 57] focus mainly on client-AP association, channel assignment and power control. Authors in [49] propose implementing a backpressure based algorithm for downlink traffic. Similarly, CENTAUR [72] uses a central controller to reduce throughput loss due to hidden terminal and exposed terminal problems. However, their solution is also applicable only to downlink traffic. MiFi [14] also uses centralized coordination of APs and focuses on ensuring fairness. Contrary to the algorithms in [14, 19, 49, 72] that aim to decrease packet collisions, Symphony encourages transmitters to collide and then suppresses the packets so that all the colliding packets can be cooperatively decoded in minimum slots.

In contrast to Mozart proposed in Chapter 4, Symphony focuses on leveraging communication in the wired backbone for cooperative decoding of collided packets. This leads to a different set of challenges as discussed in Section 2.1.

2.9 Conclusions

In this chapter, we proposed Symphony, a novel backbone-based approach for cooperative packet decoding in Enterprise WLANs. We also designed an algorithm that computes the set of suppressed transmissions. To take into account the practical challenges of high latency, absence of synchronization and a central controller, we proposed a dynamic group based distributed algorithm. To reduce the throughput loss due to variations in packet lengths, we proposed that transmitters reduce their physical layer data rate while receivers use SIC to decode multiple packets simultaneously. Experiments conducted on the USRP testbed show that Symphony provides a throughput of 43% and 187% higher than Omniscient TDMA and IEEE 802.11, respectively. Trace-driven simulations show that, throughput of Symphony is up to 1.63x compared to omniscient TDMA and 5.6x compared to IEEE 802.11.

Algorithm 1: Computes the set of nodes to be suppressed in a given slot

Input: For every packet P_i received in this slot by each AP AP_j, the transmitter of P_i (identified using PN sequence).

Output: The acyclic subgraph showing: Set of packets (or corresponding transmitters) that should be sent suppress in this slot, which AP decodes which packet and how APs share decoded packets among themselves.

`// Phase 1: Computing dependence graph` $G_d = (V_d, E_d)$

$V_d \leftarrow \{v_{ij} : P_i$ received in this slot and AP_j can decode P_i in absence of external interference$\}$

$E_d \leftarrow \{(v_{ij}, v_{kl}) : v_{ij} \in V_d$ and $v_{kl} \in V_d$ and P_i interferes at AP_l and $v_{ij} \neq v_{kl}\}$

`// Phase 2: Computing largest IDAS`

$V_s \leftarrow \phi, E_s \leftarrow \phi, G_s \leftarrow (V_s, E_s)$

while *true* **do**

  $v^* \leftarrow \phi, w^* \leftarrow \infty, E^* \leftarrow \phi$

  for $v_{ij} \in V_d \backslash V_s$ **do**

    $E_{ij} \leftarrow \{(v_{ij}, v_{kl}) : (v_{ij}, v_{kl}) \in E_d \wedge v_{kl} \in V_s\}$

    $E_{ij} \leftarrow E_{ij} \cup \{(v_{kl}, v_{ij}) : (v_{kl}, v_{ij}) \in E_d \wedge v_{kl} \in V_s\}$

    if $w_{ij} \leq w^*$ *and graph with* $V_s \cup \{v_{ij}\}$ *and* $E_s \cup E_{ij}$ *is acyclic* **then**

      $w^* \leftarrow w_{ij}, v^* \leftarrow v_{ij}, E^* \leftarrow E_{ij}$

  if $v^* \neq \phi$ **then**

    $V_s \leftarrow V_s \cup \{v^*\}$`// Add` v^* `to` G_s

    $E_s \leftarrow E_s \cup E^*$

  else

    break

return G_s

Table 2.1: Results from RTT measurements

One-way distance (in hops)	**Devices in between**	**Mean (in μs)**	σ **(in μs)**
2	One switch	117	91.90
4	2 switches + 1 router	829	182.96
5	2 switches + 2 routers	2122	264.31

CHAPTER 3

ROBINHOOD: THROUGHPUT SCALING IN DENSE ENTERPRISE WLANS WITH BLIND BEAMFORMING AND NULLING

3.1 Introduction

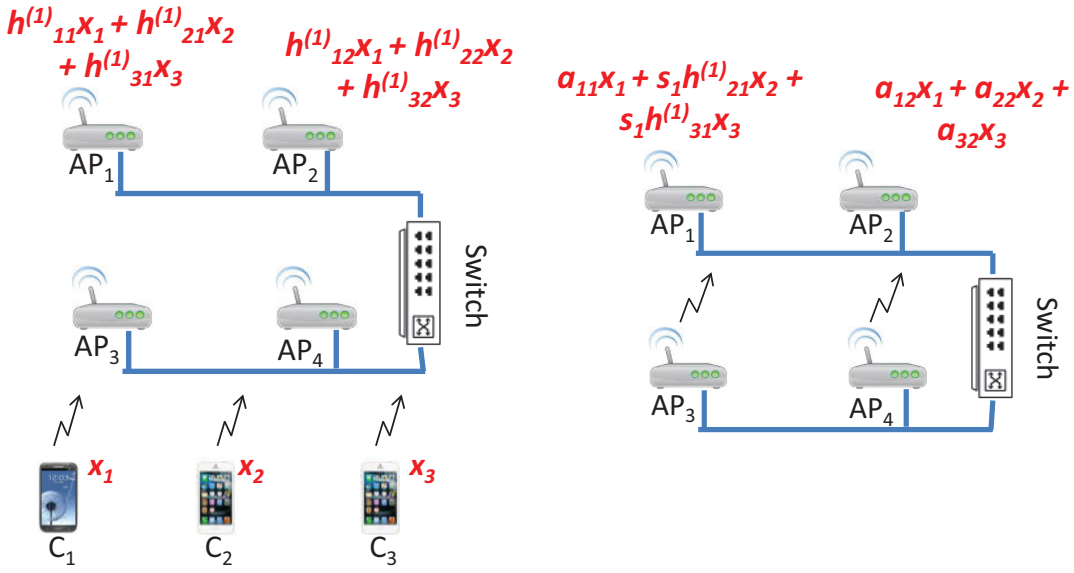
The recent explosive growth in the number of mobile devices and the data generated by these devices has led to a decrease in the channel resources available to each individual device. Network administrators have tried to tackle this problem by densely deploying access points so that users can almost always find a close by AP with good signal strength. However, dense deployment of APs does not scale well with the throughput demands. In the existing network protocols [57, 71], when one mobile client is transmitting uplink packets to an access point, the nearby clients have to remain silent to avoid interference to the ongoing transmission.

Recently, multiple algorithms have been proposed that help in scaling the throughput with number of wireless devices. **Interference Alignment (IA)** [16] is one of such techniques that requires clients to participate in a schedule with exponential number of slots. However, mobile clients are really mobile. They may not stay at the same place for a long time. **Multi-User MIMO (MU-MIMO) [27]** enables scaling of throughput with number of devices, but it requires APs to exchange samples over the backbone. Although, the wired backbone in Enterprise Wireless LANs (EWLANs) is underutilized [11, 30], exchanging

samples requires significantly higher bandwidth compared to exchanging packets which cannot be supported by current wired networks [30, 32]. **Joint beamforming** based algorithms such as [47, 65] work only for the downlink traffic. To perform joint beamforming, these algorithms require all transmitters to share the contents of all packets to be transmitted. However, mobile devices are not connected through a wired backbone, and are unable to share the packets amongst each others.

This chapter proposes RobinHood, the first implementation of Blind Beamforming and Nulling scheme that enables multiple nearby access points to concurrently receive uplink packets from multiple mobile clients, all within a single collision domain without overwhelming the backbone. RobinHood does not increase energy consumption on the clients and executes exactly over two time slots. RobinHood leverages three properties that are unique to EWLANS: (i) *Dense deployment of APs (See Fig. 3.3 and [57])*; (ii) *Capability of these APs to exchange packets with each other over the underutilized wired backbone*; and, (iii) *Immobility of APs resulting in relatively stationary channels* (See Fig. 3.2). When one AP is receiving uplink data, existing algorithms [57] including IEEE 802.11 Wi-Fi, suppress nearby APs to transmit or receive data. In contrast, RobinHood makes use of the energy-rich access points to assist their clients (mobile devices) in decoding their packets at their respective access points. In RobinHood, the clients only participate in the first slot and the access points participate for the clients in the second slot.

Consider the example enterprise WLAN shown in Fig. 3.1a where all the APs and the three clients are in a single collision domain. Assume that the three users want to upload one packet each to the backbone. An omniscient TDMA scheduling algorithm with global knowledge would require three time slots to complete this upload. In RobinHood, in the first slot as shown in Fig. 3.1a, all users will transmit at the same time. All the 4 APs will receive a combination of three transmitted packets. In the second slot, AP_3 and AP_4 will retransmit the received signals by first precoding [37] them such that the following



(a) First slot. x_1, x_2 and x_3 are the three packets transmitted by C_1, C_2 and C_3, respectively. h^1_{ij} is the channel from client i to AP_j during time slot 1.

(b) Second slot. A subset of APs transmit in the second slot while the rest of the APs receive. a_{ij} are the final channel coefficients after the transmission of the second slot. s_i is the scaling coefficient at AP_i.

Figure 3.1: Illustration of RobinHood over a topology of 3 clients and 4 APs. All devices belong to the same collision domain and can hear each other.

condition is satisfied as shown in Fig. 3.1b: At AP_1, samples corresponding to x_2 and x_3 in the second slot align with the samples corresponding to x_2 and x_3 in the first slot. Decoding happens in multiple steps as follows:

- At the end of the second slot, AP_1 scales the samples received by AP_1 in the second slot and subtracts them from the samples received in the first slot. This scaling is done such that samples corresponding to x_2 and x_3 are *nulled*. Afterwards, it is left with only the samples corresponding to x_1. AP_1 decodes the samples to obtain the packet transmitted by C_1. Next, it transmits the decoded packet over the backbone to AP_2.

- AP_2 recreates the samples corresponding to x_1 and subtracts them from the samples received in the first slot and the second slot.

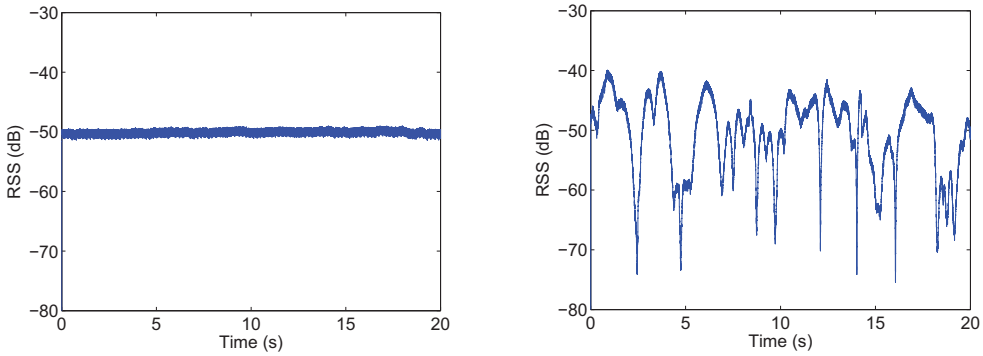
- After subtraction, AP_2 is left with two equations (one from each slot), and two variables (x_2 and x_3). AP_2 solves the two equations to obtain x_2 and x_3.

- Afterwards, AP_1 and AP_2 forward x_1, x_2 and x_3 towards their destinations.

RobinHood enables the three transmitters with single antenna to upload three packets in two slots, improving the throughput by 50% compared to omniscient TDMA. In Section 3.2, we show that in networks with high enough density of APs, RobinHood enables N mobile clients to transmit N uplink packets in exactly two slots resulting in unbounded throughput. Also, note that RobinHood requires the APs to exchange only the decoded packets instead of the raw samples.

The focus of RobinHood is to increase throughput of the uplink traffic for clients with single antenna. This is in contrast with [47, 65] that focus on downlink traffic. Recently, uplink traffic [11,32] has been growing at a fast rate due to the emergence of wide-range of applications, such as cloud computing, video conferencing, online gaming, VoIP, and traffic generated from mobile devices (e.g., location information or sensor readings). RobinHood makes extensive use of the wired backbone. Besides transmitting the decoded packets, the channel state information, which are required to do nulling in the second slot, are also exchanged over the backbone. Since RobinHood migrates most of the complexity from the mobile devices to the APs, it allows RobinHood to work even when the channel from clients to APs is rapidly changing due to client mobility. RobinHood works as long as the APs are time-synchronized with each other and it places very few requirements on the clients. This chapter makes the following contributions:

- We propose a blind beamforming and nulling scheme, RobinHood, that scales uplink



(a) Channel between a pair of APs

(b) Channel between a mobile client and an AP

Figure 3.2: Received Signal Strength (RSS) in an office environment. The channel between APs is relatively stationary compared to channel between AP and mobile client.

throughput with the number of access points. RobinHood also works over multiple collision domain.

- This thesis shows the first implementation of blind beamforming and nulling on USRP radios. Experiments performed on our testbed show that RobinHood achieves $1.48\times$ throughput compared to omniscient TDMA.

- Trace-driven simulation results show that in a large Enterprise WLAN, RobinHood can leverage the density of the access points. In EWLANs with high density of APs, RobinHood provides a throughput of $5.2\times$ compared to omniscient TDMA and $52.4\times$ compared to IEEE 802.11.

3.2 Illustration

Before discussing RobinHood in detail, we define a few notations. All of the clients and APs in RobinHood are assumed to have only one antenna. The network consists of clients

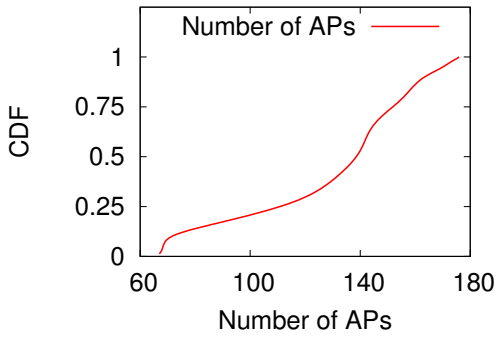


Figure 3.3: CDF of number of APs observed across different locations. The data was collected at multiple places including a hospital, a large university library and an apartment complex.

C_1, C_2 and C_3 and four APs from AP_1 to AP_4 that are connected through a wired backbone. Let $h_{ij}^{(1)}$ be the channel state information between C_i and AP_j in slot 1. In the second slot, a subset of APs are selected to transmit. For this example, this set consists of AP_3 and AP_4. Let $h_{kj}^{(2)}$ be the channel state information between AP_k and AP_j in slot 2. In this section, we assume that all the wireless devices are in single collision domain (*i.e.*, they can all hear each other). In Section 3.5, we extend RobinHood to networks with multiple collision domains. Let x_i be the packet sent by C_i in slot 1. In the following discussion, we ignore the presence of noise since it is not possible to null the noise. However, we do take noise into account in our analysis (See Section 3.4.4) and then later in our simulations (Section 3.7). Let $y_{ik}^{(t)}$ be the component of x_i received by AP_k in slot t. We have:

$$y_{ik}^{(1)} = h_{ik}^{(1)} x_i \tag{3.1}$$

Let v_k be the precoding vector for AP_k in the second slot and M be the total number of APs (In this example, we have $M = 4$). Let $y_{ij}^{(2)}$ be the component of x_i received by AP_j

in slot 2. We have:

$$y_{ij}^{(2)} = \sum_{k=3}^{M} h_{kj}^{(2)} v_k y_{ik}^{(1)} = \sum_{k=3}^{M} h_{kj}^{(2)} v_k h_{ik}^{(1)} x_i \tag{3.2}$$

We want to ensure that components of x_2 and x_3 at AP_1 are a linear combination of their components in the first slot. Let s_i be the scaling coefficient at AP_i. Thus,

$$y_{21}^{(2)} = \sum_{k=3}^{M} h_{k1}^{(2)} v_k h_{2k}^{(1)} x_2 = s_1 y_{21}^{(1)} = s_1 h_{21}^{(1)} x_2 \tag{3.3}$$

$$y_{31}^{(2)} = \sum_{k=3}^{M} h_{k1}^{(2)} v_k h_{3k}^{(1)} x_3 = s_1 y_{31}^{(1)} = s_1 h_{31}^{(1)} x_3 \tag{3.4}$$

Simplifying these equations, we get

$$\sum_{k=3}^{M} h_{k1}^{(2)} v_k h_{2k}^{(1)} - s_1 h_{21}^{(1)} = 0 \tag{3.5}$$

$$\sum_{k=3}^{M} h_{k1}^{(2)} v_k h_{3k}^{(1)} - s_1 h_{31}^{(1)} = 0 \tag{3.6}$$

Since, the right sides of Eqs. 3.5 and 3.6 are all 0, instead of 2, at least 3 variables are required to obtain non-zero solutions. One of these variables is the scaling coefficient (s_1). Thus, a total of 2 transmitting APs are required to supply these variables. Further, two receiving APs are also required such that the first AP decodes x_1 while the second AP decodes x_2 and x_3. Thus, in total $M = 2 + 2 = 4$ APs are required to support 3 clients as in Fig. 3.1.

In RobinHood, for the network shown in Fig. 3.1, at the end of slot 1, AP 3 and AP 4 solve Eqs. 3.5 and 3.6 to obtain precoding vectors which are then used during slot 2 (See Eq. 3.2). This computation may take time (due to communication among APs over the backbone). In general wireless networks, this creates inaccuracies since the channel between APs and the mobile clients may change from the time the channel state information (CSI) was measured to the time when the APs retransmit the data in the second slot. Thus, the precoding vectors that were computed based on old CSI may not be suitable for the

channel's current state. This may lead to inaccurate beamforming and nulling. However, in RobinHood, the mobile clients do not participate in the second slot. Only the APs transmit and receive data in the second slot. Due to the immobile nature of the APs, the channel (or CSI) between APs changes very slowly (See Fig. 3.2a). Thus, the CSI computed among APs is valid for longer duration compared to CSI between mobile clients and APs. *By requiring only the APs to transmit in the second slot, RobinHood ensures higher accuracy of joint beamforming and joint nulling.*

Number of APs required: In general, if there are N clients in the network, then RobinHood needs to align $N-1$ packets at the first AP, $N-2$ packets at the second AP and so on. Thus, a total of at least $(N-1)+(N-2)+\cdots+2 = \frac{N^2-N-2}{2}$ variables are required to satisfy all the constraints. However, to obtain a non-zero solution, we need to include one extra AP, *i.e.* a total of $\frac{N^2-N}{2}$ APs. However, $N-2$ of the variables are supplied by the scaling coefficients at the receiving APs. Thus, a total of $\frac{N^2-N}{2} - (N-2) = \frac{N^2-3N+4}{2}$ transmitting APs are required. Finally, $N-1$ receiving APs are also required in slot 2, where the first $N-2$ receiving APs decode one unique packet while the last AP decodes 2 packets. Therefore, with $\frac{N^2-3N+4}{2} + N - 1 = \frac{N^2-N+2}{2}$ APs, RobinHood can leverage this high density of APs to decode N uplink packets in exactly two slots. Further, in contrast to [13], RobinHood requires N fewer APs.

3.3 Challenges

Note that when the APs (*i.e.*, AP_3 and AP_4) in slot 2 transmit, they have to align the samples of x_2 and x_3 at AP_1. To achieve this, they precode the signals that they received in the first slot and transmit. However, in contrast to the existing solutions [65], in RobinHood, the transmitting APs are not aware of what they are transmitting (since they are unable to decode the samples received in the first slot). We call this *Blind Beamforming and Nulling*.

Although the idea behind RobinHood is simple, there are multiple challenges that need to be handled to make it practical.

- **Oblivious to the contents of the transmitted signal:** The APs transmitting in slot 2 are not aware of the contents of the signals transmitted in slot 2. Despite this, they need to cancel out (or align) the different contents of the signal at different receiving APs.

- **Synchronization**: In order for the APs transmitting in slot 2 to align their signals at the receiving APs, these transmitting APs are required to be synchronized at the sample level. This requirement is similar to the requirements of the other existing algorithms that focus on downlink traffic [47, 64, 65]. Observe that RobinHood does not impose synchronization requirement on the mobile clients.

- **Multi-collision domain:** The previous discussion assumes that all clients and all APs can hear each other directly. However, this may not be true for large scale EWLANs. Thus, we need a mechanism to extend RobinHood to such networks.

- **Inconsistency in the AP density:** To decode N packets, RobinHood requires $\frac{N^2-N+2}{2}$ access points nearby. However, the actual number of APs present may be higher or lower than this number. If the number of available APs is higher, then RobinHood can make use of all of them. On the other hand, if the number of available APs is smaller, than a mechanism is required to select a subset of the clients.

- **Robustness:** Unlike downlink [65], where each client individually decodes its own packet, in RobinHood, decoding happens in a cascading fashion. Decoding of a packet depends on the successful decoding of the previous packets. Clearly, in such a design, failure in decoding of one packet, makes all future decodings unsuccessful. We need a new mechanism to increase the robustness of the decoding.

The next two sections explain how we handle these challenges.

3.4 Physical Layer Design

In this section, we explain the physical layer working of RobinHood using three different phases. First, we explain how multiple clients transmit simultaneously to the APs and how the channel state information between clients and APs is estimated. Then, we show how the APs conduct blind-beamforming and nulling without knowing the contents of the transmitted signals. Finally, the decoding process is explained. In RobinHood, the clients participate in only the first phase while the APs participate in all the three phases.

3.4.1 Phase I: Client transmission

As explained in Section 3.2, the transmissions in RobinHood are divided into two slots. In the first slot, the clients transmit concurrently to the APs. Besides the received combined samples from the clients to APs, the channel state information (CSI) between all the clients and APs is also computed in this phase. To obtain the CSIs, each client sends an access code (or unique PN sequences [55] assigned to each client) that is free of interference.

The transmission timeline of Phase I is shown in Fig. 3.4. First, the APs broadcast an *approve* message. This message contains the IDs of the clients that are allowed to transmit in this slot (For more details on how the APs select the subset of clients, refer to Section 3.5 that describes the MAC design of RobinHood). The relative order of the IDs determines the time when a client should transmit its access code. Since the clients are not synchronized, the transmission of access codes may partially overlap with each other. To avoid this overlap, a small time gap, called inter-access-code-space (IACS), is inserted between the transmissions. Finally, after the transmission of access codes, the clients transmit their packets simultaneously. All the APs compute the CSI from different clients using the

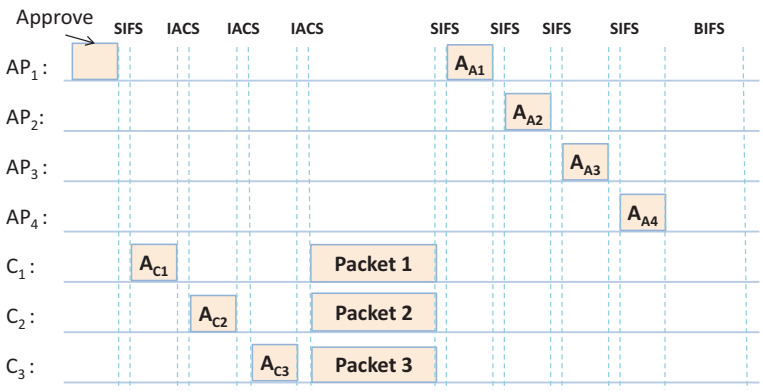


Figure 3.4: Phase I time-line: A_{Ci} and A_{Aj} represent the access codes for C_i and AP_j, respectively.

interference-free access codes and also store the received samples corresponding to the data packets. In our experiments and simulations, we set the duration of IACS to $2\mu s$.

To conduct blind-beamforming, besides the CSIs between the clients and APs, the CSIs between the transmitting APs and receiving APs are also required. As shown in Fig. 3.4, all of the APs broadcast their access codes one after the other. When one AP broadcasts, all other APs can estimate the CSI from that AP. The estimated CSIs along with the CSIs between clients and APs are forwarded to a *group-head AP* through the wired backbone network. The head AP, uses these CSIs to compute the best sets of transmitting APs, the set of receiving APs, the decoding order, and the precoding vector to be used by each of the transmitting AP. This information is then sent back by the group-head AP to every AP in the group. In Fig. 3.5, AP_3 and AP_4 are selected as the transmitting APs.

This computation at the group-head AP and the distribution of result back to APs may take some time due to delays on the wired backbone. To ensure that all APs have received the computed results back from the group-head AP, RobinHood requires all APs to wait for Backbone-Inter-Frame-Space (BIFS) duration.

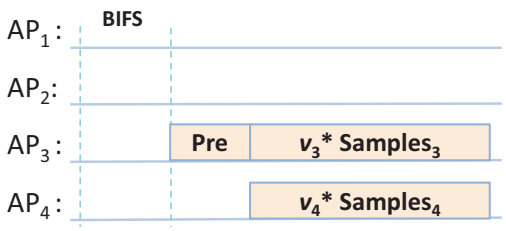


Figure 3.5: Phase II time-line: v_i denotes the precoding vector of AP_i.

3.4.2 Phase II: Blind-beamforming

After waiting for BIFS time, all APs multiply the samples received in the first slot with their precoding vector and retransmit them (See Fig. 3.5). The value of BIFS can be selected on the basis of the speed of the Ethernet and the expected delays involved. To avoid wastage of wireless channel during BIFS, APs in RobinHood participate in another set of communication (*e.g.*, downlink traffic) while waiting to hear back from the group head AP. Observe that since the APs are relatively stationary, the precoding vectors computed by group-head AP are valid for a long duration as described in Section 3.2. Further, due to the relatively stationary channel among APs, we do not need to frequently measure the channel among APs which further reduces the overhead incurred during Phase I. The short packet **Pre** sent by AP_3 is a sequence known to all of the APs. The purpose of sending this sequence is two fold: 1.) it can be viewed as a preamble for the receiving APs to detect the correct start point of the retransmission; 2.) it can be used to estimate the sampling offset between the transmitting APs and receiving APs.

3.4.3 Phase III: Decoding Packets

In RobinHood, the packets are decoded in a sequential order. The first AP decodes one packet and sends it to the next AP which, upon receiving the packet, recreates the received

samples, and then subtracts those samples from the received samples. The remaining samples are decoded to obtain the second packet. This process is continued until all packets have been decoded. Performing a successful subtraction requires estimating various offsets such as frequency offset, sampling offset, and phase offset. Once the offsets have been estimated, the AP needs to recreate the received samples. This sequential decoding and packet subtraction have been well-studied in the literature [11, 29]. We refer the reader to the existing literature.

Another practical issue to note is that there is sampling offset between the transmitting APs and the receiving APs in the second slot. This offset makes it difficult to align the components of x_2 and x_3 received by AP_1 in the second slot with the corresponding components received in the first slot. To that end, a packet **Pre** that is known to every receiving AP is transmitted by AP_3 in Phase II. This packet is used to estimate the sampling offset between the transmitting and the receiving APs using the same techniques as in packet subtraction [11, 29].

3.4.4 Computing the Packet Decoding Order

In the previous discussion (Sec. 3.2), we assumed that x_1 is decoded first, followed by x_2 and x_3. We also assumed that joint precoding leaves no residual noise. However, in practice, joint precoding and packet subtraction are not perfect and leave some residual noise. Thus, in this section, we compute the optimal order in which packets should be decoded such that the decoding accuracy is maximized in the presence of the residual noise. To determine the optimal decoding order, we need to compute the expected received signal strength (RSS) of each packet (say x_i) at each AP (say AP_j). The exact value of RSS depends on the precoding vectors which in turn depend on the rest of the matching. This makes the problem combinatorial in nature.

We compute the expected RSS of client i at AP_j using a heuristic. In the second

slot, let AP_N to AP_M be the set of transmitting APs and AP_1 to AP_{N-1} be the set of receiving APs. Observe that in the second slot, AP_j receives components of x_i that have been retransmitted by all APs in the range AP_N to AP_M. Thus, components of x_i arrive at AP_j through $M - N + 1$ different paths. Each of these $M - N + 1$ paths start at client i, pass through some transmitting AP (say AP_k) and end at AP_j. Further, each of these paths consist of two links: First from C_i to AP_k and, second from AP_k to AP_j. We say that RSS_{ij} is expected to be high only if there is at least one path on which x_i has high signal strength on both the links. If P_0 is the transmission power level, then, we can estimate the RSS of x_i at AP_j as follows:

$$RSS_{ij} \approx P_0 \times \max_{k=N...M} \left(\min \left(||h_{ik}^{(1)}||^2, ||h_{kj}^{(2)}||^2 \right) \right) \tag{3.7}$$

Consider client C_i that transmits packet x_i at data-rate R_i. Let AP_j be the receiving AP that decodes x_i. If τ_i is the minimum SNR required to decode x_i where τ_i depends on the physical layer data rate, then the residual noise that can be tolerated at AP_j during the decoding is given by [11]: $\frac{RSS_{ij}}{\tau_i}$. Using this, RobinHood computes the maximum residual noise that each packet can tolerate. Let us say AP_j decodes the i^{th} packet in the decoding sequence.

Observe that in RobinHood the packets are decoded sequentially. So, if a packet is not decoded correctly, then all other packets that depend on it can't be decoded either. So, in order to improve the decoding probability of all the packets, the decoding order is chosen by arranging the packets in non-increasing order of the maximum residual noise that they can tolerate.

3.5 MAC Design

In this section, we first explain how RobinHood works in large scale networks. Next, we explain how RobinHood leverages the variation in the density of access points to improve

the throughput of the uplink traffic. Finally, we explain how RobinHood coexists with ongoing downlink traffic in the network.

3.5.1 Multi-Collision Domain

The previous sections describe how RobinHood works in a single collision domain. To work in a practical multi-collision domain, RobinHood needs to solve multiple challenges:

- In a multi-collision domain network, an AP may not be able to hear all other APs. This makes it difficult to synchronize them since existing algorithm with high synchronization accuracy [64] works only within a single collision domain.
- The traffic distribution may be different across different parts of the network. For example, some parts of network may experience higher downlink traffic compared to others.
- The MAC algorithm should ensure fairness across different clients.
- Previous discussion of RobinHood requires that all cooperating APs and all clients are able to hear each other. Satisfying this requirement is challenging since frequent mobility of clients requires frequent re-computations.

RobinHood as described in Section 3.2 requires that: (i) **All cooperating APs should be able to hear each other**; and, (ii) **All APs should be able to hear all clients**. So, one naive way of extending RobinHood to multi-collision networks would be to arrange both the APs and clients in groups such that within each group all APs and all clients can hear each other. However, this naive approach would require frequent re-computation of groups due to client mobility.

To ensure that RobinHood works with mobile networks without requiring frequent re-computations, we divide the EWLAN into cliques of APs while only satisfying the first requirement. Satisfying that requirement implies decomposing the graph into as few cliques

of APs as possible. Since, decomposing graphs into fewest cliques is an NP-Hard problem, RobinHood uses a greedy polynomial-time algorithm to compute such cliques. Our polynomial-time algorithm repeatedly finds a maximal clique among all APs. Then, it removes the vertices (and the edges incident on them) that are part of the maximal clique. The algorithm then runs on the remaining graph to find the maximal clique. This process is repeated until every AP is a part of some clique. All the APs that are in the same maximal clique, form a single *group*. This decomposition algorithm can be run by a central server similar to [47, 71]. *Ensuring that all APs in the same group can hear each other allows RobinHood to leverage the existing synchronization algorithms (such as SourceSync [64]) to synchronize all the APs that are part of the same group.*

Observe that since the APs are immobile, once the membership of different groups has been computed, it can be used for long periods of time. It is possible that an AP may not be able to hear a client that belongs to the same group. Thus, grouping based on APs only satisfies the first requirement specified above while the second requirement may be violated. We handle this in Section 3.5.2.

Computing neighbor relationship among groups: To prevent interference from neighboring groups and to keep groups independent, RobinHood ensures that at any time if the APs belonging to group G are communicating, then the APs belonging to neighboring groups should not communicate. Two groups (say G_i and G_j) are said to be neighbors of each other if (i) There exists a wireless device (an AP or a client) in G_i that is in the interference range of a wireless device in G_j; or, (ii) There exists a wireless device (an AP or a client) in G_j that is in the interference range of a wireless device in G_i. To decouple the dependence of neighbor-relation from the location of mobile clients, RobinHood takes a conservative approach such that G_i and G_j are called neighbors even if there could potentially exist a client that can be in the transmission range of some AP in G_i while being in the interference range of some AP in G_j. *By decoupling the neighbor relationship*

from the location of mobile clients, RobinHood significantly reduces the overhead that may otherwise arise due to frequent re-computations.

Scheduling different groups: To ensure that two neighboring groups are not transmitting simultaneously, RobinHood uses a central server [47, 65, 71] that manages the interference among neighboring groups. Since the schedule length in RobinHood is always two slots across all the groups, it makes it convenient for the server to schedule the active groups. In RobinHood, at any time t, the server computes the set of groups that will communicate for the next two slots (t and $t + 1$). This set is computed using maximum independent set techniques such that there is no interference among the neighboring groups. However, due to unexpected delays on the wired backbone, the latency from the central server to the APs may result in APs unnecessarily waiting for the control messages from the server while the wireless channel is idle. To avoid this waiting, the server in RobinHood proactively computes the schedule and transmits it to the APs over the backbone.

Client-AP association: In RobinHood, clients do not permanently associate with any specific AP or a group. The clients simply wait for the *poll* packet from any neighboring AP and transmit uplink data as soon as they receive the corresponding *approve* packet as shown in Fig. 3.6. *By keeping the clients stateless, RobinHood reduces the control messages exchanged between APs and clients.*

ACK transmission: In RobinHood, the APs decode the packets during Phase 3. After decoding, the APs send ACK over the wireless to the clients as shown in Fig. 3.6.

Downlink traffic: Uplink transmissions in RobinHood can coexist with downlink traffic. Each group in RobinHood can either do downlink transmissions or uplink transmissions, independently of the other groups. For downlink communication, existing algorithms [47,47,65,71] can be used. The central server used in RobinHood can also be used for managing downlink interference as in the existing algorithms [47, 71].

3.5.2 Computing the set of transmitting clients

In general, in a system with N clients and $\frac{N^2-N+2}{2}$ APs, RobinHood guarantees that each client can transmit 1 packet every two slots. Within a single group, it is possible that the number of APs may not be high enough to support all the clients. In that case, the group-head AP selects a subset of clients that would transmit in the first time slot. To ensure fairness among clients, RobinHood uses a weighted credit based system [47] such that the credit of a client is high if it has not been scheduled for a long period of time. Thus, the clients with the highest credit are given priority to transmit. This is further described later in this subsection.

Fig. 3.6 explains the complete working of RobinHood. Initially, the APs in a group (if allowed by the central server) poll the network for uplink traffic. This is followed by a contention period in which different clients transmit short packets conveying their credit balance to contend for the uplink transmission. Next, the "group-head AP" computed the set of clients that are allowed to transmit their data packets. This information is conveyed in the *Approve* message. Finally, the approved clients transmit their data packets which are decoded by the APs in three phases as described in Section 3.4.

On the other hand, it is also possible that the number of clients are low while there are more APs available (*e.g.*, in highly dense networks such as in Fig. 3.3). In that case, RobinHood can leverage the extra APs to further improve the robustness of decoding as discussed in Section 3.5.3.

***Approve* algorithm** In each group, a single AP is elected as the *group-head AP* that executes the *Approve* algorithm to compute the set of clients that are allowed to transmit. *Approve* (Algorithm 2) greedily computes the schedule. In each iteration, it adds the client with the highest credit value to the schedule (Line 8), thereby improving fairness. For such a client, it picks the best AP (say AP_j) that has not yet been paired with some other client (Lines 11-15). Next, Approve tries to add this client-AP pair to the schedule S and checks

if S is still satisfiable (Lines 16-18). This check is done by Algorithm $Satisfiable$. If this pair makes S unsatisfiable (Lines 19-21), then the pair is removed from S. Also, C_i is marked as ineligible since it cannot be paired with any AP. This process is repeated until no more client-AP pairs can be added to S (Lines 9-10).

Algorithm $Satisfiable$ determines if a given schedule is satisfiable or not. When doing this computation, $Satisfiable$ takes into account the set of clients that each AP can hear. Without loss of generality, let S be the schedule such that $S = \{(C_i, AP_i) : AP_i$ is the receiving AP for packet x_i and x_i is the i^{th} packet to be decoded$\}$. $Satisfiable$ should return true if for every client-AP pair, say (C_i, AP_i), it can find a subset of $i-1$ unique APs in the same group that can align x_i at the receiving APs (AP_1 to AP_{i-1}). In other words, for every client-AP pair, say (C_i, AP_i), $Satisfiable$ needs to find $i-1$ other APs that are in the transmission range of C_i. This computation can be done by reducing this problem to a Max Flow problem as shown in A.3.

3.5.3 Robustness

In RobinHood, the first AP decodes one packet while $N-1$ packets are nulled using blind beamforming. The second AP decodes the second packet (while the other $N-2$ packets are nulled using blind beamforming) and so on. Thus, if an AP cannot decode a packet due to inaccuracies in blind beamforming or packet subtraction, then all the following packets that depend on it can also not be decoded. Therefore, to ensure that the first few packets in the decoding order can be decoded with high probability, RobinHood leverages the high density of the APs. Specifically, RobinHood increases the decoding robustness of the packets if the number of APs present in a group are more than the minimum required (See Sec. 3.2).

Let $C_1, C_2, C_i, \cdots, C_N$ be the order in which the clients are decoded (See Section 3.4.4). Let the number of APs in the group be M and E be the number of extra APs

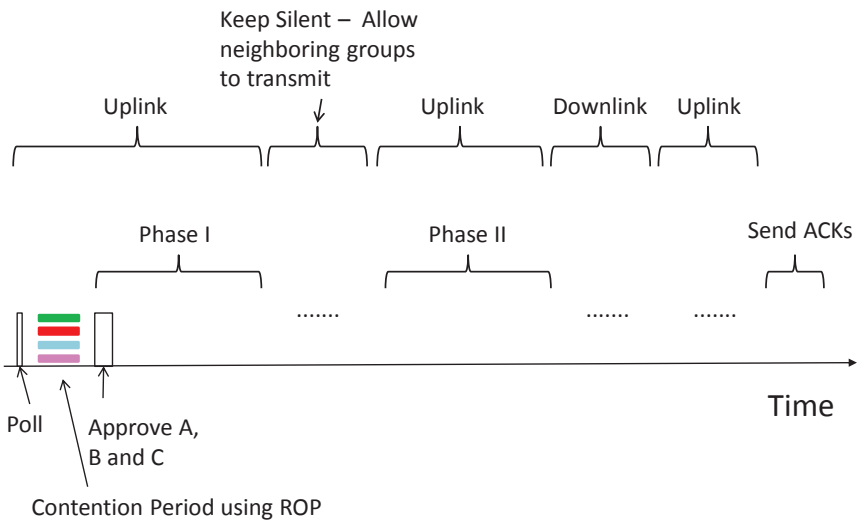


Figure 3.6: Timeline of data transmission in a large network. The data sent by clients during contention phase are transmitted using the Rapid OFDM Polling (ROP) [81] scheme to decrease overhead. Phase III is executed in the background over the wired backbone allowing wireless channel to be used for other purposes.

that are present such that $E = M - \frac{N^2-N+1}{2}$. Recall that exactly $N - i$ packets are nulled (or aligned) at the AP that decodes the packet from client C_i. If we require one of the extra APs to independently decode the packet from C_i, then we will need another $N - i$ extra transmitting APs to ensure that packets from $C_{i+1}, C_{i+2}, \cdots, C_N$ are nulled at this extra AP. Thus, to decode the packet from C_i at two different APs, we need another $N - i + 1$ APs (including one extra AP for receiving).

RobinHood increases the decoding robustness as follows: The APs in RobinHood find the *first* client C_i in the decoding sequence that satisfies the two requirements: (i) The packet from C_i is decoded at only one AP; and, (ii) $E \geq N - i + 1$. Let C_i be the first client in the decoding sequence that satisfies the two constraints. Then, RobinHood ensures that the packet transmitted by C_i can be independently decoded by two different

APs. RobinHood decreases E by $N - i + 1$ since this is the number of APs required to achieve independent decoding of C_i. Finally, this process is repeated as long as possible to achieve independent decodings of some packets. Thus, in highly dense networks, RobinHood leverages the extra APs present in the network to further increase the decoding probability of each packet. Even if extra APs are not available, RobinHood can restrict the number of clients that transmit simultaneously. This frees up some APs that can be used for increasing the robustness of decoding. Currently, we leave the problem of proactively reducing the number of transmitters to increase the decoding robustness as our future work.

3.6 Experiments

3.6.1 Setup

We evaluate RobinHood in a testbed with 7 USRP N210 nodes. The setup is as follows:

- **Hardware and software setup:** Each USRP is equipped with a WBX daughterboard and operates in the 400 MHz band. All nodes are within single collision domain. At the receiver side, we use the GNURadio for signal processing. The decoding is done offline in Matlab. All of the AP nodes are synchronized with an external clock source generated by OctoClock-G [2]. In practice, SourceSync [64] can be used to synchronize the transmitting APs to a nanosecond level accuracy.

- **OFDM and modulation setup:** We use a 512 FFT system, with 200 subcarriers used for data transmitting. The cyclic prefix length is set to 128. Unless otherwise mentioned, Binary Phase Shift Keying (BPSK) is used as the modulation scheme.

Apart from implementing RobinHood, we also implemented **Omniscient TDMA** that utilizes a central server. This server is aware of (i) packet queue at different clients; and,

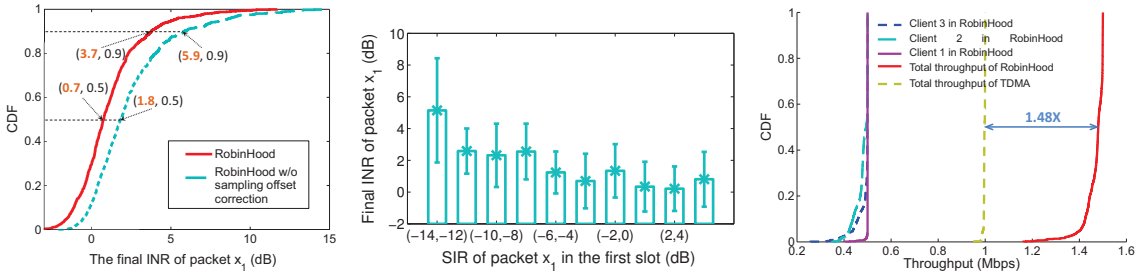
(ii) the channel between all clients and all APs. Omni-TDMA schedules the three different clients in a round-robin fashion with each client transmitting to the AP to which it has the best channel.

3.6.2 Micro-Benchmarks

Many works have shown the effectiveness of beamforming [30, 65]. Since our blind-beamforming and nulling involves transmitting unknown samples, its effectiveness and accuracy is unclear. In this section, we evaluate the performance of blind-beamforming and nulling using the signal to interference and noise ratio (SINR). In the following experiments, 3 clients and 4 APs were deployed in our testbed as shown in Fig. 3.1.

Blind-beamforming and Nulling Effects: First of all, we study the blind-beamforming and nulling effect as described in Section 3.4. Since there are a total of 12 links between all APs and clients, it is difficult to control the SNR of every link. Instead, we place the clients and APs randomly in our testbed and record the actual SNRs. We repeat the experiment 20 times for each of the 50 randomly chosen topologies. Over various topologies, the SNR between clients and APs varied from 6 dB to 35 dB. We compute the final interference to noise ratio (INR) of packet x_1 when it is decoded by AP_1. The INR distribution is shown in Fig. 3.7a. The median of the INR is 0.7 dB, which is just slightly above the noise floor, and the 90^{th} percentile INR is 3.7 dB. This indicates that residual interference from blind-beamforming and nulling is relatively small and demonstrates the practicality of RobinHood.

The INR distribution in Fig. 3.7a shows that it could be as high as 10 dB, which is a large value compared with typical SNR values, *e.g.* 20 dB. We look deeper into the INR results and present it in an another way in Fig. 3.7b. The y-axis is the final INR of packet x_1 at AP_1. The x-axis is the range of signal to interference ratio (SIR) in dB that x_1 experiences in the first slot across all of the APs. The smaller the value on the x-axis is, the



(a) Final INR distribution of x_1 at the first decoding AP. The median of the INR is 0.7 dB and the 90^{th} percentile INR is 3.7 dB.

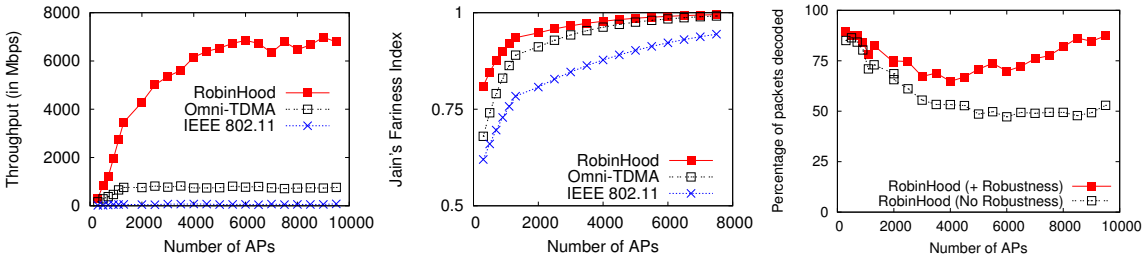
(b) Final INR of the packet x_1 at the first decoding AP.

(c) Throughput of RobinHood compared to Omniscient TDMA.

Figure 3.7: Experiment results collected over USRP testbed.

higher is the amount of interference to be canceled in the second slot. This figure shows that as the SIR increases, the final INR decreases. When the first slot SIR is larger than -12 dB, the median of the that is 0.6 dB and the 90^{th} percentile is 2.7 dB (Fig. 3.7b). Based on this result, we can enable RobinHood when the SIR value is larger than a threshold and fall back to the default IEEE 802.11 scheme when the SIR value is small. We leave the study of computing the exact threshold value as future work.

Sampling Offset: As discussed in Section 3.4.3, there is sampling offset between the samples received by AP_1 from phase I and phase II. To study the effect of the sampling offset, we turn off the sampling offset correction in RobinHood and compute the residual interference to noise ratio for x_1. The result shown in Fig. 3.7a shows that without sampling offset correction, the median INR increases by 1.1 dB and the 90^{th} percentile increases by 2.2 dB. This demonstrates that the sampling offset correction done in RobinHood reduces the residual interference.



(a) Total Throughput (b) Jain's Fairness Index (c) Percentage of packets decoded

Figure 3.8: Trace-Driven Simulation Results for Multi-Collision Domain

3.6.3 Throughput

In this section, we study the throughput performance of RobinHood. The throughput of each client in RobinHood is recorded and compared with that of omniscient TDMA. A total of 20000 packets are transmitted by each client across different topologies. Fig. 3.7c shows that on an average, RobinHood provides a throughput gain of $1.48\times$ compared with omniscient TDMA. The figure also shows the throughput of client 1 is higher than that of clients 2 and 3. This is because x_2 and x_3 are decoded only if x_1 is decoded. Further, even if x_1 is decoded, x_2 and x_3 may not be decoded due to the residual interference from subtraction.

3.7 Trace-Driven Simulation

This section explains the setup and the results from the trace-driven simulations.

3.7.1 Simulation Setup

Apart from implementing RobinHood, we also implemented two other algorithms: (i) **Omniscient TDMA algorithm:** Described before in Section 3.6. However, this time similar

to RobinHood, Omniscient TDMA also uses a credit-based system where a client has high credit value if it has not transmitted for a long time. In each slot, it schedules an maximum independent set of "client to AP" links (where weight of link = credit of the client $\times$ throughput of the client when using that link). The physical layer rate of a link is chosen by picking the highest data rate that can be decoded by the AP.; and, (ii) **IEEE 802.11 (without RTS/CTS).** To evaluate the gain provided by RobinHood irrespective of the downlink algorithm used, only the uplink traffic from clients to APs was generated. Various traces were incorporated into the simulation: (i) **Noise due to Blind Beamforming and Nulling:** The simulator incorporated noise arising due to imperfect nulling. For this, we used the traces collected from our experiments (See Fig. 3.7a). (ii) **Noise due to subtraction:** When an AP subtracts a packet, it has to recreate its samples and correct for various offsets such as sampling offset and frequency offset. An imperfect correction leads to imperfect subtraction resulting in residual noise. The simulator incorporated this residual noise using the traces collected by us in experiments. (iii) **Path Loss between clients and APs:** Incorporated from the traces [74]. (iv) **Path Loss between APs:** Incorporated from the traces [74].

In this section, we study the behavior of RobinHood in a large EWLAN (Enterprise Wireless LAN) that spans over multiple collision domains (*e.g.*, the campus of a university). Our simulator first randomly deploys 500 clients in a field of size 500m $\times$ 500m. APs are also deployed randomly and the count of APs is varied. In this setup, different devices may belong to different groups as described in Section 3.5. The overhead of different protocols was taken into account during the simulation. Also, APs in RobinHood used extra APs to further increase the decoding robustness as described in Section 3.5.3. Finally, clients in RobinHood and IEEE 802.11 used the Auto Rate Fallback (ARF) algorithm to determine the physical layer data rate.

3.7.2 Results

Next, we describe the results from our trace-driven simulations.

- **Total Throughput across all clients:** Throughput increases for all algorithms as they leverage the increase in the physical layer data rate (See Fig. 3.8a). For 802.11, increase is not substantial since high number of collisions (due to hidden terminals) reduces the number of successful transmissions. With increase in number of APs, the throughput in RobinHood increases because of two reasons: (i) Higher AP density implies more APs are present in each group, resulting in higher throughput since more clients can be supported at the same time; and, (ii) Higher data rate at clients due to higher AP density. As the density of the APs increase, throughput in RobinHood increases substantially compared to TDMA. When the number of APs is 2000, each client is in the range of an average of 180 APs. At that density, RobinHood throughput is $5.2\times$ compared to TDMA, and $52.4\times$ compared to IEEE 802.11. This is lower than the expected gain since in RobinHood, clients use ARF to adjust their physical layer data rate while clients in Omniscient TDMA transmit at the best possible data rate.

  Once the number of APs grow beyond 4000, the total throughput of the network does not increase much. This is because most of the groups already have enough APs to support all the clients. However, the decoding robustness provided by redundant APs still helps in slightly increasing the throughput.

- **Fairness:** Fig. 3.8b shows the variation in Jain's fairness index with variation in number of APs. The fairness index of RobinHood is higher than other algorithms since RobinHood allows all clients to transmit. RobinHood has higher fairness than TDMA since RobinHood performs precoding over transmissions from all clients. Thus, even the clients that are far away from all APs may experience high throughput

due to beamforming from multiple helper APs. With increase in the number of APs, the fairness for all the algorithms increases since the chances that a client would be close to some AP increases.

- **Decoding probability:** As the density of the network becomes higher, the length of the decoding chain increases. Thus, a decoding failure on one packet implies a decoding failure on all the other packets that depend on it. Fig. 3.8c shows the percentage of packets decoded successfully decreases with increase in density. However, still the throughput in RobinHood increases (See Fig. 3.8a) since higher density enables multiple clients in APs to transmit successfully. Further, with high density of APs, RobinHood can use robustness techniques discussed in Section 3.5.3 to increase the decoding probability. Fig. 3.8c also shows the decoding probability when RobinHood does not use robustness techniques described before. With the increase in number of APs, there is a higher chance that RobinHood can leverage those APs to improve robustness. Thus, with increasing density, the improvement provided by robustness further increase.

  When the number of APs increases from 300 to approx. 4000, the decoding probability keeps decreasing since APs try to simultaneously decode multiple packets and a decoding failure on one packet results in decoding failure on all the dependent packets. However, as the number of APs grow beyond 4000, the redundant APs help in increasing the decoding robustness as shown in Fig. 3.8a.

3.8 Discussion

In this section, we discuss some further modifications that make RobinHood more practical.

Reducing overhead of channel estimation: To compute the precoding vectors, the APs in RobinHood require the knowledge of channel between all clients and APs as well

as the channel between all APs. The problem of computing the channel from clients to APs has been well studied in the context of MIMO networks [30, 80]. To compute the channel values, we are planning to use PN sequences to estimate the channel from multiple transmitters simultaneously [52].

Overhead on the backbone: In RobinHood when decoding N uplink packets, the APs need to exchange $\frac{(N-1)\times(N-2)}{2}$ data packets. This is in contrast with [65] that requires exchange of $(N-1)\times(N)$ data packets when performing joint beamforming for the downlink traffic. In addition, APs in RobinHood also need to relatively smaller exchange control packets related to channel state information, scheduling etc. Currently, we are exploring techniques to adjust the number of participating clients based on how much overhead can be tolerated on the wired backbone.

APs with multiple antennas: If the APs are equipped with multiple antennas, RobinHood can leverage them to reduce the number of required APs. Specifically, if each AP is equipped with K antennas, then to receive N uplink packets simultaneously, RobinHood would require only $\frac{N'^2-N'+2}{2K}$ APs where $N' = N - (K-1)$, a reduction by a factor of more than K.

3.9 Related Work

Although RobinHood builds on several prior work, it differs from them in various ways.

Backbone usage: The idea of using the wired backbone to increase wireless throughput is not new. In MegaMIMO [65], multiple APs cooperatively precode the transmissions such that each client receives only the packets intended for it while the other transmissions are canceled out. However, MegaMIMO requires that transmitters exchange packets among themselves and thus, it works only for the downlink transmissions. On the other hand, RobinHood improves the throughput for the uplink traffic. Also, in contrast to MegaMIMO

and OpenRF [47], transmitters in RobinHood jointly perform nulling without knowing the actual contents of the packets.

A recently proposed protocol Symphony [11] also focuses on uplink traffic. However, in contrast to RobinHood, Symphony improves the network throughput only when the APs are in different collision domains. In Epicenter [32], authors propose that APs should exchange coarse representations of symbols to decode corrupted bits. Similarly, authors in [79] also propose that APs exchange bits or raw samples on the backbone to facilitate packet decoding. In all these algorithms, the APs cooperate to decode the same packet whereas in RobinHood, APs encourage transmitters to collide and then cooperate to decode multiple packets simultaneously without exchanging the raw samples.

Finally, [13] also proposed using the backbone to improve the uplink throughput. However, in contrast to [13], RobinHood provides the first implementation of blind beamforming and nulling. Further, RobinHood works in multi-collision domains, uses robustness techniques to increase decoding probability, and, requires N fewer APs compared to [13].

Interference Alignment: Previously, researchers (see [37] and references therein) have used interference alignment to improve the capacity of wireless networks. However, unlike RobinHood, they either require APs to exchange samples over the backbone [8], work only for the downlink traffic [76], assume presence of significant number of clients [58], require multiple antennas at transmitters or receivers [30], require the antennas to be physically moved [4] to a certain point, require the channel to change from one slot to another [16], precode over exponential number of time slots [16], or provide limited throughput gain [4, 30]. These assumptions are not practical in mobile networks since if the client is stationary, the channel may not change [78] from one packet to another. In contrast to the previous works, RobinHood works even if the channel stays stationary.

Wireless Relays: Researchers [46, 66] have also looked at the problem of using special relay nodes to assist in high speed communication between specific pairs of source and

destination nodes. In contrast, the focus of RobinHood is to leverage the high density of APs and the wired backbone to carefully select the set of destination APs, determine which AP decodes which packet, and to use the wired backbone to migrate all the complexity away from the clients. Further, with previous works, it is possible that the destination AP is unable to decode a packet due to low SNR. However, in RobinHood, APs leverage the high density of APs to increase robustness (See Sec. 3.5.3).

Information Theory Results: Researchers have studied the capacity of the wireless networks from information theory perspective. We divide the previous work in this area in three categories:

- **Interference-model based without APs:** This category of works treat interference as a black box. Authors in [34] showed that in a single collision wireless network with N transmitters and N receivers, the maximum total throughput is $O(\sqrt{N})$. Authors in [26] proposed a percolation-theory approach to construct an algorithm that indeed achieves $O(\sqrt{N})$ throughput. However, these papers did not assume the presence of a wired backbone that connects the receivers and thus the results in this thesis are not in conflict with these previous papers.

- **Interference-model based with infrastructure support:** Researchers have also explored the capacity of wireless networks that are supplemented by an infrastructure of APs that are connected through a wired backbone. Liu et al. [53] showed that for such networks, only if the number of APs are more that $O(\sqrt{N})$, only then the infrastructure network is useful in improving the wireless capacity. In [5], authors show that it is possible to achieve $O(N)$ throughput for a hybrid wireless network with N transmitters, N receivers and $O(N)$ APs that are connected by wired infrastructure. However, like the previous category, their analysis is restricted to interference based wireless networks, *i.e.*, signals received from nodes other than the particular transmitter are regarded as interference degrading the communication link.

However, in RobinHood, we explore more sophisticated physical layer processing to align interference resulting in higher throughput.

- **Physical layer processing techniques without infrastructure support:** On the other hand, in [59], authors show that in the absence of APs, by using physical layer processing techniques such as MU-MIMO, it is possible to achieve $O(N)$ throughput.

In all these previous works [5,26,53,59], it is assumed that all wireless links in the network follow the same path loss model. In other words, they assume that for any transmitter, say t_1, if receiver r_2 is farther away from t_1 compared to another receiver r_3, then r_2 will have lower RSS from t_1 compared to r_3. This assumption is used to create clusters of devices where: (i) All nodes in the same cluster have low path loss between each other; and, (ii) Nodes belonging to different clusters have high path loss between each other. However, in practical wireless networks (*e.g.*, with mobile devices in indoor environments), it may not be possible to create such clusters because nodes closer to each other may have high path loss while nodes far away may actually have lower path loss (resulting from shadowing, multipath etc.). In contrast, our algorithm does not make any such assumptions about the path loss model.

From the information theory perspective, with a total of $O(M)$ devices, RobinHood achieves $O(\sqrt{M})$ throughput. This is significantly lower than the currently known [59] upper bound of $O(M \log M)$ in single collision domain wireless networks. For practical indoor wireless networks, where it is not possible to create such clusters, the currently known best algorithm is TDMA that provides an uplink throughput of $O(1)$. Thus, RobinHood provides a throughput improvement of $O(\sqrt{M})$ compared to existing algorithms. RobinHood has taken the first step in the direction of improving throughput for such networks where different links follow different path loss models. It remains an open question if it is possible to further improve throughput for such wireless networks.

3.10 Conclusions

In the previous sections, we discussed RobinHood, a blind beamforming and nulling scheme that leverages the high density of access points to enable multiple mobile devices to transmit simultaneously. Feasibility of RobinHood was verified on a USRP testbed. Measurements show that RobinHood achieves a throughput gain of $1.48\times$ over omniscient TDMA. Using trace-driven simulations, we showed that in dense wireless LANs, RobinHood provides a throughput of up to $5.2\times$ compared to omniscient TDMA.

Algorithm 2: $Approve$: Computes the set of clients that will be approved in this slot

Input: For every eligible packet P_i, its transmitter C_i. Also, information on which AP can hear which client.

Output: (i) Set of clients that will be approved in this slot. (ii) The matching from the approved clients to the APs indicating which AP decodes which packet. (iii) The decoding order.

$E_i \leftarrow true \; \forall i : 1 \leq i \leq N$

$\mathcal{A} \leftarrow$ All APs in the current group

$S \leftarrow \{\}$

while $true$ **do**

  $CSet \leftarrow \{C_x : E_x = true$ and $C_x \notin S\}$

  $C_i \leftarrow C_i \in CSet$ and C_i has the highest credit balance

  if $C_i = null$ **then**

    return S

  $Set \leftarrow \{(C_i, AP_j) : AP_j \notin S\}$

  $(C_i, AP_j) \leftarrow (C_i, AP_j) \in Set$ and RSS_{ij} is maximum

  if $AP_j = null$ **then**

    $E_i \leftarrow false$

    continue

  $S \leftarrow S \cup \{(C_i, AP_j)\}$

  Compute the decoding order in S based on the residual noise tolerance.

  $isSatisfiable \leftarrow Satisfiable(S, \mathcal{A})$

  if $isSatisfiable$ = *false* **then**

    $E_i \leftarrow false$

    $S \leftarrow S \backslash \{(C_i, AP_j)\}$

return S

CHAPTER 4

MOZART: ORCHESTRATING COLLISIONS IN WIRELESS NETWORKS

4.1 Introduction

Previous studies have shown that the throughput achieved in real wireless networks is often significantly lower than their capacity [72]. This difference is frequently attributed to *discrepancy between the interference at the transmitter and the receiver.* In wireless networks, it is impossible for transmitters to precisely estimate the interference at receivers which leads to hidden and exposed terminal scenarios. It has been shown [72] that hidden and exposed terminals can cause throughput loss in 30% and 61% links, respectively. Collisions due to hidden terminals are especially common in indoor wireless networks with obstacles [10]. Secondly, IEEE 802.11 protocol and its derivatives (e.g., 802.11 with RTS-CTS, 802.11ec [55], ZigZag [29] etc.) require the channel to remain idle when the nodes are undergoing backoffs. Such *idle listening* is necessary in these protocols to avoid collisions and leads to up to 30% loss of throughput [40].

The above two factors significantly impact the throughput. Most practical distributed scheduling algorithms in today's wireless networks such as the IEEE 802.11 family and IEEE 802.15.4 protocols are based on CSMA (Carrier Sense Multiple Access). Various improvements to these schemes have been proposed in recent years. RTS-CTS (IEEE 802.11)

based approaches have been proposed for addressing the hidden terminal problems. Approaches to reduce the overhead of RTS-CTS packets [55], channel wastage due to collided packets [69], and overhead of backoff [68] have been proposed. Several mechanisms that attempt to salvage bits or entire packets out of collided transmissions have also been proposed [29,33,38,43,52]. However, these algorithms are insufficient (details in Sec. 4.7) as they either do not work in multi-collision domains [29, 33], abandon the bits under collision [38,69], require multiple antennas per node [68] or have long critical periods resulting in lower packet delivery ratio [55].

This chapter presents a new cross-layer algorithm called *Mozart*, that encourages collisions and hidden terminal transmissions in a planned way to enable fast recovery of colliding packets via a new approach called *Successive Packet Subtraction* (SPS). In Mozart, receivers encourage neighboring nodes to transmit simultaneously resulting in collision. The receiver then smartly suppresses transmissions in subsequent slots based on its estimation of signal strengths from various senders so as to best apply SPS to recover all the colliding packets (See Figure 4.1). *SPS allows wireless receivers in a multi-collision domain network to receive and decode multiple packets without knowing which node has data to send to which other node.*

Mozart eliminates backoffs before transmission of data packets. Mozart also embraces hidden terminal transmissions and therefore implicitly addresses it. Thus, Mozart provides throughput improvement for both *downlink* and *uplink* traffic. Through theoretical analysis, we show that the *critical period* of Mozart is 2μs. This is much shorter than the critical period of the state-of-the-art schemes that have critical period of 39.4 μs [55]. The shorter critical period ensures higher packet delivery ratio in Mozart. This thesis, makes the following additional contributions:

- We propose Successive Packet Subtraction (SPS), a new approach for wireless nodes to simultaneously receive multiple packets and then decode them one-by-one while

controlling the retransmission pattern. Unlike SIC (Successive Interference Cancellation [70]), SPS works even when all packets have low SINR.

- On receiving collided packets, receivers in Mozart need to identify the set of transmitters and estimate their received signal strengths with high precision. Doing both in the presence of multiple collisions is extremely hard. Existing techniques to measure signal strength do not work in the presence of collisions [33]. We present a novel iterative algorithm for estimating the RSS of multiple colliding packets. Our results show that even in the presence of interference from 14 other packets, Mozart is able to estimate the RSS within 1 dB with 96% accuracy compared to 21% accuracy of state of the art schemes.

- This chapter proposes an algorithm to compute the set of nodes to be suppressed at the end of each slot such that the probability of correctly decoding the packets is maximized across all slots.

- We implement Mozart on a USRP testbed [1]. Our evaluation results show that Mozart's throughput is up to 3.70x compared to IEEE 802.11. Our trace-driven evaluations show that on an average, Mozart provides a throughput of 2.55x and 4.10x compared to 802.11ec and 802.11, respectively.

The rest of the chapter discusses our algorithms, implementation and comparison results in detail.

4.2 Mozart: Detailed Description

Mozart works by *encouraging collisions* among transmissions. However, upon collision, the receiving node controls the future retransmissions such that it is able to decode the

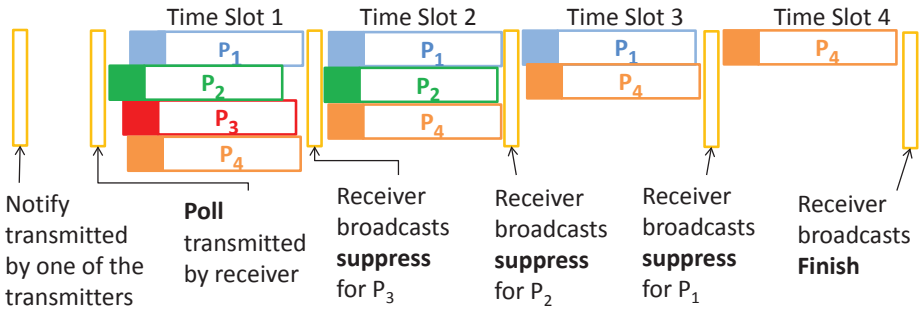


Figure 4.1: Collision Recovery Period. Through control messages, the receiver ensures that the number of transmitters is reduced by 1 from the previous slot. Data transmissions in the same slot may arrive at different times at the receiver due to propagation delays and radio's TX-RX turn-around time. At the end of the recovery period, the receiver will reconstruct samples for P_4 and subtract it from samples received in slot 3. The remaining samples are then decoded to obtain P_1. Similarly, P_2 and P_3 are decoded from the samples of slot 2 and slot 1, respectively.

collided transmissions in a short time. The next subsection explains the working of Mozart in detail. Section 4.3 explains how Mozart handles various practical challenges.

4.2.1 Successive Packet Subtraction (SPS)

In Mozart, packet transmissions and the subsequent decoding takes place in multiple phases (Fig. 4.1):

- **Notification**: If a node B has data for A, then B sends a notification message to A, indicating that B has outstanding data for A.

- **Polling**: Upon receiving the notification message, A backs off for a random duration (between 1 and 5 μs as described in Subsection 4.3.4) and sends a *poll* message to solicit transmissions in its neighborhood. After sending the poll, A is said to be

undergoing *recovery*. However, in Mozart, a node (say A) can poll only if all the following conditions are true: (i) It observes (through virtual sensing) that no other polling node in its neighborhood is *recovering*; (ii) No other node in its neighborhood is transmitting data to its receiver; and, (iii) A's backoff is over. If any of these condition is not true, A undergoes a backoff and then attempts to poll again.

- **Data Transmission**: Upon receiving a poll from a neighboring node (say A), a node (say B) sends a data packet to A if both these conditions are true: (i) There is no other node (except A) in its neighborhood undergoing recovery; and, (ii) B has data to send to the polling node.

- **Suppress**: The polling node may simultaneously receive data packets from multiple nodes resulting in a collision. From the received packets, the polling node selects one transmitting node and transmits *suppress* to it. Upon receiving suppress, the node stays silent for the remaining duration of the recovery period. Other nodes upon overhearing the suppress, re-transmit the same data packets until they either receive the *suppress* or the *finish packet* from A. Section 4.3.3 explains how the polling node selects the node to whom the suppress is sent. The receiver also stores the received samples in a buffer for decoding in the future. Mozart does not have any synchronization overhead since synchronization happens *implicitly* through the poll and suppress messages transmitted by the receiver.

- **Finish**: In each slot one of the transmitting nodes becomes silent after receiving the suppress from A. So, the number of colliding packets decreases by one in each slot. Eventually, in some slot, only one node transmits. At the end of that slot, the polling node decodes all the packets. Upon successful decoding, A broadcasts a *finish* packet. The finish packet serves two purposes: (i) It indicates successful decoding, thereby acknowledging the transmitted data, allowing nodes to transmit

new data; and, (ii) Indicates the end of the recovery period, thus allowing nearby nodes to transmit poll packets or data packets.

By reserving the channel in the neighborhood of the receiver (using the poll message) as well as by encouraging collisions among transmitters, Mozart is able to implicitly handle *hidden transmissions*. Similarly, before transmitting a poll packet or a data packet, a node ensures that no other node in its neighborhood is recovering. This check prevents its transmissions from interfering with any data packets that the recovering node in the neighborhood might receive. To determine if any other node is recovering, nodes use virtual sensing by monitoring poll/suppress packets and the matching finish packets. If a node receives a poll packet but not the corresponding finish packet, it indicates that the other node is recovering. In case when the node does not receive the matching finish packet, timeout after the last received poll/suppress packet can be used to indicate the end of recovery. The timeout duration is set to twice the channel access time as discussed in Section 4.3.2. If a node receives no packet transmissions after polling, then it sends a finish and enters a backoff period before polling again.

Packet Structure: To transmit the suppress at the end of each slot, the receiver needs to determine the id of at least one transmitter. However, with colliding transmissions, the receiver cannot decode the transmitter's ID from the packet's MAC header. To that end, Mozart requires that transmitters identify themselves by sending a Correlatable Symbol Sequence (CSS) before the preamble. It has been shown [55] that CSSs[1] can be correlated without correcting for frequency or sampling offsets. Another property of CSSs is that they can be correlated even under collision. This allows a node to receive a control message even if its neighboring node is transmitting. Thus, Mozart allows neighboring nodes to simultaneously transmit, thereby eliminating the *Exposed Terminal Problem*. Further,

[1] Also referred to as Pseudo Random Sequences (PN sequences)

usage of PN sequences reduces the transmission duration of the packet considerably as demonstrated before [55].

Mozart uses 5 different packet types: (i) **Notify**: It consists of the PN sequence of the receiver; (ii) **Poll**: It consists of the PN sequence of the polling node; (iii) **Suppress**: It contains the PN sequence of the receiver followed by the PN sequence of the node being suppressed; (iv) **Data packet**: This packet has the PN sequence of the transmitter, PN sequence of the receiver, preamble, MAC header and the data body; and, (v) **Finish**: It contains the PN sequence of the receiver, followed by PN sequences of the nodes whose transmissions were successfully decoded. *By using PN sequences that are correlated [55] by the receiver, Mozart significantly reduces the rate of loss of the control packets due to collisions.*

Packet Decoding: In Mozart, the receiver decodes all received packets in the *reverse chronological order*. Decoding starts with the last slot in which only one node transmitted. Figure 4.1 shows a recovery period in which four nodes send data in response to A's poll. Three suppress packets were sent at the end of the first three slots, and, so in the fourth slot, only one node transmits. In that slot, P_4 is available in the clear and can simply be decoded. To decode P_1, after correcting for different offsets (details in Section 4.3.6), A recreates samples for P_4 as received in slot 3 and subtracts them from samples received in slot 3. After subtraction, it is left with only the samples corresponding to P_1 which A decodes. Similarly, to decode P_2, A re-creates samples for both P_1 and P_4 and subtracts them from the samples received in slot 2. After subtracting, it decodes the remaining samples to get P_2. This process is repeated for each slot until A has decoded all the collided packets. Thus, by carefully selecting the retransmitters, A is able to decode 4 packets in 4 slots.

SPS allows the receiver to control the retransmission pattern without precisely knowing the complete set of transmitters in each slot. This is in contrast with a naive approach where all transmitters transmit their PN sequences in the first slot and then the receiver schedules

different transmitters in a TDMA fashion. In this naive approach, the receiver may not detect PN sequences with low SINR (See Section 4.4) resulting in their starvation. On the other hand, when using SPS, such transmitters will be detected in later slots when the number of colliding packets become small. *Thus, the flexible approach of SPS increases fairness.* This approach is also different from 802.11 polling mode where the APs poll all the potential transmitters. This results in wasted polls and throughput loss when a transmitter has no data to send in response to a poll. In contrast to that, receivers in Mozart receive packets from all the transmitters and decode them by controlling the retransmission pattern.

4.2.2 Challenges towards practical implementation

Although the idea behind Mozart is simple, however, multiple challenges need to be solved to make Mozart practical:

- **Identifying set of transmitters and estimating their RSS:** To maximize the decoding accuracy, at the end of each slot, receivers in Mozart need to carefully determine the transmitter to be suppressed. As explained in Section 4.3.3, this requires receivers in Mozart to determine the following additional information: (i) ID of transmitters; and, (ii) SINR of transmitters. However, determining this information for all packets in the presence of interference is challenging. Existing techniques [33] to measure signal strength do not work in the presence of collisions and require transmitters to transmit one-by-one, thus constituting a significant overhead.

- **Determining transmitter to suppress:** Once the set of transmitters is determined, the receiver in Mozart needs to determine which transmitter should be suppressed among all transmitters such that the decoding accuracy is maximized. Different transmitters may have different SINR and may transmit at different physical layer

data rates. This makes it difficult to determine which transmitter should be suppressed.

- **Handling heterogeneous data rate and packet sizes:** The channel access time is fixed for all transmitters in Mozart. However, if some transmitters only have small amount of data to send, this may lead to channel wastage.

4.3 Practical Considerations

This section explains how we handle the aforementioned challenges. In Subsection 4.3.1, we explain how receivers in Mozart identify the set of transmitters and estimate their RSS. Subsection 4.3.2 explains how transmitters in Mozart determine the physical layer data rate to be used. Once the receiver knows the id of transmitters, their RSS and the modulation scheme used, the receiver then uses the algorithm described in Subsection 4.3.3 to determine the set of nodes to be suppressed at the end of each slot. Subsection 4.3.4 explains how compared to existing algorithms, Mozart significantly increases the probability of successful packet decoding. The next subsection explains how receivers in Mozart handle decoding errors. Finally, Subsection 4.3.6 explains how offset correction and PN sequence assignment is done in Mozart.

4.3.1 Identification and RSS estimation of collided packets

It is beneficial for Mozart to identify the transmitters of all the collided packets, since with more IDs, it is more likely that the receiver will find the right set of transmitters to suppress. Observe that the receiver does not need to determine the source ID of all the packets. However, more IDs it can determine, higher are the chances that the receiver will suppress the right set of transmitters (explained later in Section 4.3.3).

For -8dB SINR, state of the art correlation schemes have been shown [55] to suffer from

as much as 70% false negatives when 127-symbol Gold code sequence (a type of CSS) is used. Clearly, this false-negative rate is too high for Mozart where multiple transmissions may collide resulting in low SINR. Our identification algorithm is also based on the general approach of computing the cross correlation, however, there are significant differences when compared to existing schemes [29, 55, 69]:

- **Bounding cross correlation by circular padding**: To improve the accuracy of correlation, we harness a property of Gold codes that guarantees that the circular cross correlation of two instances of Gold code is bounded [28]. For 127-symbol sequence Gold codes, the bound is 1/7. Thus, the correlation works better when Gold codes collide only with other Gold codes, and not with samples from arbitrary data packets. Since in Mozart, the nodes do not undergo backoffs before transmitting data, the PN sequences for the collided packets are expected to be *approximately-synchronized*. However, due to propagation delays and hardware artifacts, it is possible that the PN sequences do not collide with other PN sequences at the receiver. Based on the IEEE 802.11 standard [36], the arriving time for different PN sequences can differ by at most 4μs (2μs for radio's turn-around time [55] and 1 μs for propagation delay in each direction [55]). So the transmitters in Mozart cyclically pad 2μs symbols of the Gold code before and after the actual Gold code. The symbols padded before a certain Gold code instance are the last few symbols of Gold code while the symbols padded after the end are the first few symbols. This ensures that the Gold code samples interfere with only the combination of samples of other Gold codes resulting in low cross-correlation. Figure 4.2(b) illustrates the padding process.

- **Improving correlation for low RSS packets through cancellation**: Instead of trying to detect all the collided Gold codes at the same time, Mozart decodes one Gold code at a time. Then, it subtracts the samples of the detected Gold code from the received samples. Figures 4.2(c) and 4.2(d) show two packets with Gold codes and

data before subtraction and after Gold code of P_1 was subtracted from the collided samples. This process is similar to the one that we used during the packet cancellation step (See Section 4.2). This allows Mozart to detect those packets that have low RSS even if they are interfered by high RSS packets since after subtracting the high RSS packet, the correlation value for the lower RSS packet increases. Subtracting packet 1 reduces the noise for P_2 and thus, helps in improving the SINR of the Gold code in P_2 as shown in Figures 4.2(c) and 4.2(d).

- **Iteratively improving the RSS estimation**: In the above two steps, it is possible that due to high interference from other packets, the computed correlation is not correct resulting in inaccurate estimation of RSS. In such a case, even after subtraction, a part of the packet (residual samples after subtraction) would still be left in the original samples (albeit with much lower RSS). To correct the RSS estimate, Mozart performs correlation and cancellation repeatedly to detect these copies of the packet. For this, Mozart records the correlation value and the location (i.e., starting time) for each detected packet. When Mozart detects some packet again that was already detected in previous steps, then the power level of the packet is updated to be the sum of the current and old correlation values. As Mozart detects the same packet multiple times, its estimate of the RSS of the packet improves while the residual signal strength of the packet decreases. For examples, Mozart detects Gold codes of Packet 1 first in Figure 4.2(c) and later in Figure 4.2 (e) among the residual samples. Upon, detection for the second time, the receiver estimates the RSS of the residual Gold code of Packet 1 and adds that estimate to the previous estimate of RSS of Packet 1. This step is repeated as long as the correlation value is at least twice the strength of the residual samples.

Thus, the first technique is used by the transmitter while second and third techniques are iteratively used by the receiver for improving the accuracy of identifying the set

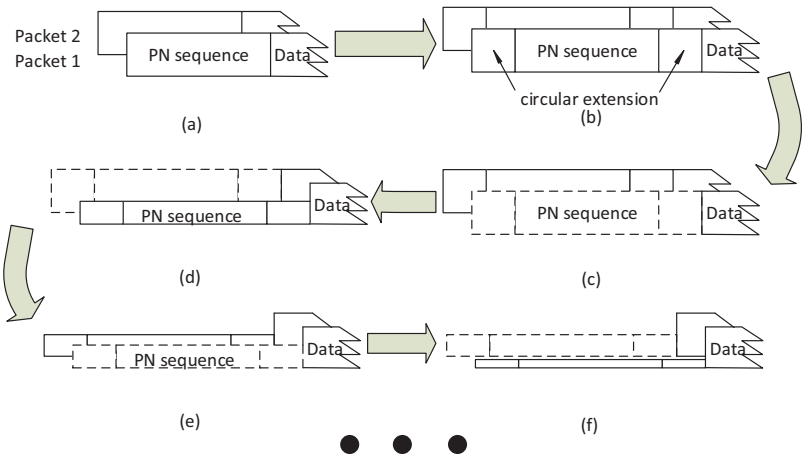


Figure 4.2: Iterative algorithm for transmitter identification and RSS estimation.

of transmitters and estimating their RSS. Apart from this, the receivers in Mozart need to determine the their modulation and coding scheme used by each of the transmitters. In our algorithm, multiple CSSs are assigned to represent different physical layer data rates. The transmitter of the data packets include the CSS that corresponds to the data rate used for that particular data packet.

4.3.2 Heterogeneous data rate and packet sizes

Due to varying SNR, different transmitters may select different physical layer data rates when transmitting to the same receiver. Since, wireless channels are bidirectional in nature (shown in [30]), so the transmitters in Mozart use the received poll packet to estimate the channel to the receiver. To that end, the transmitters correlate the received poll packet with the known PN sequence. The peak value of the correlation indicates the channel quality between the transmitter and the receiver with higher correlation value indicating better channel quality. This is similar to estimating the channel quality through preamble and has been well studied [29] in the context of improving the accuracy of recreating the

samples. However, the key difference is that here, the transmitter estimates the channel to the receiver using the received poll packet and then, transmits the data packet at a rate that is suitable for the channel's current condition. *Thus, the poll packet transmitted by receivers in Mozart also allows transmitters to pick the appropriate physical layer data rate without extra overhead.*

During each recovery, Mozart fixes the channel access time for each slot. Thus, a node with higher data rate may be able to send more bits compared to a node with lower data rate. However, if a node does not have enough pending data, then it will reduce its data rate such that its transmission still fits in the slot size. This reduced data rate sometimes allows the receiver to simultaneously decode two transmissions using Successive Interference Cancellation (SIC) [70]. SIC is a well known physical layer technique that is used by the wireless receivers to simultaneously decode two packets where the packet with the higher *noise tolerance* is decoded first, followed by the packet with the lower noise tolerance. Since in Mozart, multiple transmitters transmit simultaneously, this increases the probability that the receiver will find a pair of packets among the received transmissions that satisfy the requirements for simultaneous decoding.

For the example scenario of Figure 4.1, let's say that the transmitter of P_1 has small amount of pending data. Thus, on receiving the poll, it would transmit P_2 at the lowest possible data as explained above. At the end of first slot, the receiver observes that P_1 has been transmitted at low physical layer data rate and high power, and thus can be decoded even in the presence of P_3 as noise. So, it will send suppress to both P_1 and P_3. When decoding, the receiver decodes P_1 and P_3 using SIC and sends a *finish* packet indicating successful decoding. Mozart is able to harness SIC benefits since here, the transmitters *proactively* reduce their data rate. Thus, the results described in this chapter are not in conflict with [70] where the authors had argued that SIC has limited applicability when not applied proactively.

4.3.3 Determining set of nodes to suppress

Observe that during reverse chronological decoding, the residual noise is higher in the earlier slots since more packets need to be subtracted before actual decoding (*e.g.* when decoding P_3 in Fig.4.1, the residual noise would come from P_1, P_2 and P_4). Therefore, to improve the decoding accuracy, receivers in Mozart suppress transmissions with high noise tolerance in earlier slots. This approach maximizes the probability of successful decoding across all the slots of the recovery, thereby improving the resilience. Further, to minimize the number of slots, Mozart first checks if it is possible to suppress two nodes simultaneously as explained in Sec. 4.3.2.

Next, we explain Algorithm 3, that determines the set of nodes to be suppressed at the end of the current slot. Here, T_{ij} denotes the *noise tolerance* (in mW) of the pair of packets P_i and P_j, which denotes the maximum residual noise level that can be present during successful decoding of these two packets. If a pair of packets has high value of T_{ij}, it implies that it is possible to decode both the packets using SIC even if residual noise left after canceling other packets is high. The algorithm also computes (Line 4) the noise threshold (in mW), τ that indicates the expected residual noise that will be left after the receiver has subtracted all packets that were received in this slot. τ is approximated by dividing sum of RSS (in mW) of all packets received in this slot by 100 (i.e., 20 dB)[2]. Then, in Lines 5-10, all those pairs of packets are added to the set $\mathbf{G}$ that have sufficient noise tolerance. If noise tolerance of P_j is higher than that of P_i (Line 6-7), then P_j would be decoded in presence of interference from P_i. Thus, its noise tolerance would be $\frac{RSS_j}{r_j} - RSS_i$ where r_j is the minimum SINR at which P_j can be successfully decoded with high probability [33]. On the other hand, noise tolerance of P_i would simply be $\frac{RSS_i}{r_i}$ since

[2]20 dB denotes the cancellation that can be achieved by subtracting the packet in most cases (See Section 4.4 for detailed experiment results).

P_j would already be subtracted. For successful simultaneous decoding, we need to ensure that the residual noise is less than the minimum of these two values (Lines 7-9).

Finally, the algorithm returns the pair that has the highest noise tolerance (Line 10). However, for some slots, it may not be possible to do SIC and thus no pair of packets may have the required tolerance. Then (Lines 12-15), Mozart finds a single packet that has the highest noise tolerance. It is also possible that during the recovery period, the receiver does not find any node with noise tolerance above the expected residual noise. In that case, the receiver handles the errors as explained in Subsection 4.3.5.

4.3.4 Near-Zero Critical Period

We define *Critical period* of a MAC algorithm as the duration of the interval, before and after the beginning of a transmission, during which an interfering transmission may corrupt *both the transmissions*. *MAC protocols with longer critical periods are expected to have a higher collision rate and thus, lower throughput.* The longer critical period also requires the nodes to spend excessive time in backoff. In Mozart, if a single receiver receives multiple colliding packets, it can still decode all those packets by suppressing one transmitter in each slot. So, intuitively, Mozart should have shorter critical period compared to existing algorithms. In A.1, we formally show that the critical period of Mozart is 18.7 μs[3]. By comparison, the critical period of some standard protocols are A.1: 152 μs for 802.11 with RTS-CTS and 39.4 μs for 802.11ec.

4.3.5 Handling Decoding Errors

While decoding, it is possible that the channel noise or residual noise may cause a failed checksum resulting in decoding error. In that case, the receiving node follows one of the

[3]To reduce the probability of two neighboring transmissions from colliding, we require nodes to backoff for a short duration (5 μs) before sending a poll.

Algorithm 3: Computes the set of nodes that should be sent suppress in this slot

Input: For each packet P_i received in this slot: its power level expressed in mW (RSS_i) and the minimum SINR level (r_i, expressed as a dimensionless ratio) at which it can be decoded. r_i is contingent upon the physical layer data rate used to transmit P_i.

Output: Set of packets (or corresponding transmitters) that should be sent suppress in this slot.

$\mathbf{P} \leftarrow \{$Set of packets received in this slot$\}$, $\mathbf{G} \leftarrow \{\}$

$\tau \leftarrow \frac{Sum\ of\ RSS\ of\ all\ packets\ in\ \mathbf{P}}{100}$

for $(P_i, P_j) : P_i, P_j \in \mathbf{P}$ **do**

  if $\frac{RSS_j}{r_j} \geq \frac{RSS_i}{r_i}$ **then**

    $T_{ij} \leftarrow \min\{\frac{RSS_j}{r_j} - RSS_i, \frac{RSS_i}{r_i}\}$

  else $T_{ij} \leftarrow \min\{\frac{RSS_i}{r_i} - RSS_j, \frac{RSS_j}{r_j}\}$

  if $T_{ij} > \tau$ **then** $\mathbf{G} \leftarrow \mathbf{G} \cup \{(P_i, P_j)\}$

if $\mathbf{G} \neq \{\}$ **then return** $\arg\max_{(P_i,P_j)\in\mathbf{G}} T_{ij}$

else

  $T_i \leftarrow \frac{RSS_i}{r_i}\ \forall\ P_i \in \mathbf{P}$

  $\mathbf{S} \leftarrow \{P_i : P_i \in \mathbf{P} \wedge T_i > \tau\}$

  if $\mathbf{S} \neq \{\}$ **then return** $\arg\max_{P_i\in\mathbf{S}} T_i$

  else Send finish

three options: (i) It first requests re-transmission of data for that slot. For example, in Figure 4.1, if receiver A is unable to decode P_2 during slot 2, then it will ask P_2's transmitter to re-transmit. The receiver performs this notification by sending a special *nack* message to the transmitter of P_2. Upon receiving the retransmitted data, A can decode P_2 and continue to decode P_1 using the slot 1 samples. (ii) However, if P_2's transmitter is not able to transmit P_2 because one of its other neighbors is undergoing recovery, then A decodes the remaining packets by simply subtracting the samples of P_2 from the earlier slots (without actually decoding P_2). The receiver would decode as many packets as possible using this scheme. (iii) Finally, if the receiver decoded only a subset of packets, then it would send *finish* with the PN-sequence of only the transmitters whose packets it was able to decode.

4.3.6 Offsets Correction and PN Sequence Assignment

The efficiency of successive packet subtraction in Mozart depends on the accuracy of re-creating the samples. To increase the accuracy, the phase, frequency and sampling offsets need to be compensated [29]. This problem of computing the offsets in presence of interference has been well studied [29, 30, 43]. In our implementation, after evaluating multiple schemes, we decided to use the one proposed in ZigZag [29] as it gave the best results. Mozart also requires that every node in the network be assigned a PN sequence. Further, no two neighboring nodes should be assigned the same PN sequence. Magistretti et al. [55] argue that such an assignment can be either done by APs or can be done using hash functions. In the experiments and evaluations of Mozart, the PN sequences were assigned using the former approach.

4.4 Experiments

In this section, we describe the results from our experiments performed on the GNU radio platform and a testbed of Universal Software Radio Peripheral (USRP) N210 version 4 [1]

radios. We used WBX daughterboards [1] as the RF front end. Mozart performs decoding at the sample level, and thus, is independent of the modulation and coding choice for transmission. In our experiments, we used BPSK with 1/2 convolutional code. Our recreation process compensates for the phase offset, frequency offset and the sampling offset as described in Section 4.3.6. In this section, we also measure the accuracy of transmitter identification and RSS estimation for the proposed algorithm (Sec. 4.3.1) as well as the existing state of the art technique. The frequency used was 1078 MHz and the physical layer data rate was set to 62.5 Kbps. The rate was kept low to ensure that the delay between the host computer and the radio was *small* compared to the packet duration. Thus, the relative delay values would approximately match the delay values from off-the-shelf wireless cards.

In Mozart's implementation, the receiver stores all the received samples offline which are later used to compute the number of successful transmissions and the throughput. Besides Mozart, we also implemented a version of the IEEE 802.11 protocol. One of the challenges in implementing 802.11 was that USRP has different hardware parameters compared to the commercial 802.11 cards. For example, due to higher latency from the radio to the host computer, the packet decoding time was observed to be around 150 μs which prevents the receiver from sending an ACK within the SIFS duration. So, using experiments, we re-measured the optimum values of all 802.11 parameters (SIFS, DIFS, slot size, ACK timeout) for the N210 hardware as per their definitions. For example, we ensured that slot size was such that if two neighboring nodes choose consecutive values of backoff, then one of the nodes will sense the other's transmission and will not send its own packet. Due to the aforementioned reasons, the packet size was kept constant, and thus the receivers had no opportunity to use SIC to do simultaneous decoding.

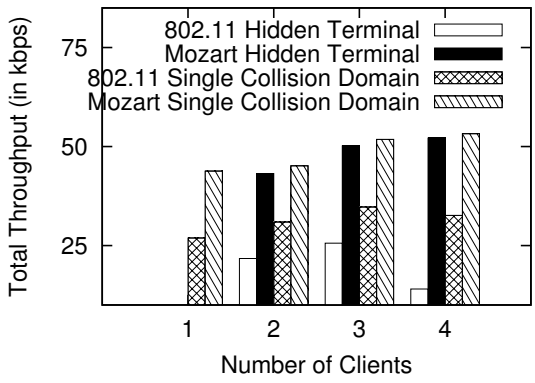
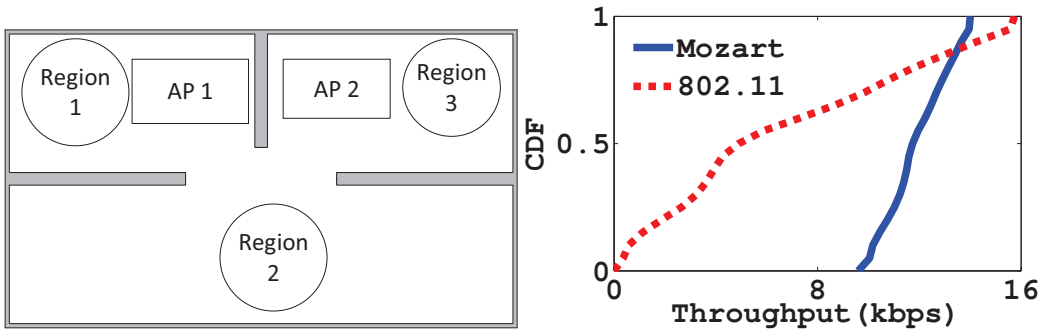


Figure 4.3: Experiment Results with single AP.

4.4.1 Testbed Results

Single AP: In this experiment, we use a single N210 node as the AP and vary the number of clients. We place the client in various positions around the AP to create two types of topologies: (i) **Single collision domain:** Where all clients can hear each other; and, (ii) **Hidden Terminal:** Where no two clients can hear each other. The throughput results are shown in Figure 4.3 with varying number of clients. Mozart increases throughput due to fewer collisions and close to zero backoff as compared to the IEEE 802.11 protocol. When the nodes are hidden to each other, the throughput gain provided by Mozart is much higher due to increased collisions in 802.11. With 4 hidden nodes, Mozart provides 2.70x more throughput than IEEE 802.11 protocol.

Multiple APs: In the next experiment, we set up two N210 nodes as APs and four others as clients. For this experiment, we placed the two APs in two different rooms (as shown in Figure 4.4a). Each of the four clients were placed at different locations in the three regions so as to create ten different topologies (hidden, non-hidden, mix etc.). In all the topologies, the clients associated to the AP with the strongest signal. Figure 4.4b shows the CDF of the throughput of clients for both Mozart and 802.11. Averaging over



(a) Experiment Topology (b) CDF of throughput

Figure 4.4: Experiment topology and results with multiple APs.

all topologies, 802.11 provides throughput of 6.9 Kbps per node while Mozart (no SIC) provides 12.3 Kbps, an increase of 78%.

4.4.2 Experimental Analysis of Micro Benchmarks

In this section, we present micro benchmark results from our experiments. The computed micro benchmarks are also used as input to our evaluations (Sec. 4.5).

Sender Identification and RSS Estimation: We measure the accuracy of sender identification through multiple experiments. For this (See Figure 4.5a), we ensured that all the packets have equal RSS (Equal RSS is the worst case for evaluations since *all* transmitters have low SINR in this case). As Figure 4.5a shows that when 20 packets collide, Mozart identifies 12 more senders compared to the existing approach [55]. The false positives were observed to be less than 1% for all the schemes.

Next, we studied the accuracy of RSS estimation of Mozart compared to existing techniques [55]. Figure 4.5b shows the probability that the estimated RSS of a packet is within 1dB difference of the actual RSS under different SNR values (computed without considering other packets as noise). When 15 packets (each having 20 dB SNR) collide, the

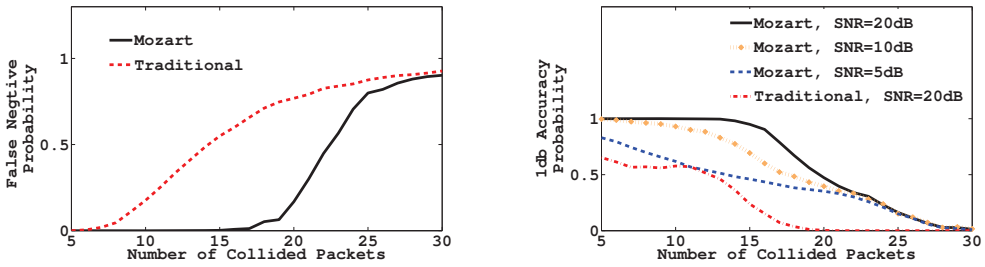


Figure 4.5: Detection and RSS Estimation of colliding packets for Mozart and traditional approach [55] under equal RSS setting (worst case analysis).

probability that Mozart estimates RSS of the collided packets within 1dB of the actual RSS is 0.96 compared to 0.21 for existing techniques [55], an improvement by a factor of 3.57x.

Subtraction Accuracy: In this experiment, we measure the cancellation accuracy. To quantify the subtraction accuracy independent of modulation and coding, we define metric, ΔP that represents the power reduction achieved through cancellation. ΔP quantifies the achieved reduction in power after subtracting P from a set of collided packets. We define $\Delta P(in\, dB) =$ RSS of P (in dBm) $-$ Residual Power of P left after subtracting P (in dBm). A high value of ΔP implies that the residual noise is lower and it increases the decoding probability of the remaining packets.

Results: Figure 4.6 shows the variation in cancellation accuracy (ΔP metric) with variation in number of samples in the packet and the SINR of the packet being subtracted. The figure shows that for any packet consisting of more than 1600 samples, the cancellation is quite efficient. Figures 4.6b presents the Cumulative Distribution Function (CDF) of the distribution of ΔP for 4000 samples per packet. From Figure 4.6b, we see that as expected, the SINR for the subtracted packet decreases, the variation in cancellation accuracy (ΔP

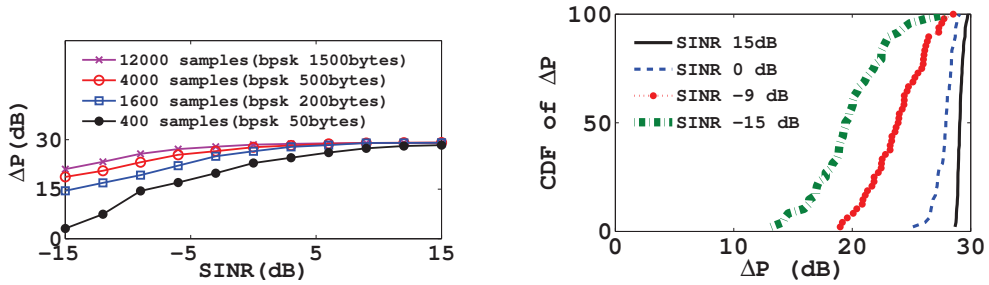


Figure 4.6: Evaluation of packet subtraction: Higher ΔP implies lower residual noise and better cancellation accuracy. (a) Variation in ΔP with varying SINR of the subtracted packet. (b) CDF of ΔP when packet has 4000 samples.

metric) increases. The results from our experiments are also fed in the ns-3 simulator as explained in the next section.

4.5 Comparison Results

To evaluate the performance of Mozart, we conducted extensive trace-driven ns-3 evaluations. In this section, we explain our evaluation setup and the results.

To make evaluations more realistic, we first setup a testbed of 40 nodes (comprised of laptops) and collected the RSS values between all pairs of nodes. The nodes were distributed over two floors of our building and spanned multiple rooms. This RSS data was then fed into the ns-3 simulator. We randomly designated varying number of the nodes as APs while the remaining nodes were designated as clients. Each client associated with the AP from which it received the strongest signal. In the evaluations, TCP connections were established between each client and its AP. For this, we downloaded the previously collected traffic traces during SIGCOMM [67] and computed the pdf distribution of packet sizes and also the pdf distribution of packet inter-arrival time over all connections. These

two pdf distributions were used to generate both uplink and downlink traffic. To create network saturation condition, the number of connections between each client and its associated AP was set to 20.

Further, the results from the experiments (See Section 4.4) were fed into the ns-3 simulator as follows: (i) **PN-Sequence detection accuracy**: In the evaluations, we used the values from Figure 4.5 for determining if a PN-sequence can be detected or not; (ii) **Residual noise level**: The power of the residual noise was fed from the data collected from experiments (See Figure 4.6); and, (iii) **Imprecise Signal Strength Estimates**: For Mozart, imprecision in signal strength estimations could lead to unsuccessful decoding. We fed the imprecision from our experiments (See Figure 4.5b), into our evaluations.

Apart from Mozart, we also implemented and evaluated the following algorithms: (i) **Optimal omniscient zero-overhead slotted TDMA**: Here, we assume a central scheduler knows the interference between all links at all possible data rates. This scheduler computes the set of links that should be active in a given slot. For this, at the beginning of the slot, the APs first collect information about which node has data to send to which other node. The nodes convey this information using short PN sequences. The APs forward this information to a central scheduler that computes the set of active links and sends this information to APs. This information is then broadcasted back by the APs (again using short PN sequences). To put the optimal TDMA in the best light and to eliminate the effect of the backbone capacity, the APs and the central scheduler were connected through wired Ethernet with unlimited bandwidth and zero latency. (ii) **802.11ec [55]**: 802.11ec reduces the overhead of RTS-CTS packets by encoding them as CSSs. (iii) **IEEE 802.11 without RTS-CTS**. In Mozart, transmitters picked the best data rate based on channel conditions as explained in Section 4.3.2. Channel quality information was also provided to transmitters in Optimal TDMA out-of-band (*i.e.* without any overheads.). In all other algorithms, the transmitters varied their data rate using ARF algorithm [42].

4.5.1 Results for bidirectional TCP Traffic

Next, we evaluated the performance of different algorithms for TCP traffic. When Mozart is used for downlink traffic (AP to client), then the length of the recovery period was always 1 since the client would receive packet from at most one AP.

Throughput: We vary the number of APs in the network and compute the total throughput over all nodes (Fig. 4.7a). On average, Mozart provides 4%, 155% and 310% more throughput than TDMA, 802.11ec and 802.11 algorithms, respectively. These differences are primarily because of three factors:

- **Collisions**: From Fig. 4.7c, we see that in other algorithms the percentage of transmissions that are acknowledged is significantly lower resulting in retransmissions and lower throughput. In Mozart, some of the transmissions may not be decoded by the receiver due to error in correlation of the PN-sequence or because of higher residual noise. However, when the number of colliding transmissions is low, Mozart decodes all of the transmissions resulting in $\geq 95\%$ acknowledgment rate. The high acknowledgment rate also implies that the decoding accuracy of Mozart exceeds 95%. This can be attributed to high accuracy of transmitter identification, RSS estimation and careful selection of the transmitters to be suppressed (Sec. 4.3.3). Although, 802.11ec reduces the overhead of control packets, still its critical period (See Section 4.3.4) is sufficiently large leading to high collision rate. For 802.11 and 802.11ec, with increase in number of APs, the higher SINR of different links results in higher acknowledgment rate.

- **Backoff**: Figure 4.7d shows the total time spend by nodes in backoffs between every successful transmission (averaged over all successful transmissions). Observe that transmitters in other algorithms spend a significant amount of time in backoffs. In IEEE 802.11 and its derived protocols, nodes decrement their backoff counter only

if the channel is idle, implying that the channel resource is wasted in these protocols due to high backoff. On the other hand, nodes do not experience backoffs during the recovery phase in Mozart as discussed in Section 4.3.4.

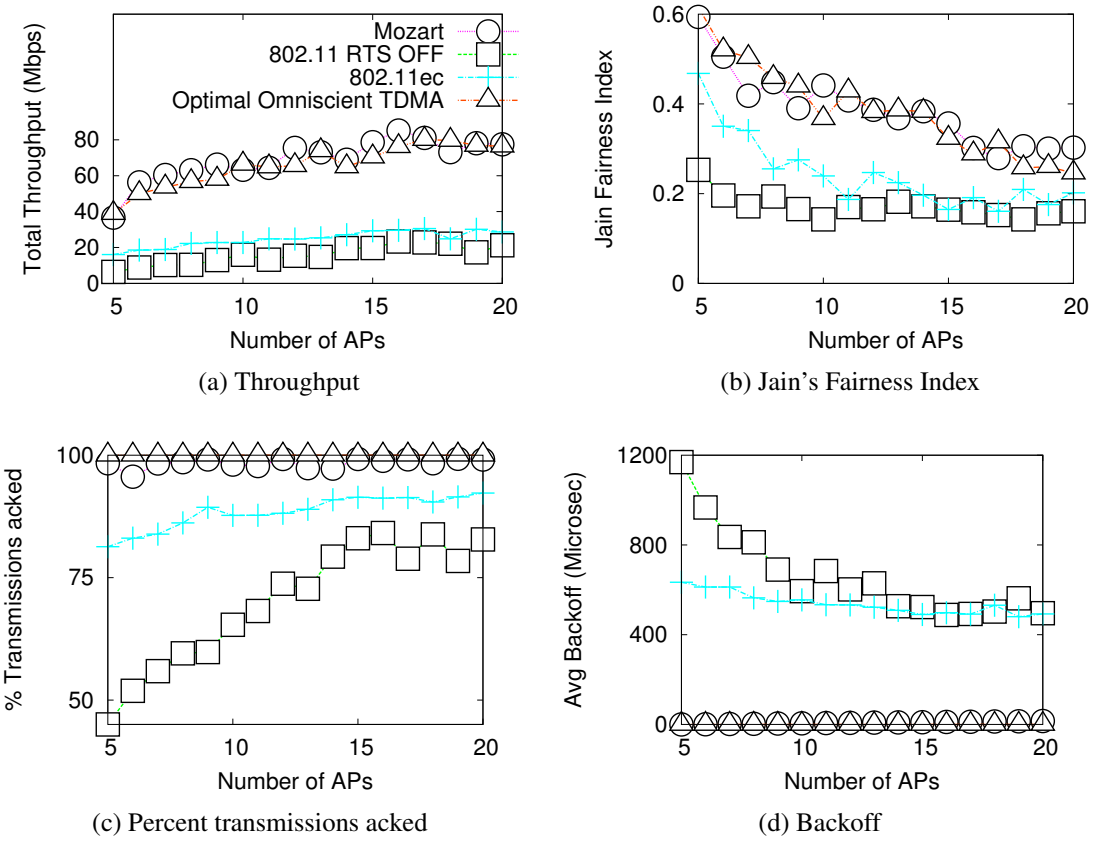
- **SIC Applicability**: In our evaluations, we observed that on average, Mozart was able to apply SIC in 10.25% of the slots. This allows Mozart to have a higher throughput than Optimal Omniscient TDMA since the TDMA scheduler uses fixed slot lengths, irrespective of the amount of the data to be transmitted by the user. Although, receivers in Mozart are also unaware of the amount of pending data, still the SPS-based decoding enables them to leverage SIC. This allows the Mozart receivers to decode multiple packets simultaneously (Sec. 4.3.2), resulting in higher throughput.

Fairness: We use Jain's Fairness Index to compare the fairness for different protocols. Fig. 4.7b shows that with fewer APs, Mozart provides higher fairness compared to 802.11 and 802.11ec. This long term starvation in IEEE 802.11 and its derived protocols is consistent with the previous literature [35]. With an increase in the number of APs, the fairness index for all algorithms is almost equal due to higher SNR of the links.

4.6 Discussion

Here, we discuss how Mozart can be extended to make it more suitable for real networks.

Co-existence with legacy 802.11 devices: Mozart uses the NAV feature of IEEE 802.11 to ensure coexistence with legacy 802.11 devices. Upon overhearing a MAC packet, the node reads the "Duration" field of the MAC header and does not initiate any transmissions for that duration. To ensure coexistence with 802.11, packets in Mozart can be modified as follows: (i) Before transmitting poll or suppress PN sequences, the receivers transmit a zero-payload packet with duration field equal to the duration of one slot; and, (ii) Similarly, transmitters also transmit a zero-payload packet before transmitting the data



(a) Throughput

(b) Jain's Fairness Index

(c) Percent transmissions acked

(d) Backoff

Figure 4.7: Comparison of different algorithms for TCP traffic.

packet. These packets will prevent 802.11 from interfering with Mozart. To give a fair chance to 802.11, receivers in Mozart also need to undergo certain amount of backoff before transmitting. We leave the detailed methodologies of coexistence that ensure fairness for our future work.

Handling interference from partially overlapped channels: Partially overlapping channels in Wi-Fi can lead to loss of throughput as the data packets may not be decodable due to interference from devices operating on partially overlapped channel. To handle such interference, the receivers in Mozart can transmit the poll packet at reduced power level. This will force the transmitters to use data rates lower than optimal, thereby enabling

successful decoding of packet even in the presence of interference from partially overlapping channels. The amount by which the receiver reduces its power level depends on the interference it experiences from partial overlapping channels. We leave the detailed discussion and analysis of Mozart's performance in presence of such interference for future work. Further, the receivers can also employ the approaches mentioned before (Sec. 4.3.5) to handle decoding errors.

Compatibility with MIMO nodes: Mozart is compatible with wireless nodes with multiple antennas. If a node has N antenna, then Mozart allows such nodes to receive and decode N packets per recovery slot. To enable this, the receivers in Mozart would suppress N packets in each slot, thus reducing the length of the recovery period by a factor of N.

4.7 Related Work

Collisions are known problems for wireless networks. Currently, carrier sensing (CSMA) is used in WLANs (Wireless LAN Networks) to avoid collisions. To further reduce the collision probability, various other schemes have been introduced such as RTS-CTS. However, transmitting RTS-CTS control packets at low physical layer data rate leads to significant overheads that become worse at higher data rates such as those observed in 802.11g or 802.11n networks. Further, RTS-CTS packets do not prevent collisions when interference range is higher than the transmission range even though they cause unnecessary blocking of transmissions [63].

Recently, Magistretti *et. al.* have proposed 802.11ec [55] that employs Correlatable Symbol Sequences (CSSs) to replace the RTS-CTS packets, thereby significantly reducing the overhead of control packets. However, the transmissions of nodes may still end up colliding due to the longer critical window of duration 39.4μs (discussed in Section 4.3.4) resulting in backoffs and unnecessary retransmissions. When the density of the clients increases, more collisions will happen because of increase in simultaneous transmissions,

resulting in further loss of throughput. On the other hand, with a shorter critical period of 2μs, Mozart has fewer collisions.

Apart from preventing collisions, other protocols have been proposed that either try to abort the collided transmissions or salvage the bits that did not undergo a collision. CSMA-CN [69] proposes to use two antennas at the transmitter for receiving the collision notifications from the receiver. Partial Packet Recovery [38] tries to recover bits that did not undergo collision. However, both CSMA-CN and PPR will abandon the collided samples, resulting in loss of throughput.

ZigZag [29] tries to utilize the collided information by employing ZigZag decoding. This reduces the number of useless retransmissions but the nodes still waste time in back-offs. Further, in ZigZag collided packets will be wasted if they are intended for different destinations since one of the transmitter may receive ack from its intended receiver and may not retransmit its data packet. In CRMA [52], nodes transmit the same information on different sub-channels. The receiver decodes the collided transmissions by solving a set of linear equations. However, for optimal channel utilization, nodes in CRMA require good estimation about the network load at other transmitters as well.

Mozart is a receiver-driven cross-layer algorithm that takes a different approach where it encourages multiple transmitting nodes to collide. Receiver-driven protocols have been proposed before in context of wireless sensor networks [77]. However, there the main objective was to reduce the energy consumption instead of maximizing the throughput. Recently proposed, AutoMAC [33] also encourages collisions among transmissions. However, AutoMAC implicitly assumes that all nodes are in a single collision domain. All its analysis, experiments and traces are also for single collision domain. In multi-collision domain networks, the receivers may face interference from transmitters that are transmitting to some other receiver and thus, can't be suppressed using Speculative Ack [33]. It is possible that if two AutoMAC receivers are in the range of two transmitters such that

one receiver can suppress only one transmitter, then the decoding may take a very large number of slots. This also makes extending AutoMAC to multi-collision domain networks non-trivial. Secondly, receivers in AutoMAC estimate the channel state by requiring transmitters to send non-overlapping training symbols one-by-one. On the other hand, Mozart's receivers can estimate the channel state of multiple colliding transmitters resulting in lower overhead.

Successive Interference Cancellation (SIC) [70] has been used before to decode interfering packets in WLANs [29]. Mozart uses SIC only when at least one transmitter has reduced its physical layer data rate to fit the slot width. Thus, Mozart is able to derive benefit from SIC and is not in conflict with existing literature [70].

4.8 Conclusions

In this chapter, we presented Mozart, a cross-layer algorithm that takes a new approach of encouraging collisions among nodes. By doing so, Mozart handles the hidden terminal collisions as well as reduces the critical period of transmissions. To implement Mozart in practice, we presented novel algorithms for identifying multiple transmitters as well as estimating their RSS in presence of interference. USRP-testbed based evaluations show that, Mozart throughput is up to 3.70x compared to IEEE 802.11. Evaluations performed using real-world traces show that Mozart's throughput is 2.55x and 4.10x when compared to 802.11ec and 802.11, respectively.

CHAPTER 5

R2D2: EMBRACING DEVICE-TO-DEVICE COMMUNICATION IN NEXT GENERATION CELLULAR NETWORKS

5.1 Introduction

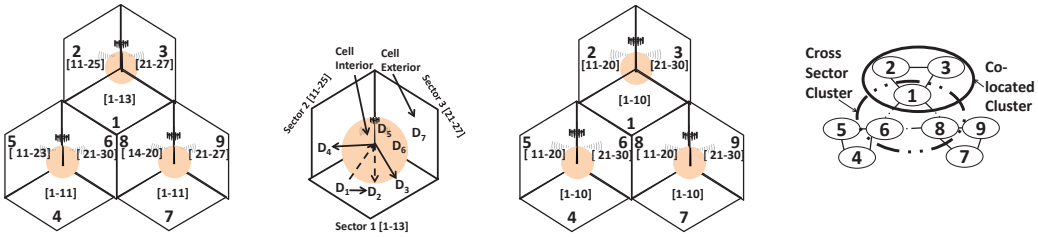
Device-to-device (D2D) communications is being pursued as an important feature [3] for the next generation cellular networks (LTE-advanced). Being an underlay to cellular networks, the goal is to leverage the physical proximity of communicating devices to improve cellular coverage in sparse deployments, provide connectivity for public safety services and improve resource utilization in conventional deployments [20, 51]. We focus on D2D's ability to improve resource utilization in this work.

D2D can improve resource utilization in two ways: (i) *offload:* a data session between two devices (D1, D2) in the same sector which conventionally incurs two hop transmissions in the cellular mode (D1→BS, BS→D2) now requires only a single hop transmission (Fig. 5.1b) in D2D mode (D1→D2), and (ii) *reuse:* leveraging the physical proximity, the D2D communication can further operate on resources on which conventional cellular users are already scheduled. While existing works [20, 21, 39] have highlighted the benefits of both these components, the study was not conducted under practical multi-cell deployments with an inherent pattern of resource reuse (called fractional frequency reuse, FFR) for the cells. Indeed, we show that in multi-cell deployments with even a static FFR pattern, the existing reuse provided by the cellular deployment leaves little room for D2D communication to

provide additional reuse. Hence, most of D2D's gain is restricted to its ability to offload cellular traffic.

Given the large spatial and temporal variations in traffic load in practice [61], it is important to consider offloading with D2D in the presence of dynamic FFR schemes. Here, D2D brings both an opportunity as well as a challenge. (i) While the uplink (UL) and downlink (DL) radio resources are fixed across different base stations, their traffic load can vary significantly within a cell. With D2D traffic capable of being scheduled in both UL and DL resources, it presents an opportunity in that it serves as a *flexible* load that can be intelligently placed (in DL/UL resources) to efficiently utilize the cell's net radio resources (see example in Fig. 5.2). This can help both during dynamic FFR pattern determination as well as during scheduling, albeit at the expense of coupling the DL and UL resource allocation problems that have conventionally been addressed separately. (ii) In a sectored deployment with dynamic FFR, while cross-sectors (*e.g.*, 1, 6, 8 in Fig. 5.1a) need to split resources to alleviate interference; co-located sectors (*e.g.*, 1, 2, 3 in Fig. 5.1a) can potentially schedule their cellular users (*e.g.*, D_3 and D_4 in Fig. 5.1b) on overlapping resources (say RB 11) without interference (due to *directional* BS transmission/reception). However, with the introduction of D2D traffic that is *omni-directional*, this creates interference conflicts between co-located sectors on shared resources (*e.g.* between $D_1 \leftrightarrow D_2$ and D_4 on RB 11) that diminishes the resource utilization and reuse capability of dynamic FFR (Fig. 5.1a), potentially offsetting the offloading benefits from D2D. Hence, the challenge is to alleviate such interference conflicts either as part of the FFR solution and/or through joint scheduling of D2D and cellular traffic *across* the co-located sectors. Thus, we find that intelligent D2D traffic placement (during FFR) coupled with scheduling of D2D and cellular traffic jointly on DL and UL resources as well as across co-located sectors is critical to achieve effective traffic offloading and resource utilization.

Toward addressing these challenges, we propose *R2D2*- a framework for holistic Radio



(a) Dynamic FFR (b) D2D usage (c) Static FFR (d)

Figure 5.1: (a)-(c): [] indicates the set of Resource Blocks (RBs, *i.e.* time-frequency allocation units [3]) available for allocation to cell exterior traffic. Total number of RBs to be allocated is 30. (d) Graph for the network shown in Fig. 5.1a.

Resource Management (RRM) with D2D communication in cellular networks. *R2D2* operates at two time scales. At the *beginning of every epoch* (lasting several tens of frames), *R2D2* estimates the average traffic (resource) demand from cellular and D2D traffic in each sector in either direction (DL and UL) based on history (from previous epochs). It partitions the network into disjoint (small) clusters of interfering sectors (called cross sectors) and leverages the flexible nature of D2D traffic to *jointly* determine the dynamic FFR patterns for DL and UL for each of these clusters in a completely distributed and localized manner. Based on the dynamic FFR pattern determined, the resources of operation are determined for each sector in the cluster in DL and UL directions. Then, *in every frame*, for the set of sectors co-located at the same base station and instantaneous traffic demands, *R2D2* solves the coupled problem of D2D traffic placement and scheduling of cellular and D2D traffic jointly on both the DL and UL resources as well as across the sectors. Thus, while the coarse time-scale component in *R2D2* allocates resources and removes interference only between cross sectors (through dynamic FFR) for an entire epoch, the fine time-scale component is responsible for alleviating the interference between co-located sectors

generated by D2D traffic (through joint sector scheduling) and maximizing the utilization of allocated resources in every frame in the epoch.

While the first step in *R2D2* is solved efficiently through a light-weight and scalable dynamic FFR scheme, solving the scheduling problem in the second step is an NP-hard problem. Toward solving this per-frame scheduling problem, *R2D2* designs efficient algorithms with performance guarantees that are also amenable to implementation at the time scale of a frame (1 ms in LTE). Our evaluations with realistic LTE settings reveal that intelligent scheduling of D2D and cellular traffic when combined with appropriate D2D traffic placement can provide gains of 2.79x in resource utilization (throughput) compared to existing schemes [39]. Our contributions in this work are multi-fold:

- Contrary to existing belief, we show that while D2D communications offer offloading benefits, their ability to reuse cellular resources is limited in multi-cell environments, where FFR patterns are already deployed in practice.

- We propose *R2D2* - A framework for holistic approach to efficient offloading with D2D traffic. *R2D2* incorporates a two time-scale solution: localized computation of dynamic FFR patterns jointly for DL and UL (leveraging flexible nature of D2D traffic) in each cluster of cross sectors at epoch granularity; and intelligent scheduling of cellular and D2D traffic *jointly* on both the allocated DL and UL resources as well as across co-located sectors in every frame.

- Toward solving the scheduling problem in each frame, we provide a $\frac{1}{2}$ approximation algorithm with a complexity of $O(N^2K^3)$, where N and K are the number of OFDMA resource blocks and users in each sector respectively. We also provide an alternate $\frac{1}{4}$ approximation algorithm that has a lower complexity of $O(N^2K)$. Through extensive evaluations, we show that in practice, the performance of both algorithms are significantly better than their worst-case guarantees and are within

5% of the optimal. While these algorithms apply to TDD systems, for FDD systems, where a D2D traffic session is constrained to not be allocated resources from both DL and UL simultaneously, we provide an $\frac{1}{3}$ approximation algorithm.

5.2 Background

We consider an OFDMA based next generation cellular network (LTE is used as a reference). The network could be time (TDD) or frequency (FDD) divisioned, where downlink (DL) and uplink (UL) resources for all the cells in the network are determined a priori (same across all base stations). Coverage area of all Base Stations (BSs) is typically divided into three sectors[1] with the help of 120^o directional (sectored) antennas and each sector is considered to be a separate cell in itself for operational purposes (See Fig. 5.1a). Each frame in LTE is 1 ms long and consists of time-frequency allocation units called resource blocks (RBs) on which users' data are scheduled.

LTE advanced users will potentially [3] support D2D communication with their peers that will avoid data having to go through the BS and the mobile core network when it is destined for users in the same sector. We focus on network-assisted D2D communication within the same sector as it is more realistic without requiring neighboring base stations to interact/coordinate with each other. Here, although, data is transferred between the peers directly in D2D communication, the control signaling still goes through the BS and scheduling of D2D traffic is also managed by the BS.

Inter-sector interference in cellular networks is handled with the help of FFR patterns. In the popular 1-3 FFR scheme, the DL (UL) spectrum is divided into four fixed-size frequency bands. One band is used by all the cell-interior clients (in each BS), who do not see interference due to the close proximity to their BS, while the other three bands are split

[1]Discussions and solutions are also applicable to other sectorization models.

across the three sectors (Fig. 5.1c) of a BS to mitigate interference with sectors of adjacent base stations.

5.2.1 Related Work

D2D in single cell: D2D communication has been considered before [20, 22, 25, 39, 44, 50, 62] in the context of single base stations and with no sectorization. However, cellular deployments are multi-cell and sectored (120^o) in nature and employ FFR patterns. Further, [22,25,39,44,50] restrict their focus to reusing the D2D transmissions with only the uplink transmissions resulting in under-utilization of resources.

D2D with complete information about path loss: Doppler *et al.* [21] consider the D2D mode-selection problem in multi-cell scenarios. However, to decide which RB should be allocated to a given D2D transmission, their algorithm requires knowledge of path loss between all pairs of D2D transmitters/receivers and non-D2D transmitters/receivers. This constitutes a significant signaling overhead.

Dynamic FFR: Algorithms have been proposed for dynamic FFR algorithm. However, existing algorithms are centralized [6, 18] and/or D2D-oblivious [75]. The centralized approach to computing the FFR allocation and configuring all the base stations (BSs) in every frame is not feasible for cellular networks. Further, the D2D oblivious FFR algorithms are expected to have low throughput when there is an asymmetry in UL and DL traffic load.

5.3 Benefits and Challenges

As discussed in Section 5.1, D2D can provide gains by offloading and by reusing resources already allocated to cellular users in the same sector. This section highlights the true potential for D2D in practical multi-cell networks and the associated challenges in realizing their benefits. We use simulations from an LTE system with a D2D underlay (details in Section 5.5) to emphasize our inferences.

5.3.1 Potential for Reuse from D2D

Existing works [20, 39] have attempted to leverage reuse by D2D links in the absence of multiple cells. However, in multicell deployments with FFR schemes, as discussed in Section 5.2, spectral resources are already reused in cells in adjacent BSs. Hence, for a D2D link to reuse resources without impacting existing cellular transmissions in the same sector, the separation between the D2D devices should be much smaller compared to that between the D2D transmitter and cellular user operating on the same resource (on say DL) in either the same or an adjacent sector. Such opportunities are however, not common, especially given the omni directional nature of D2D communications (see Fig. 5.1b) and small sector sizes. Even if such opportunities arise, to be able to detect and leverage such opportunities, the channel gains between all cellular/D2D and D2D users need to be measured and reported to the BS, which must then use this information to perform per-frame scheduling. Clearly, accomplishing this entire process within a single frame is not feasible due to the prohibitive overhead.

Finally, to understand the potential for reuse from D2D in a hypothetical scenario, we compare three schedulers in the presence of 1-3 static FFR (Fig. 5.1c): (i) Requires all traffic to go through the BS by classifying all traffic (even D2D) as cellular; (ii) Does not allow D2D traffic to reuse RBs allocated to cellular users in the same sector; and, (iii) A *genie* scheduler that has the channel gain information between all users as well as between users and BS along with power control (details of schedulers, LTE simulation settings, D2D traffic classification etc. are covered in Section 5.5) and uses that to schedule D2D transmissions. In the first two schedulers, each sector independently assigns resources to different transmitters by solving a single cell resource allocation problem [7]. The result in Fig. 5.3a indicates that *only a marginal gain comes from the reuse of resources by D2D traffic even with a hypothetical genie scheduler, while a large portion of the gain from D2D comes from its offloading capability*.

5.3.2 Challenge and Opportunity in D2D Offloading

Traffic Variations across Sectors: It is important to align cellular resources to cater effectively to traffic load variations across cells that are common in practice [61]. This is realized with the help of dynamic FFR schemes that allocate spectral resources to interfering cells, taking into account their traffic load. Interestingly, a simple distributed solution can be applied to realize dynamic FFR in a sectored deployment. Each sector belongs to a cluster of cross sectors (located at different cell-sites) and a cluster of co-located sectors (Fig. 5.1d). Dynamic FFR can be applied independently to each disjoint cluster of three (cross) sectors, whose cell-exterior traffic interfere with each other. Since each sector belongs to exactly two clusters (Fig. 5.1d), once the operational resources are determined for each sector by dynamic FFR, it is possible that its allocated resources may overlap with those of neighboring co-located sectors in the other cluster. Note that, this is not a problem if all traffic is cellular since the co-located sectors employ directional (sector) antennas to begin with (*e.g.*, sectors 1 and 2 in Fig. 5.1b can simultaneously transmit to D_3 and D_4, respectively). This allows for applying the above light-weight dynamic FFR scheme for cellular traffic that operates locally on individual clusters of cross sectors alone. However, with the introduction of D2D traffic that is omni-directional, such an approach would not work and would create interference between co-located sectors on overlapping resources (*e.g.* between $D_1 \leftrightarrow D_2$ and D_4 on RB 11 in Fig. 5.1b) that could bring down the benefits from dynamic FFR as well as D2D offloading.

We compare the performance of two systems - one that classifies all traffic as cellular, and another that classifies traffic as cellular and D2D. Both systems first apply dynamic FFR to the clusters with cross sectors and then perform scheduling in each of the sectors independently and then assign resources to different transmitters by solving a single cell resource allocation problem [7]. The result in Fig. 5.3b clearly indicates the magnitude of

the interference created by D2D that did not exist in a pure cellular environment, resulting in the D2D system performing worse than one that classifies all traffic as cellular.

The interference generated by D2D traffic can be addressed as part of the dynamic FFR process, where co-located sectors are incorporated in the FFR process. However, this would couple all the cross and co-located sectors across the network, preventing the dynamic FFR scheme from no longer being local and light-weight. An alternate approach is to *retain the light-weight nature of dynamic FFR (only applied for cross sectors) but to alleviate the interference generated by D2D through intelligent scheduling of cellular and D2D traffic in each cluster of co-located sectors jointly*. We will adopt the latter approach in this work.

Traffic Variations within a Sector: In addition to traffic variations across sectors, there is also a lot of asymmetry [61] in traffic load between uplink (UL) and downlink (DL) that changes both spatially and temporally. However, the spectral allocations to the UL and DL cannot be varied dynamically across sectors as this would lead to asymmetric interference between UL and DL traffic across cells, which is a much harder problem to address. For this reason, in practical systems, a resource block can either be used for UL traffic or DL (but not both), and this classification is same across all cells. This makes it difficult to leverage the traffic load variations between UL and DL within each cell. In this regard, we note that placing the D2D traffic on DL or UL resources does not have any relative advantages - placing it on the DL resources may create interference to cellular users from D2D transmitter, while placing it on the UL resources may create interference from the cellular user to D2D receiver. Hence, D2D can be used as a *flexible* load to balance and match the UL and DL traffic load to their available resources, resulting in better resource utilization.

We compare the benefits of a system that employs D2D as a flexible load against one that classifies all traffic as cellular. Both the systems use dynamic FFR but the former

computes the FFR resource allocation using D2D as a flexible load (See Sec.5.5 for details). Fig. 5.3c clearly indicates that the performance of the former system is resilient to variation in traffic disparity while the latter loses 85% throughput when traffic disparity is high. However, to realize these benefits, one would need to solve the D2D traffic placement problem, both as part of dynamic FFR as well as during scheduling. A unique aspect of this problem is that compared to conventional cellular systems that solve the DL and UL resource allocation problems separately, *addressing the D2D traffic placement problem would entail solving the DL and UL resource allocation problems jointly*.

In summary, most of the benefits from D2D arise from its offloading capability and not from reuse. To maximize the offloading gains from D2D, it is important to solve the problem of intelligent D2D traffic placement (during FFR) coupled with schedule of D2D and cellular traffic jointly on DL and UL resources as well as across co-located sectors.

5.4 *R2D2*: RRM with D2D

5.4.1 Overview of *R2D2*

To retain the localized nature of dynamic FFR schemes for scalability without sacrificing performance, we propose *R2D2*- a holistic approach to Radio Resource Management with D2D communication in cellular networks. *R2D2* operates at two time scales.

- At the beginning of each epoch (lasting several tens of frames), *R2D2* leverages the flexible nature of D2D traffic to determine the dynamic FFR patterns for DL and UL jointly for each of cross-sector clusters (Fig. 5.1d) in a completely localized manner. Based on the dynamic FFR patterns determined, the radio resources of operation are determined for each sector in the cluster in DL and UL directions for the entire epoch.

- The burden of alleviating interference generated by D2D traffic between co-located

sectors is moved to the frame level time granularity. Here, in every frame, for every cluster of co-located sectors, *R2D2* solves the coupled problem of D2D traffic placement and intelligent scheduling of cellular and D2D traffic on both the DL and UL resources jointly across the sectors.

Note that, *R2D2* carefully assigns the coarse time-scale dynamic FFR process to the cross sectors and the per-frame joint scheduling to the co-located sectors, as coordinating the latter practically comes for free (due to co-location at the BS). *This step ensures that R2D2 is able to effectively leverage spatial reuse of the resources while requiring finer cooperation among only the sectors that are co-located at the same base station.* We now explain each of these components of *R2D2* in detail.

5.4.2 D2D Traffic Classification

Flows that originate and end in the same sector can possibly be offloaded to D2D. Let $\hat{r}_{d2d}, \hat{r}_{uplink}$ and $\hat{r}_{downlink}$ be the average physical layer data rates achievable from a given transmitter to a given receiver using D2D, from transmitter to the BS and from the BS to the receiver, respectively. These rates are estimated based on channel quality information from reference signal received power (RSRP) measurements that span over all RBs [3, 9]. At the beginning of every epoch, a BS in *R2D2* offloads a flow to D2D if both of the following conditions are satisfied: (i) Flow originates and ends in the same sector; and, (ii) $\frac{1}{\hat{r}_{d2d}} < \frac{1}{\hat{r}_{uplink}} + \frac{1}{\hat{r}_{downlink}}$. The second condition ensures that the time taken by the flow over D2D is less than the time taken by the flow when routed through the BS. These data rates can be estimated by the BS from the corresponding SINR values. Note that since control messages and signaling overhead are involved in setting up a D2D session, this process cannot be invoked at the same granularity of per-frame scheduling. This justifies the rationale behind employing average data rates for D2D traffic classification at an epoch granularity.

5.4.3 Dynamic FFR in *R2D2*

At the epoch level time granularity (several tens of frames), *R2D2* is executed locally by all the three sectors that form a cross-sector cluster (See Fig.5.1d).

Step 1 (Traffic Classification): In a dynamic FFR solution, the size of the four bands in the FFR pattern are adapted and chosen to meet the requirements from both cell-interior and exterior traffic. Further, the FFR patterns can be re-configured at coarse time scales (several tens of frames) to track the traffic load variations. In the case of a cellular user, simple SINR thresholds are used to classify the user as a cell-interior or exterior user (*e.g.*, D_5 in Fig. 5.1b is an interior user due to its high SINR with the BS). For a D2D link, its classification is done taking into account both the devices in the link. We consider a D2D link as interior traffic only if both ends of the link are cell-interior users; otherwise the pair is classified as exterior traffic (*e.g.*, $D_5 \leftrightarrow D_6$ in Fig. 5.1b is interior traffic while $D_5 \leftrightarrow D_7$ is exterior.) .

Step 2 (Resource Demand Estimation): For each cross-sector cluster, *R2D2* estimates the average traffic (resource) demand from cellular and D2D traffic in each sector for the current epoch based on information from previous epochs. It keeps track of the aggregate resource allocation (R, in RBs) to the interior and exterior traffic of both cellular and D2D users in each epoch (for every sector j) and computes the estimate for average resource demand ($\bar{R}$) for the current epoch in sector j as a weighted (α) moving average: $\bar{R}_{j,xyz}(t) = \alpha R_{j,xyz}(t-1) + (1-\alpha)\bar{R}_{j,xyz}(t-1)$. Here $x = \{C, D\}$ indicates cellular (C) or D2D (D) traffic; $y = \{i, e\}$ indicates interior (i) or exterior (e) traffic; and $z = \{d, u\}$ indicates DL (d) or uplink (u) traffic in sector j. The above equation estimates the average resource demand for a given traffic type in a particular direction.

Step 3 (Determining resources allocated in the cross-sector cluster): Let N_d and N_u be the total available resources in DL and UL spectrum, respectively. This spectrum needs to be shared across the three cross sectors. Further, we also need to determine how much

spectral resources would be allocated to interior traffic. Let $F_{0,d}$ and $F_{0,u}$ be the number of RBs allocated to interior DL and UL traffic, respectively that is common to all the cross sectors. $F_{j,d}$ and $F_{j,u}$ denote the number of RBs allocated to exterior DL and UL traffic, respectively for sectors $j = 1, 2, 3$. The allocation of resources (*i.e.* dynamic FFR pattern) can be formalized as a linear optimization problem shown in (5.1). This formulation exhibits the following properties: (i) Leverages the flexible nature of D2D traffic to carefully split it across UL and DL resources; (ii) Maximizes the total traffic demand satisfied; (iii) Allows interior traffic to be scheduled on exterior traffic resources but not vice-versa as the latter would receive interference from interior traffic in the neighboring cross sectors; and, (iv) Ensures proportional fairness across the three sectors. Using $A_{j,xyz}$ to indicate the resources (number of RBs) allocated to traffic type xy (cellular/D2D, interior/exterior) in direction z (DL/UL) in sector j, we have:

$$\max_{A_{j,xyz},F_{0,z},F_{j,z}\forall j,x,y,z} \sum_{x\in\{C,D\}} \sum_{y\in\{i,e\}} \sum_{z\in\{d,u\}} \sum_{j\in\{1,2,3\}} A_{j,xyz} \tag{5.1}$$

$$\text{subject to } F_{0,z} + \sum_{j\in\{1,2,3\}} F_{j,z} \leq N_z \; \forall z \in \{u,d\} \tag{5.2}$$

$$A_{j,Ciz} \leq \bar{R}_{j,Ciz} \; \forall j \in \{1,2,3\} \; \forall z \in \{u,d\} \tag{5.3}$$

$$A_{j,Ciz} + A_{j,Cez} \leq \bar{R}_{j,Ciz} + \bar{R}_{j,Cez}$$
$$\forall j \in \{1,2,3\} \; \forall z \in \{u,d\} \tag{5.4}$$

$$A_{j,Diu} + A_{j,Did} \leq \bar{R}_{j,Di} \; \forall j \in \{1,2,3\} \tag{5.5}$$

$$A_{j,Did} + A_{j,Diu} + A_{j,Ded} + A_{j,Deu} \leq \bar{R}_{j,Di} + \bar{R}_{j,De} \; \forall j \in \{1,2,3\} \tag{5.6}$$

$$A_{j,Ciz} + A_{j,Diz} \leq F_{0,z} \forall j \in \{1,2,3\} \; \forall z \in \{u,d\} \tag{5.7}$$

$$A_{j,Cez} + A_{j,Dez} \leq F_{j,z} \forall j \in \{1,2,3\} \; \forall z \in \{u,d\} \tag{5.8}$$

$$\frac{\bar{R}_{i,Ceu} + \bar{R}_{i,Ced} + \bar{R}_{i,De}}{F_{i,u} + F_{i,d}} = \frac{\bar{R}_{j,Ceu} + \bar{R}_{j,Ced} + \bar{R}_{j,De}}{F_{j,u} + F_{j,d}}$$
$$\forall i,j \in \{1,2,3\} \tag{5.9}$$

Here, (5.1) requires us to maximize the total resource allocation (traffic demand satisfaction) over all the three cross sectors. (5.2) requires that the total DL (UL) resources allocated to interior and exterior traffic in all the three sectors cannot be more than total DL (UL) resources available in the system. (5.3) and (5.4) require that the allocation to interior traffic and net traffic be no more than the interior and net traffic demands respectively in each sector in either direction for cellular traffic. These two constraints together allow for allocation of exterior traffic resources to interior traffic, while at the same time preventing exterior traffic from operating on interior traffic resources. A similar set of constraints apply to D2D traffic in (5.5) and (5.6), with the additional *flexibility that the D2D traffic demand can be met by DL and UL resources jointly, thereby allowing for better resource utilization*. (5.7) and (5.8) require the net allocation to interior and exterior traffic in each

sector in either direction be limited by the interior and exterior traffic resources respectively that would be made available from dynamic FFR. Finally, (5.9) ensures a proportional allocation across the cross sectors.

Observe that the resource allocation variables $F_{0,z}$, $F_{1,z}$, $F_{2,z}$ and $F_{3,z}$ are required to be integers. However, we solve the above optimization problem after relaxing that constraint and later round off the values to the nearest integer such that all resources are allocated to at least one sector. This makes the above optimization a linear optimization problem that can be solved efficiently. Further, it needs to be executed only once every epoch resulting in low overhead. Note that generally the number of resources to be allocated is a large number, and thus, rounding off does not cause significant loss of optimality.

5.4.4 Joint Cellular and D2D Scheduling in R2D2

At frame level granularity, *R2D2* resolves the conflicts generated by D2D traffic due to localized dynamic FFR. In the process, it also performs efficient scheduling of cellular (interior and exterior) and D2D traffic (interior and exterior) jointly across DL and UL resources as well as across co-located sectors to maximize resource utilization.

Scheduling Model: Scheduling problems are typically formulated as utility maximization problems, where the objective is to maximize the end-to-end system throughput subject to a desired fairness model (captured by the utility function, U_k). We assume proportional fairness (PR, $U_k = \log(\bar{r}_k)$) that is the de-facto fairness model in cellular networks [54]. The problem now reduces to $\max_k \beta_k \log \bar{r}_k$, where β_k captures the priority weight of user k's QoS class and $\bar{r}_k$ its average throughput. The system solution can be shown to converge to the optimum PF allocation at longer (epoch) time scales if the scheduler's decisions at each frame are made to maximize the aggregate marginal utility, $S_{\max} = \arg\max_S \{\sum_{k \in S} \Delta U_k\}$ [73]. ΔU_k denotes the marginal utility received by

user k in a valid schedule S and is given by $\frac{\beta_k r_k}{\bar{r}_k}$ for PF, where r_k is the aggregate instantaneous rate received by the user in the frame. Thus, in each frame t, user weight $v_k(t) = \frac{\beta_k}{\bar{r}_k(t)}$ varies with $\bar{r}_k(t)$ and accounts for both fairness and QoS. The scheduling problem at each cell-site of co-located sectors then reduces to determining the frame schedule for the co-located sectors that maximizes the following aggregate weighted rate: $S_{\max}(t) = \arg\max_S \sum_{k \in S} v_k(t) \cdot r_k(t)$

Problem Formulation: Next, we formulate our joint DL-UL scheduling problem. We use $x_{k_j,n}$ to denote if user k_j in sector j is scheduled on RB n. $F_{j,z}$ denotes the number of exterior resources available to sector j in direction z (UL/DL). $F^j_{0,z}$ denotes the number of interior resources available to j in direction z. *R2D2* computes both these values for all the sectors in both the directions as explained in Section 5.4.3. Observe that different sectors in a co-located cluster can differ in the number of interior resources available ($F^j_{0,z}$).

$$\max_{x_{k_j,n}\forall k_j,n,j} \sum_{j\in\{1,2,3\}} \sum_{k_j\in\mathcal{K}_j} v_{k_j} \sum_{n\in F_j} x_{k_j,n} r_{k_j,n} \tag{5.10}$$

$$\text{where,} \quad \mathcal{K}_j \leftarrow \mathcal{C}_{j,d} \cup \mathcal{C}_{j,u} \cup \mathcal{D}_j$$

$$F_j \leftarrow F^j_{0,d} \cup F_{j,d} \cup F^j_{0,u} \cup F_{j,u}$$

$$\text{subject to} \sum_{k_j\in\mathcal{K}_j} x_{k_j,n} \leq 1, \ \forall n \in F_j, \forall j \tag{5.11}$$

$$x_{k_j,n} = 0; \forall n \in F^j_{0,d} \cup F_{j,d}$$

$$\forall k_j \in \mathcal{C}_{j,u}, \forall j \tag{5.12}$$

$$= 0; \forall n \in F^j_{0,u} \cup F_{j,u}$$

$$\forall k_j \in \mathcal{C}_{j,d}, \forall j \tag{5.13}$$

$$= 0; \forall n \in F^j_{0,u} \cup F^j_{0,d}$$

$$\forall k_j : k_j \text{ is exterior}, \forall j \tag{5.14}$$

$$\sum_{n\in F_j} x_{k_j,n} r_{k_j,n} \leq B_{k_j}; \forall k_j \in \mathcal{K}_j, \forall j \tag{5.15}$$

$$x_{k_j,n} = \{0,1\}; \quad r_{k_j,n} = h(n, k_j, k_\ell, k_m) \tag{5.16}$$

where $\mathcal{C}$ and $\mathcal{D}$ represent the cellular and D2D traffic; and $\mathcal{K}_j$ denotes the set of users in sector j. (5.11) indicates that in a given sector, at most one transmission can be scheduled on each resource block (RB). (5.12) and (5.13) indicate that every RB in the UL and DL can be allocated only to their respective cellular users. (5.14) prevents an exterior user from using an RB allocated to interior traffic. (5.15) limits the allocation to each user to be bounded by the finite amount of data in its buffer.

TDD vs. FDD: We note that a D2D user may be allocated RBs both from DL and UL simultaneously in the above formulation. While this can be realized in TDD systems, this would pose a problem for half-duplex clients in FDD systems. Hence, FDD systems would have an additional constraint to ensure that a D2D user be allocated RBs either from DL or UL but not from both.

Reuse across co-located sectors: The rate of a user (k_j) in a sector (j) on a RB (n) (denoted by: $r_{k_j,n}$) depends not only on its individual rate in its sector in the absence of interference ($r'_{k_j,n}$) but also on the other potentially interfering traffic that is co-scheduled on the same RB (due to overlapping dynamic FFR bands) in the other co-located sectors (ℓ and m). While there would be no interference if all traffic were cellular (due to sectorization), the presence of omni-directional D2D traffic could create interference. To estimate the true impact of interference on rate, one would need the channel gain information between all users in all the co-located sectors, which is not feasible to obtain for per-frame scheduling (not to mention the associated overhead). Hence, ***R2D2*** adopts a conservative approach in identifying users from co-located sectors that can be co-scheduled on the same RB without any interference using information that is already readily available. Note that every user keeps track of the reference signal receiver power (RSRP) from its neighboring cells/sectors. Using RSRP, a user can determine which of its co-located sectors it is closer to, thereby localizing [60] it to a particular half of its sector (Fig. 5.4). Now to determine if a user can be co-scheduled on an RB with another user, the receiver of the first link and the transmitter (interferer) of the second link are considered. If both of them are located in opposite halves (e.g. region A and D in Fig. 5.4) and are both not interior traffic, then they can be co-scheduled, i.e. $r_{k_j,n} = r'_{k_j,n}$ and 0 otherwise. Note that if either of the two users under consideration is a cellular BS, then there is no interference due to sectorization. When a user is co-scheduled with two other users, the same check can be employed with each of the interfering users independently. Hence, we have $r_{k_j,n} = \{r'_{k_j,n}, 0\}$ depending on if co-scheduled users, *i.e.* k_ℓ and k_m conflict with k_j.

Thus, we see that the *above formulation not only captures the scheduling of cellular and D2D traffic jointly across DL and UL resources but also across co-located sectors (leveraging reuse) in a scalable manner without incurring additional overhead.*

Hardness of the Problem: Based on the reduction of 3-bounded 3-dimensional matching problem to a simpler version of (5.10), we show in appendix:

Theorem 5.4.1. *Maximizing* (5.10) *is a NP-Hard problem and admits no* $(1-\delta)$*-approximation algorithm unless P=NP.*

Proposed Algorithms: We propose three polynomial-time algorithms with performance guarantees. The first algorithm, *Alg1* has a complexity of $O(N^2K^3)$ and an approximation ratio of 1/2; while the second algorithm *Alg2* has a lower complexity of $O(N^2K)$ to aid in low-latency implementations at the BS (to meet 1ms LTE frame timing) and an approximation ratio of 1/4. Both these algorithms yield close-to-optimal performance in practical evaluations (Section 5.5) and apply to TDD systems. For FDD systems with the additional constraint that a D2D user can be allocated resources in either DL or UL but not both, we provide an alternate algorithm *Alg3* with a different approach that incurs a complexity of $O(N^4K^3)$ and provides an approximation ratio of 1/3.

The proofs for the complexity and approximation ratios for all the three algorithms are included in the appendix.

Alg1 *Alg1* is a greedy algorithm that requires two inputs: (i) Set of users $\mathcal{K}$ for which the schedule needs to be computed; and, (ii) The set of RBs that can be allocated to each of the user (F_{k_j} for user k_j). To compute the schedule, the BS invokes *Alg1* at the beginning of for every frame by setting $\mathcal{K}$ to the set of users across the co-located sectors: $\mathcal{K} = \mathcal{K}_j \cup \mathcal{K}_\ell \cup \mathcal{K}_m$. For user k_j, F_{k_j} is computed as follows:

$$
\begin{aligned}
F_{k_j} &= F^j_{0,z} \cup F_{j,z} \text{ ; if } k_j \in \mathcal{C}_{j,z} \wedge k_j \text{ is interior}, z \in \{u, d\} \\
&= F^j_{0,d} \cup F_{j,d} \cup F^j_{0,u} \cup F_{j,u} \text{ ; if } k_j \in \mathcal{D}_j \wedge k_j \text{ is interior} \\
&= F_{j,z} \text{ ; if } k_j \in \mathcal{C}_{j,z} \wedge k_j \text{ is exterior}, z \in \{u, d\} \\
&= F_{j,d} \cup F_{j,u} \text{ ; if } k_j \in \mathcal{D}_j \wedge k_j \text{ is exterior}
\end{aligned}
$$

This allows D2D to be scheduled on both DL and UL resources and allows interior traffic to use exterior resources. *Alg1* works greedily (Line 2-9 of Algorithm 4) and in each iteration, adds that tuple of (n, k_j, k_ℓ, k_m) to the schedule S that maximizes the incremental utility (Line 6-8, $f(n, k_j, k_\ell, k_m)$). Recall (from Section 5.3) that there is negligible benefit to reusing cellular resources by D2D traffic within a sector. Hence, *Alg1* considers only those tuples, where at most three users k_j, k_ℓ, k_m, one from each of the co-located sectors, are scheduled on the same RB (Line 3). This would include tuples with less than three users as well, as these may provide higher utility on a RB depending on the interference conflicts between the co-located sectors on that RB. For feasibility purposes, among all tuples, only those tuples are considered where it is possible to schedule all the users present in the tuple on the associated RB (See Line 4). Further, to ensure that the computed schedule is feasible, before finally adding a tuple to the schedule S, *Alg1* verifies (Line 5) that the updated schedule does not violate any of the constraints specified in (5.10). Among all the valid tuples (set $\mathbf{T_3}$), the tuple that maximizes the incremental utility is added to S (Line 6-8). *Alg1* returns (Line 9) when it can no longer find a valid tuple that can be added to the schedule.

Alg2 *Alg2* is a lower complexity algorithm that decouples the co-located sectors during RB allocation. Similar to *Alg1*, *Alg2* is invoked by the base station by providing two inputs (set of all users across three sectors and the set of RBs for each user). *Alg2* works by greedily (Line 2-7 of Algorithm 5) adding *valid* (Line 3) tuples of (n, k_i) to the schedule

Algorithm 4: *Alg1*: Computes the assignment of resources to users that maximizes (5.10) in a TDD system.

Input : Set of users $\mathcal{K}$, $F_{k_i} \forall k_i \in \mathcal{K}$

$S \leftarrow \{\}$

while *true* **do**

  $\mathbf{T_1} \leftarrow (n, k_j, k_\ell, k_m) : k_i \in (\mathcal{K}_i \cap \mathcal{K}) \cup \{\phi\}\ \forall k_i \in \{k_j, k_\ell, k_m\}$

  $\mathbf{T_2} \leftarrow (n, k_j, k_\ell, k_m) : (n, k_j, k_\ell, k_m) \in \mathbf{T_1} \wedge (k_j = \phi \,||\, n \in F_{k_j}) \wedge (k_\ell = \phi \,||\, n \in F_{k_\ell}) \wedge (k_m = \phi \,||\, n \in F_{k_m})$

  $\mathbf{T_3} \leftarrow (n, k_j, k_\ell, k_m) : (n, k_j, k_\ell, k_m) \in \mathbf{T_2} \wedge$ Adding (n, k_j, k_ℓ, k_m) to S does not violate any constraints

  $(n^*, k_j^*, k_\ell^*, k_m^*) \leftarrow \text{argmax}_{(n,k_j,k_\ell,k_m)\in\mathbf{T_3}} f(n, k_j, k_\ell, k_m)$

  if $n^* \neq \phi$ **then**

    $S \leftarrow S \cup (n^*, k_j^*, k_\ell^*, k_m^*)$

  else return S

that maximize the incremental utility, *i.e.* it allocates an RB to only one user in one of the co-located sectors at a time, although all users from the co-located sectors are considered during the decision process. Further, a new user can also be scheduled on an RB that is already allocated to user(s) from other sectors, subject to interference conflicts and the resulting additional utility the new user would bring to the given RB. This in turn would allow for reuse of RBs across co-located sectors.

Theorem 5.4.2. *Alg2 has a complexity of $O(N^2K)$ and guarantees an approximation ratio 1/4.*

Alg3 *Alg3* is tailored for FDD systems but can also be applied to TDD systems. In a FDD system, a D2D transmission can either use the UL RBs or the DL RBs (but not both).

Algorithm 5: *Alg2*: Computes the assignment of resources to users that maximizes (5.10) in a TDD system.

Input : Set of users $\mathcal{K}$, $F_{k_i} \forall k_i \in \mathcal{K}$

1 $S \leftarrow \{\}$

2 **while** *true* **do**

3 $\quad \mathbf{T_1} \leftarrow (n, k_i) : n \in F_{k_i} \wedge k_i \in \mathcal{K} \wedge$ Adding (n, k_i) to S does not violate any constraints

4 $\quad (n^*, k_i^*, k_\ell^*, k_m^*) \leftarrow \text{argmax}_{(n,k_i,k_\ell,k_m):(n,k_i)\in\mathbf{T_1}\wedge(n,k_\ell)\in S\wedge(n,k_m)\in S} \leftarrow f(n, k_i, k_\ell, k_m) - f(n, \phi, k_\ell, k_m)$

5 $\quad$ **if** $n^* \neq \phi \wedge (f(n, k_i^*, k_\ell^*, k_m^*) - f(n, \phi, k_\ell^*, k_m^*)) > 0$ **then**

6 $\quad\quad S \leftarrow S \cup (n^*, k_i^*)$

7 $\quad$ **else return** S

This additional constraint coupled with finite user buffers makes the problem even more challenging and warrants a new approach to ensure performance guarantees. Unlike *Alg1* and *Alg2* that allocate at the granularity of RBs, *Alg3* instead makes allocations at the granularity of users, *i.e.* allocation of RBs to a user is executed in a single step before moving to another user. *Alg3* first determines the set of RBs that can be allocated to user k_i. Here, $\mathcal{K}'$ denotes the set of users for which the set of RBs that can be allocated to them has not been determined yet (Line 1 of Algorithm 6). While cellular DL (UL) users are restricted to the set of DL (UL) RBs (Line 6-11), D2D users have the flexibility of being allocated either from the set of DL or UL RBs (but not both, Line 3-5). In every iteration of the while loop (Line 12-14), *Alg3* determines the set of RBs that can be allocated for exactly one user. This is done by greedily selecting (Line 13) the tuple (k_i, F_{k_i}) that when added to the possible schedule maximizes the value of the incremental utility. Computing the incremental utility of adding a user (along with its set of potential RBs for allocation)

to the current schedule, in turn corresponds to the scheduling problem (5.10) considered in TDD systems. Hence, *Alg1* is invoked (Line 13) to compute the incremental utility of adding a tuple to the schedule. *Alg3* then determines the user and its set of RBs (whether DL or UL RBs for D2D users) that provides the maximum incremental utility and adds it to the current schedule (Line 14). Note that at each iteration only a user and its operational set (DL or UL RBs if D2D user) are determined and remain fixed as the schedule evolves. The specific RBs themselves that are allocated to existing users in the schedule may change as new users are added to the schedule, due to the re-computation of a TDD schedule in each iteration.

5.5 Evaluation

5.5.1 Setup

A frame-level simulator written in C++ is considered for evaluation of the proposed algorithms. In our evaluation, we set up 19 base stations each with 3 sectors. Further, each sector has number of users varying between 50-150 and they are placed at random location within the sector. Each user generates a request to a randomly selected file server for a file of size 500 KB using an exponential distribution. The default percentage of UL transmissions in each sector was 30%. Further, 30% of the generated requests are D2D, implying that the destination file server is another user in the same sector. For all algorithms, number of RBs available for UL and DL communication were 9 and 21, respectively. Every data session (user) has a finite amount (125 KB) of data (buffer). The Okumara-Hata urban path loss model is employed along with log-normal shadowing and fast (Rayleigh) fading. The doppler fading for each user's Rayleigh channel is equivalent to a velocity of 3-10 Km/hour. The SNR from the model is mapped to a specific data rate [9]. The cells are deployed in

the hexagonal fashion with a pre-determined sector radius (1000m, by default). Each data point presented here is an average over results from 10 randomly generated topologies.

We restrict our evaluations to TDD systems here (since inferences for FDD are similar). Apart from the two flavors of *R2D2* (D2D-aware dynamic FFR with *Alg1* or *Alg2*), we implemented multiple other algorithms for comparison. In all these algorithms (except StaticPowerReduce), at epoch level time granularity, dynamic FFR scheme is executed for determining the FFR bands (as explained in Section 5.4.3). However, the algorithms behave differently at the frame level time granularity as follows. *a*) **Alg1NoD2D:** Every BS computes the schedule using *Alg1*. Since, this algorithm classifies all traffic as cellular, there is no conflict among transmissions scheduled by co-located sectors. *b*) **D2DDynamicGenie:** Each BS computes the joint schedule for all the three co-located sectors. Further, D2DDynamicGenie is aware of all the path loss values (including, path loss between users), and can potentially leverage that information to schedule multiple transmissions in the same sector on the same RB. It also dynamically controls the transmission power level of D2D transmissions to maximize the reuse of RBs. *c*) **StaticPowerReduce** proposed in [39] works by reducing the transmission power level of all D2D links. It is expected that this reduction will prevent D2D from interfering with UL transmissions, allowing them to be co-scheduled on the same RBs in the same sector. This algorithm does not allow outdoor D2D traffic to coexist with cellular DL traffic (to avoid interference to the cellular users operating on the same RB in the same sector). We extend [39] to multicell deployments through a static FFR scheme that allocates equal number of RBs to all sectors. We evaluate the scheduling algorithms at per-frame granularity based on their objective function (weighted sum rate), which in turn corresponds to throughput with equal user weights.

5.5.2 Results

Variation in D2D traffic load: With increase in D2D traffic, the throughput for both *Alg1* and *Alg2* increases (See Fig. 5.5a). This is primarily because of *offloading* benefit from D2D. Further, D2D is also able to leverage the disparity in UL-DL traffic distribution and thus, can be scheduled on the RBs that would have otherwise gone wasted. Fig. 5.5a shows that *Alg2* gives a throughput of 3.79x and 3.00x compared to StaticPowerReduce [39] and Alg1NoD2D, respectively. Although, *Alg2* has lower complexity than *Alg1*, we observe that on an average, throughput of *Alg2* is within 3.18% of the throughput of *Alg1*, making it an ideal candidate for implementation. This is because when *Alg2* adds a single user to the schedule, it also considers the interference that may arise between this user and the other users that are already scheduled on the same RB.

From the result, we can also see that *Alg2* has throughput within 4.8% of D2DDynamicGenie. This implies that even with all the information about path loss values between users, D2DDynamicGenie is unable to schedule more transmissions than *Alg2*. Dynamic control of transmission power level is also not helpful since D2D transmitters need to transmit at low power to avoid interference to other transmissions in the same sector. This coupled with the interference faced by the D2D receiver, reduces the rate that can be supported on the D2D link. This confirms that D2D provides low reuse benefit in multicell scenarios with FFR. Although, StaticPowerReduce allows D2D transmissions, reducing the transmission power level of D2D transmissions also reduces D2D offload benefit. Further, even after reducing the power levels, D2D transmissions may still interfere with the UL traffic and vice versa. From our evaluations, we observe that at 30% D2D traffic load, 25% of the transmissions scheduled by StaticPowerReduce were unsuccessful due to interference.

Variation in sector radius: For larger sectors, all algorithms have lower throughput since the increase in distance results in lower physical layer data rates. When sector radius is less than 1000m, throughput of *Alg2* is within 4.3% of D2DDynamicGenie. However, at

sector radius of 2000m, the gap between them is higher at 12.3%, but still within reasonable limits. This is because in larger cells, D2DDynamicGenie is able to leverage the path loss information to schedule multiple transmissions within the same sector.

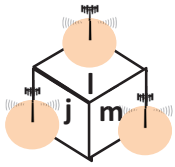
Variation in DL-UL traffic ratio: Fig. 5.5c shows the variation in throughput with variation in the percentage of UL traffic across different sectors. Here, an x-axis value of 30-60% implies that the UL traffic percentage of all sectors was uniformly distributed between 30% and 60%. For StaticPowerReduce and Alg1NoD2D, the total throughput peaks when the average offered UL traffic (30%) is close to the percentage RBs allocated for UL traffic (30%). The maximum throughput of *Alg2* stays almost constant with variation in UL traffic. On the other hand, for Alg1NoD2D and StaticPowerReduce, their maximum throughput is 85% and 142% more than their minimum throughput. This clearly implies that D2D-aware dynamic FFR algorithm of *R2D2* is able to leverage the flexible nature of D2D traffic and is able to achieve high throughput even when disparity is high.

Variation in hotspot D2D traffic: We model the D2D traffic distribution as "hotspot traffic" such that for $h\%$ of sectors, 80% of their traffic is D2D, while the remaining sectors have only 20% of their traffic as D2D (to emulate scenarios of stadiums, event centers, etc.). Fig. 5.5d shows that *Alg2* provides 2.33x and 3.45x throughput compared to StaticPowerReduce [39] and Alg1NoD2D, respectively. The gains here are higher compared to when D2D traffic is uniformly distributed since here the dynamic FFR scheme of *R2D2* is able to better leverage the variation in D2D traffic across sectors.

5.6 Conclusions

In this chapter, we showed that D2D traffic does not provide significant reuse benefit in multicell networks that employ FFR schemes. To best leverage the D2D in such networks, we proposed that D2D should be used as a flexible traffic so that it can be used on resources which would have otherwise been wasted due the disparity between uplink and downlink

traffic across sectors. To that end, we proposed a novel D2D aware dynamic FFR algorithm that is executed at epoch level granularity and is scalable. We also proposed provable approximate scheduling algorithms that maximize the benefit from D2D. Through simulations, we showed that R2D2 improves performance by 2.79x compared to existing D2D algorithms.



(a) Cross-Sector Clique.

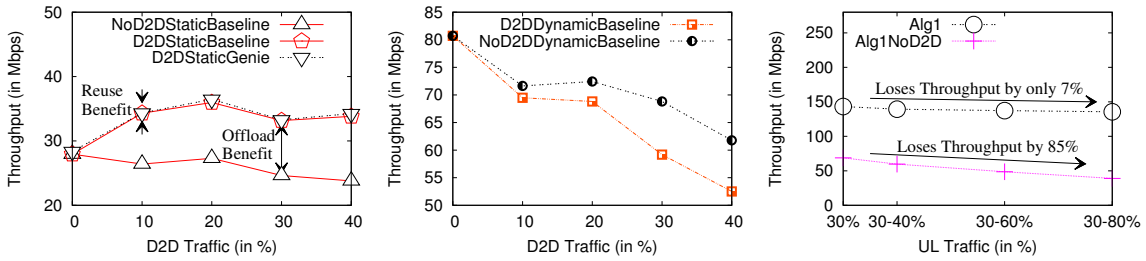
Cell\Demand	DL	UL	D2D
Cell *j*	1	2	4
Cell *l*	3	3	9
Cell *m*	5	1	8

(b) Traffic demand at each sector (or cell).

	D2D Oblivious FFR Allocation					D2D Aware FFR Allocation (R2D2)				
Cell	Total Demand		Resources Allocated		Traffic Supported	Total Demand		Resources Allocated		Traffic Supported
	DL	UL	DL	UL		DL	UL	DL	UL	
j	1	2 + 4 = 6	3	3	1 + 3 = 4	1 + 1 = 2	2 + 3 = 5	2	5	2 + 5 = 7
l	3	3 + 9 = 12	8	5	3 + 5 = 8	3 + 6 = 9	3 + 3 = 6	9	6	9 + 6 = 15
m	5	1 + 8 = 9	13	4	5 + 4 = 9	5 + 8 = 13	1 + 0 = 1	13	1	13 + 1 = 14
Total			24	12	**21**			24	12	**36**

(c) The number of resources that can be allocated to UL and DL are 12 and 24, respectively. R2D2 will carefully split D2D traffic across UL and DL, serving **71% higher traffic.**

Figure 5.2: A D2D Oblivious dynamic FFR algorithm will put all D2D traffic on UL resources (to avoid interference with the downlink cellular transmissions in the co-located sectors) and allocate resources proportionally. On the other hand, a dynamic FFR allocation scheme aware of D2D traffic (R2D2) will carefully split D2D traffic across UL and DL. When coupled with resource allocation for interior traffic, the D2D-Aware dynamic FFR allocation becomes more challenging (shown in Sec. 5.4.3) and provides even higher benefits.



(a) Using D2D with static FFR. (b) Using D2D with dynamic FFR. (c) Asymmetric UL-DL traffic.

Figure 5.3: Evaluation results illustrating challenges in leveraging D2D.

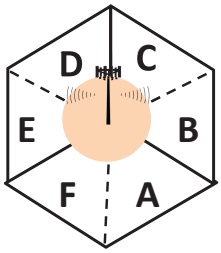
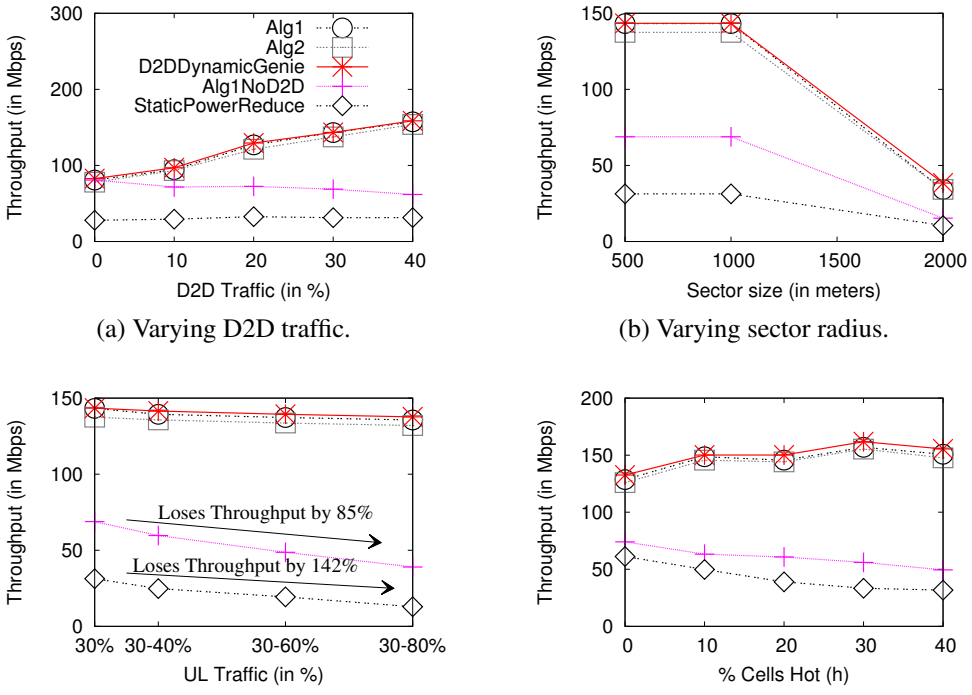


Figure 5.4: Constraints when scheduling transmissions on the same RB.



(a) Varying D2D traffic.

(b) Varying sector radius.

(c) Varying UL traffic ratio.

(d) Varying number of hot sectors (h).

Figure 5.5: Evaluation results illustrating challenges in leveraging D2D.

Algorithm 6: *Alg3*: Computes the assignment of resources to users that maximizes (5.10) in a FDD system.

Input : Set of users $\mathcal{K}$

$\mathcal{K}' \leftarrow \phi, S \leftarrow \phi$

forall the $k_i \in \mathcal{K}$ **do**

  if $k_i \in \mathcal{D}_j \cup \mathcal{D}_\ell \cup \mathcal{D}_m$ **then**

    if k_i is exterior **then** $F_{k_i}[0] \leftarrow F_{i,d}$, $F_{k_i}[1] \leftarrow F_{i,u}$

    else $F_{k_i}[0] \leftarrow F^i_{0,d} \cup F_{i,d}$, $F_{k_i}[1] \leftarrow F^i_{0,u} \cup F_{i,u}$

  else if $k_i \in \mathcal{C}_{j,d} \cup \mathcal{C}_{\ell,d} \cup \mathcal{C}_{m,d}$ **then**

    if k_i is exterior **then** $F_{k_i}[0] \leftarrow F_{i,d}, F_{k_i}[1] \leftarrow \phi$

    else $F_{k_i}[0] \leftarrow F^i_{0,d} \cup F_{i,d}, F_{k_i}[1] \leftarrow \phi$

  else if $k_i \in \mathcal{C}_{j,u} \cup \mathcal{C}_{\ell,u} \cup \mathcal{C}_{m,u}$ **then**

    if k_i is exterior **then** $F_{k_i}[0] \leftarrow F_{i,u}, F_{k_i}[1] \leftarrow \phi$

    else $F_{k_i}[0] \leftarrow F^i_{0,u} \cup F_{i,u}, F_{k_i}[1] \leftarrow \phi$

while $\mathcal{K} \neq \mathcal{K}'$ **do**

  $(k^*, F^*_{k^*}) \leftarrow \text{argmax}_{(k_i, F_{k_i}): k_i \in \mathcal{K} \setminus \mathcal{K}' \wedge F_{k_i} \in \{F_{k_i}[0], F_{k_i}[1]\}} f(\textit{Alg1}(\mathcal{K}' \cup \{k_i\}, F_{k_j} \forall k_j \in \mathcal{K}' \cup \{k_i\})) - f(\textit{Alg1}(\mathcal{K}', F_{k_j} \forall k_j \in \mathcal{K}'))$

  $\mathcal{K}' \leftarrow \mathcal{K}' \cup \{k^*\}, F_{k^*} \leftarrow F^*_{k^*}$

Call *Alg1* $(\mathcal{K}', F_{k_j} \forall k_j \in \mathcal{K}')$

return schedule computed by *Alg1*

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The recent increase in the number of mobile devices and the popularity of mobile apps has led to a huge increase in the mobile wireless traffic. Channel frequencies being a natural resource are limited in number. Although many MAC layer algorithms have been proposed that maximize the usage of a channel frequency, all these algorithms prohibit nearby devices to transmit or receive data simultaneously.

The focus of this thesis is to propose solutions that make best use of given wireless resources by leveraging the high density of wireless devices. We proposed different solutions to improve the wireless throughput in Wi-Fi and cellular networks, where each solution is suitable for certain category of deployments. We also demonstrated the efficacy of the proposed protocols by implementing them on our testbed.

Traditionally, the network administrators have deployed high density of APs to ensure complete wireless coverage. This thesis demonstrates that it is possible to leverage this high density of wireless devices to achieve high throughput, whereas in such networks the traditional 802.11 algorithm may fail due to high collision rates. For non-cooperating wireless networks, we proposed Mozart that uses a novel successive packet decoding mechanism to solve both the hidden terminal and exposed terminal problems. Further, we also showed that by leveraging cooperation among APs and the underutilized wired backbone, it is possible to significantly improve the performance of the mobile wireless networks.

The solutions presented in this dissertation needs further investigation and analysis. Next, we present some directions for future research on these problems:

- In the future, the number of mobile devices is expected to increase significantly. This thesis takes the first few steps in discussing how the more powerful wireless devices such as APs can cooperate to handle the interference generated by mobile devices. It remains to be seen how the interference generated in highly dense deployments of mobile devices can be handled using techniques discussed in this thesis.

- This thesis proposed Symphony and RobinHood, two solutions for Wi-Fi networks where the access points cooperate. It needs to be explored if the techniques proposed in these solutions will also be applicable to cellular networks. In cellular networks, the distance from the wireless devices to the base stations is much higher compared to distance in the Wi-Fi networks. Further, in cellular networks, the latency between base stations is higher compared to the latency between access points in Wi-Fi networks. The longer distance and the higher latency presents new challenges. It is possible that modifications may be required to Symphony and RobinHood in order to make them suitable for cellular networks.

- Wireless devices have limited battery capacity. It is important to design wireless solutions that minimize the energy consumption of these devices. RobinHood requires wireless devices to transmit only once while the APs transmit in the second slot. Thus, RobinHood minimizes the energy consumption of wireless devices. On the other hand, Mozart and Symphony require clients to transmit multiple times. This may increase the energy consumption of the devices. It needs to be investigated how the energy consumption of these solutions can be further reduced while minimally affecting the throughput.

- In Chapter 5, we studied device to device communication from one client to another.

Another application of D2D communication is multicasting data from a single source to multiple clients over multiple hops. This class of applications are expected to be deployed in shopping malls (for advertising) and large stadiums (broadcasting replays) etc. It needs to be investigated how D2D multihop traffic can be integrated in the existing cellular networks.

APPENDIX A:

A.1 Mozart

A.1.1 Critical Period

In this section, we compute the critical period of Mozart. Using a simpler proof, we first show that critical period of Mozart is shorter than 18.7 μs. This duration of 18.7 μs is still significantly shorter than existing techniques: 39.4μs (802.11ec [55]), 152 μs (802.11 with rts,cts).

Lets say a receiver r_a starts transmitting a poll at t=0 (See Figure A.1). Then, this poll would be received by all its neighboring nodes within 6.35+1+2= 9.35 μs where 6.35μs is the duration of poll, 1 μs for propagation delay and 2 μs for the turn-around time. So, if all neighboring nodes do not participate in any new exchange (i.e., keep their radio in the receive mode) for these 9.35 μs, then they would hear r_a's poll and would not transmit unless they have packets that are intended for r_a. Thus, all such nodes would transmit simultaneously to r_a and r_a would be able to decode them by carefully using suppress. Thus, r_a would be able to decode all the received data packets since it won't hear any spurious (unintended) transmissions. Therefore, the upper bound on the critical period can be computed by taking twice of 9.35 μs (duration before and after the transmit of poll). This shows that the critical period of Mozart is upper bounded by 18.7 μs.

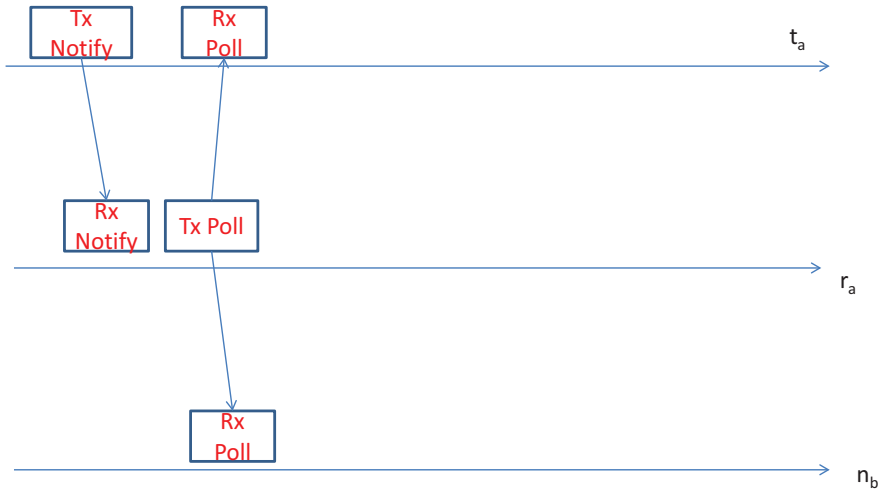


Figure A.1: Critical Period Computation. If all radios do not initiate a new exchange within 9.35 μs, then they would hear the poll.

A.1.2 Critical Period for other protocols

802.11ec [55] In 802.11ec, lets consider the network with one receiver (say r) and 2 hidden transmitters (say t_a and t_b). If t_a sends I_c at t = 0, then the receiver will start transmitting R_c at t=10.35μs (See Figure 2 in [55]). This R_c would be heard by all nodes by t=17.70. Thus, if t_b starts changing its radio from rx to tx state just before t=17.70μs, then it would not hear R_c and would start transmitting I_c at t=19.70. This, I_c would corrupt the transmission of t_a at the receiver. In other words, if t_b starts its exchange within 19.70 μs of t_a, then both the exchanges would be corrupted. Thus, the critical period for 802.11ec is 39.40μs (twice of 19.70 μs). Similarly, it can be shown that the critical period of 802.11 is 152 μs (See [55] for details).

A.2 Symphony

Theorem A.2.1. *Non Flex TDMA is NP Hard: Given an infrastructure network N, then computing a schedule of length k slots that allows all clients to transmit exactly 1 packet to their associated APs is NP-Hard.*

Proof. Proof is similar to [23]. We reduce the k-coloring problem to k-non-flex scheduling. Given a graph $G = (V, E)$, we construct an infrastructure-based wireless network (say N). For each vertex v_i in V, a client node n_i is placed in the network N. Further, for every client node n_i in N, an AP (say $a(n_i)$) is also placed with in the transmission range of n_i.

Further, if two vertices v_i and v_j are connected by an edge in G, then N is modified such that: (i) $a(n_i)$ is in interference range of n_j but outside it's transmission range; and, (ii) $a(n_j)$ is in interference range of n_i but outside it's transmission range.

We now show that if G is $k-$colorable, then it is possible to schedule all the client nodes in N in k slots. First observe that a transmission from n_i can interfere at $a(n_j)$ *iff* v_i and v_j share an edge in G. Therefore, we can say: (i) If two nodes can be assigned the same color in G, then the corresponding nodes can transmit simultaneously to their AP in N; and, (ii) If two nodes can transmit simultaneously to their associated APs in N, then the corresponding vertices can be assigned the same color in G.

Hence, if G has a k-coloring, then all the client nodes can transmit in k slots in N. Similarly, if all the client nodes can transmit in k slots in N, then G has a k-coloring. □

Theorem A.2.2. *Flex TDMA is NP Hard: Given an infrastructure network N, then computing a schedule of length k slots that allows all clients to transmit at least one packet to any of the neighboring APs is NP-Hard.*

Proof. With Flex TDMA, each client can send its uplink data to any AP that is in its transmission range. We use the same construction as used before in Theorem A.2.1. Observe that in that construction of N, every client had only one AP in its transmission range.

So, any k length schedule for Flex-TDMA MAC protocol would also be a valid k length schedule for Non-Flex TDMA protocol. Thus, if Non-Flex TDMA scheduling problem is NP-Hard, then Flex-TDMA scheduling problem must also be NP-Hard. □

Theorem A.2.3. *Symphony scheduling is NP-Hard: Given an infrastructure network N, then computing a schedule of length k slots that allows all clients to transmit at least one packet to any of the neighboring APs such that APs are also allowed to exchange packets on the backbone is NP-Hard.*

Proof. We use the same G to N construction as used in the proof of Theorem A.2.1. Recall that in that construction of N, every client had only one AP in its transmission range. Further, for any two client nodes n_i and n_j, if $a(n_i)$ is in the interference range of n_j, then $a(n_j)$ is also in the interference range of n_i. Thus, if links $n_i \rightarrow a(n_i)$ and link $n_j \rightarrow a(n_j)$ are included in the dependence graph, then they will form a directed cycle. Thus, for such a network Symphony would not be able to harness the backbone since if two neighboring clients are picked in the subgraph, then a cycle is created. So, any k length schedule for Flex-TDMA MAC protocol would also be a valid k length schedule for Symphony protocol. Thus, if Non-Flex TDMA scheduling problem is NP-Hard, then Symphony scheduling problem must also be NP-Hard. □

Theorem A.2.4. *A set of client-AP transmissions can be decoded if and only if the vertex induced subgraph (say G_s) of the corresponding set of vertices in G_d is a DAG (Directed Acyclic Graph).*

Proof. We first prove that if G_s is acyclic, then the corresponding client-AP transmissions can be decoded. This proof is by construction. Since, G_s is ayclic, then there must exist a topological sorting of G_s. To decode all client-AP transmissions, the APs can decode them in the order of topological sort which has the property that any transmission be decoded is

only dependent on transmissions preceding it (in the order of the topological sort). Thus, all the transmissions would be successfully decoded.

Next, we prove that if G_s is cyclic, then the corresponding client-AP transmissions can't be decoded. By contradiction, assume that there exists some order (say O) of decoding that allows the CS to decode all the client-AP transmissions. Assume $n_i \rightarrow AP_j$ and $n_k \rightarrow AP_l$ are two transmissions in this order such that $n_i \rightarrow AP_j$ precedes $n_k \rightarrow AP_l$. Then in G_d there can not exists an edge from $n_k \rightarrow AP_l$ to $n_i \rightarrow AP_j$ since presence of such an edge would make decoding of $n_i \rightarrow AP_j$ impossible without having decoded $n_k \rightarrow AP_l$. Thus, the vertices in O must form a topological sort with no edges from later vertices to preceding vertices. This implies that G_s must be acyclic since only acyclic graphs have a valid topological sort. This contradicts the fact that G_s is cyclic. Thus, our assumption must be wrong and it must be impossible to decode all client-AP transmissions.

We proved that if G_s is acyclic, then the corresponding client-AP transmissions can be decoded. We also proved that if G_s is cyclic, then the corresponding client-AP transmissions can't be decoded. Combining the two results proves the given theorem. □

A.3 RobinHood

A.3.1 Algorithm *Satisfiable*

As discussed before in Section 3.5.2, Algorithm *Satisfiable* determines if a given schedule is satisfiable or not. Without loss of generality, let S be the schedule such that $S = \{(C_i, AP_i) : AP_i$ is the receiving AP for packet x_i, and x_i is the i^{th} packet to be decoded and $1 \leq i \leq N\}$. Then, we draw an undirected graph $\mathcal{G}$ where we have two sets of vertices: (i) $\mathcal{V}_1$: Each vertex in $\mathcal{V}_1$ corresponds to a pair (C_i, AP_i) such that $(C_i, AP_i) \in S$; and, (ii) $\mathcal{V}_2$: Let $\mathcal{A}$ be the set of all APs in the group. Each vertex in $\mathcal{V}_2$ corresponds to an AP (say AP_j) in $\mathcal{A}$ such that AP_j is not in S.

Next, we draw an edge from vertex $V_i \in \mathcal{V}_1$ to a vertex $V_j \in \mathcal{V}_2$ if and only if AP_j can hear from C_i. We set the capacity of all these edges to 1.

Next, we construct a source vertex (say V_s) and draw edges from V_s to every vertex, say $V_i \in \mathcal{V}_1$. Further, we set the capacity of edge from V_s to V_i as $i-1$ where $1 \leq i \leq N-1$. We also set the capacity of edge from V_s to V_N as $N-2$. Finally, we also construct a termination (or sink) vertex (say V_t). Then, we add edges from each vertex $V_j \in \mathcal{V}_2$ to V_t and set the capacity of these edges to 1.

We solve the Maximum Flow problem on $\mathcal{G}$ from vertex V_s to V_t. We say that the given schedule S is satisfiable only if the flow value is at least $\frac{N^2-N-2}{2}$ where N is the length of the schedule. Next, we prove the correctness of this reduction.

Theorem A.3.1. *If $\mathcal{G}$ has the desired max flow, then S is satisfiable.*

Proof. $\mathcal{G}$ can have a flow of at least $\frac{N^2-N-2}{2}$ only if the following two conditions are satisfied: (i) From every vertex (C_i, AP_i), there is an outgoing flow of $i-1$ when $1 \leq i \leq N-1$; and, (ii) Vertex (C_N, AP_N), there is an outgoing flow of $N-2$. This implies that for every client C_i, there are at least $i-1$ APs that can hear it when $1 \leq i \leq N-1$. This also implies that for client C_N, there are at least $N-2$ APs that can hear it. These two conditions ensure that S must be satisfiable. □

Theorem A.3.2. *If S is satisfiable, then $\mathcal{G}$ has the desired max flow.*

Proof. If S is satisfiable, then for every client C_i in S, there must be at least $i-1$ unique APs that can hear it when $1 \leq i \leq N-1$. This also implies that for client C_N, there are at least $N-2$ unique APs that can hear it. These two conditions ensure that the max flow in the graph $\mathcal{G}$ should be at least $1+2+\ldots+(N-2)$ from the clients from C_1 to C_{N-1} and a flow of $N-2$ for the C_N. Thus, the total flow should be at $\frac{N^2-N-2}{2}$. □

A.4 R2D2

Proof of Theorem 5.4.1 To show that maximizing (5.10) is NP-Hard, we show that a special variant of (5.10) is NP Hard. In the special variant of (5.10), for a given base station, all its users are present in only one of the three sectors. Further, the D2D traffic demand is 0 and the objective is to maximize the throughput of the cellular users given finite buffer traffic for each user. It has been shown [7] that this simplified problem is NP-Hard and it can't be approximated within $(1-\delta)$ of the optimal (for some δ).

Theorem A.4.1. *Alg1 has a complexity of $O(N^2K^3)$ and guarantees an approximation ratio of $\frac{1}{2}$ compared to exponential time optimal algorithm.*

Proof. Since for a given base station, at most one tuple can be assigned to a RB, therefore, the for loop in Line 4 of *Alg1* will be invoked at most N times. Further, within the for loop, there are at most $O(NK^3)$ tuples that need to be evaluated (Line 5-9) . Evaluating a tuple requires validating the buffer constraints of at most three users takes $O(1)$ time since the number of users in the tuple is at most 3. Thus, the complexity of *Alg1* is $O(N^2K^3)$.

We compare the schedule (S^1) computed by *Alg1* with the schedule computed by optimal (S^*) where both S^1 and S^* are set of tuples ((n, k_j, k_ℓ, k_m)). The tuples in S^1 are arranged in the order of their throughputs (value of function f). During this comparison, we maintain the following four invariants:

- Throughput of tuples removed from S^1 is at least half the throughput of tuples removed from S^*.
- $B^1_{k_i} \geq B^*_{k_i} \forall k_i : k_i \in S^*$.
- *Alg1* is free to choose any tuples that appear in S^* without violating any constraints in (5.10).
- If S^1 has no tuple that is using RB n, then S^* has no tuple such tuple either.

Initialization: Initially, no tuples have been removed from S^1 or S^*, so the throughput of tuples removed from both S^1 and S^* are 0. Hence, the first invariant holds. Secondly, all the users have the same buffer sizes before any of the scheduling algorithm has executed. Thus, $B^1_{k_i} = B^*_{k_i} \forall k$. This satisfies the second invariant. Thirdly, since S^* is a valid schedule and all users have the same buffer size in both the schedules (invariant 2), therefore, *Alg1* is free to choose any tuples from S^* without violating constraints. Fourthly, both S^1 and S^* will schedule at least one user on each RB. Thus, the fourth invariant is true.

To show that S^1 has at least half the throughput compared to S^*, we start removing one tuple each from S^1 and S^* without violating the invariants. Let (n, k_j, k_ℓ, k_m) be the tuple with maximum value in S^1 and (n, k'_j, k'_ℓ, k'_m) be the tuple chosen by S^* for RB n. Clearly, $f(n, k_j, k_\ell, k_m) \geq f(n, k'_j, k'_\ell, k'_m)$ because otherwise the greedy algorithm would have chosen (n, k'_j, k'_ℓ, k'_m) (follows from invariant 3). We remove $f(n, k_j, k_\ell, k_m)$ from S^1 and $f(n, k'_j, k'_\ell, k'_m)$ from S^*. Apart from this, for every user $k_i \in \{k_j, k_\ell, k_m\}$, let b_{k_i} be the number of bits that k_i was able to transmit/receive when scheduled on RB n by S^1. From the remaining tuples in S^*, we reduce the bits transmitted/received by k_i by b_{k_i}.

Therefore, the tuple removed from S^1 (*i.e.* (n, k_j, k_ℓ, k_m)) provides at least as much throughput as each of the two removals. Thus, throughput of tuple removed from S^1 in this step is at least half the throughput of tuples removed from S^*. This proves that the first invariant is maintained.

For user k_i, let $B^*_{k_i,e}$ be the size of the remaining buffer of k_i after the tuples has been removed from S^*. Similarly, let $B^1_{k_i,e}$ be the buffer size for k_i after the single tuple has been removed from S^1. Then, we claim that the invariant $B^1_{k_i,e} \geq B^*_{k_i,e} \forall k_i : k_i \in S^*$ remains true after this step. During this step, for every user k_i that was in the tuple (n, k_j, k_ℓ, k_m), the buffer size for k_i dropped by b_{k_i}. However, for all such users that were also in S^*, we removed b_{k_i} bits from the buffer of k_i in S^* as well. Thus, the buffer of k_i will also drop by b_{k_i} in the optimal algorithm implying $B^1_{k_i} \geq B^*_{k_i}$. Only case where we would not be able

to remove b_{k_i} from the buffer of k_i in S^* is when k_i was not present in S^*. However, in this case as well, the second invariant is satisfied since the invariant is defined only for users that are in S^*.

Third invariant can simply be derived from the second variant. Since every user k_i scheduled in S^* has at least as much buffer left in S^1, therefore, if S^* can schedule k_i, then S^1 can also schedule it.

Observe that only the tuples corresponding to RB n were removed from S^1. At the same time, we also removed the tuple from S^* that corresponds to RB n. Thus, the fourth invariant also holds true after this step.

This proves that the four invariants hold true during this step of removing tuples from S^1 and S^*. Now, once all the tuples have been removed from S^1, then S^1 would be empty. Similarly, S^* would be empty (4th invariant). From the first invariant, we know that the throughput of tuples removed from S^1 is at least half the throughput of tuples removed from S^*. Thus, S^1 has at least half the throughput compared to S^*. □

Theorem A.4.2. *Alg2 has a complexity of $O(N^2K)$ and guarantees an approximation ratio $\frac{1}{4}$ compared to exponential-time optimal algorithm.*

Proof. Since for a given base station, at most three tuple can be assigned to a RB, therefore, the for loop in Line 4 of *Alg2* will be invoked at most $3N$ times. Further, within the for loop, there are at most $O(NK)$ tuples that need to be evaluated (Line 5-9) . Evaluating a tuple requires validating the buffer constraints of at most three users takes $O(1)$ time since the number of users in the tuple is at most 3. Thus, the complexity of *Alg2* is $O(N^2K)$.

The proof for approximation ratio for *Alg2* is similar to proof used in Theorem A.4.1. We compare the schedule (S^2) computed by *Alg2* with the schedule computed by optimal (S^*) where both S^2 and S^* are set of tuples ((n, k_i)). Further, the tuples in S^2 are arranged in the order of their throughputs (value of function f). During this comparison, we maintain the following four invariants:

- Throughput of tuples removed from S^2 is at least 1/4 the throughput of tuples removed from S^*.
- $B^1_{k_i} \geq B^*_{k_i} \forall k_i : k_i \in S^*$
- *Alg1* is free to choose any tuples that appear in S^* without violating any constraints in (5.10).
- If S^2 has no tuple on RB n, then S^* has no tuple on RB n either.

Initialization: Initially, no tuples have been removed from S^2 or S^*, so the throughput of tuples removed from both S^2 and S^* are 0. Hence, the first invariant holds. Secondly, all the users have the same buffer sizes before any of the scheduling algorithm has executed. Thus, $B^1_{k_i} = B^*_{k_i} \forall k_i$. This satisfies the second invariant. Thirdly, since S^* is a valid schedule and all users have the same buffer size in both the schedules (invariant 2), therefore, *Alg1* is free to choose any tuples from S^* without violating constraints. Fourthly, both S^2 and S^* will schedule at least one transmitter for each RB. Thus, the fourth invariant is true.

To show that S^2 has at least 1/4 the throughput compared to S^*, we start removing multiples tuples from S^2 and S^* without violating the invariants. Let (n, k_i) be the tuple with maximum throughput value in S^2 and (n, k'_i) be the tuple with max. throughput among all tuples in S^* for RB n. Clearly, $f(n, k_i) \geq f(n, k'_i)$ because otherwise the greedy algorithm would have chosen (n, k'_i) (follows from invariant 3). We remove (n, k_i) from S^2 and remove all tuples from S^* that are using RB n. Note that there can be at most 3 such tuples and (n, k_i) will have throughput at least as much as each of the three tuples (because invariant 3 implies that otherwise S^2 would have chosen one of those tuples instead of (n, k_i)).

Apart from this, for user k_i, let b_{k_i} be the number of bits that k_i was able to transmit/receive as a part of tuple of (n, k_i) in S^2. From the remaining tuples in S^*, we reduce the rate achieved by user k_i by b_{k_i}. This forms the fourth removal. Therefore, tuple $(n, k_i) \in S^2$

provides at least as much throughput as each of the four removals. Thus, throughput of tuple removed from S^2 in this step is at least 1/4 the throughput of tuples removed from S^*. This proves that the first invariant is maintained.

For user k_i, let $B^*_{k_i,e}$ be the buffer size of k_i after the tuples have been removed from S^*. Similarly, let $B^1_{k_i,e}$ be the buffer size for k_i after the single tuple has been removed from S^2. Then, we claim that $B^1_{k_i,e} \geq B^*_{i,e} \forall k_i : k_i \in S^*$. This is because before this step, this statement was true. Now during this step, for every user k_i that was in the tuple (n, k_j, k_ℓ, k_m) and S^*, we removed equal number of bits from S^* as well. Thus, the buffer of k_i will also drop by b_{k_i} in the optimal algorithm implying $B^1_{k_i} \geq B^*_{k_i}$. Only exception is that k_i may not be present in S^*. In that case, we wont be able to remove bits of k_i. However, the second invariant is still satisfied since it is defined only for users that are in S^*.

Third invariant can simply be derived from the second variant. If a user k_i has more buffer left in S^2, then if S^* can schedule k_i, then S^2 can also schedule it.

Observe that only the tuples corresponding to RB n were removed from S^2. At the same time, we also removed the tuple from S^* that corresponds to RB n.Thus, the fourth invariant also holds true.

This proves that the four invariants hold true during the process of removing tuples from S^2 and S^*. Now, once all the tuples have been removed from S^2, then S^2 would be empty. Similarly, S^* would be empty (4th invariant). From the first invariant, we know that the throughput of tuples removed from S^2 is at least 1/4 the throughput of tuples removed from S^*. Thus, S^2 has at least 1/4 the throughput compared to S^*. □

Theorem A.4.3. *Alg3 has a complexity of $O(N^4K^3)$ and guarantees an approximation ratio of $\frac{1}{3}$.*

Proof. We first show that complexity of *Alg3* is $O(N^3K^3)$. Observe that the for loop in line 12 of *Alg3* will be executed at most $2N^2$ times. For each execution of for loop, *Alg3*

executes one instance of *Alg1* in Line 14. Since, *Alg1* has complexity of $O(N^2K^3)$ (See Theorem A.4.1), therefore, complexity of *Alg3* is $O(N^4K^3)$.

To prove the approximation ratio, we will invoke nested sub-modularity, and leverage the following result from [17, 31]: If the incremental oracle is only α-approximable, then the approximation guarantee of greedy sub-modular maximization changes to $\frac{\alpha}{p+\alpha}$, where the maximization is subject to a p-independence system.

Submodularity: We show that at the highest level, we have the following submodular problem: $\psi = \{k_i, F_{k_i}\} \forall k_i \in S \forall F_{k_i} : F_{k_i} \in \{F_{k_i}[0], F_{k_i}[1]\}\}$ $\phi_{k_i} = \{k_i, F_{k_i}\} \forall F_{k_i} : F_{k_i} \in \{F_{k_i}[0], F_{k_i}[1]\}$

Now the set of all feasible schedules would correspond to a partition matroid [7] of ψ, whereby there can be at most one element from each ϕ_{k_i}, for any $A \in S$. For two sets A and B such that $A \subseteq B$, we can see that, $f(A \cup \{k_i, F_{k_i}\}) - f(A) \leq f(B \cup \{k_i, F_{k_i}\}) - f(B)$. This is because the benefit provided by $\{k_i, F_{k_i}\}$ when it is added to a bigger set B, can also be realized when it is added to a smaller set A. Therefore, the marginal gain will always be less than the marginal gain when adding to smaller set. Also, clearly $f(\phi) = 0$. Also, $f(A)$ is a non-decreasing function on A since adding one more user to A cannot decrease the objective function provided computation done by *Alg1* in Lines 15-16 is optimal. Thus, the scheduling problem at the highest level is submodular on a partition matroid and the approximation ratio of the greedy algorithm *Alg3* would be 1/2 [7] if *Alg1* in Lines 15-16 is optimal.

However, since *Alg1* is not optimal and has an approximation ratio of $\frac{1}{2}$ (See Theorem A.4.1), therefore, we invoke the result discussed above to obtain approximation ratio for *Alg3*. Setting $p = 1$ (because we can pick at most one element from each ϕ_{k_i}, for any $A \in S$, implying independence is 1), and α as $\frac{1}{2}$, we get: Approximation ratio $= \frac{\frac{1}{2}}{1+\frac{1}{2}} = \frac{1}{3}$ $\square$

BIBLIOGRAPHY

[1] Ettus Research. http://www.ettus.com/.

[2] OctoClock-G, accessed Jan. 2014. `https://www.ettus.com/product/details/OctoClock-G`.

[3] 3rd Generation Partnership Project (3GPP). *LTE Release 12*, 2013-2014. `http://www.3gpp.org/Release-12`.

[4] F. Adib, S. Kumar, O. Aryan, S. Gollakota, and D. Katabi. Interference Alignment by Motion. In *Proc. of ACM MobiCom 2013*.

[5] A. Agarwal and P. Kumar. Capacity bounds for ad hoc and hybrid wireless networks. *ACM SIGCOMM Computer Communication Review*, 34(3):71–81, 2004.

[6] S. H. Ali and V. C. Leung. Dynamic Frequency Allocation in FFR OFDMA Networks. *IEEE TWC*, 8(8):4286–4295, 2009.

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

[8] V. S. Annapureddy, A. El Gamal, and V. V. Veeravalli. Degrees of Freedom of Interference Channels with CoMP Transmission and Reception. *IEEE Transactions on Information Theory*, 58(9):5740–5760, 2012.

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

[10] A. Bachir and *et al*. Hidden Nodes Avoidance in Wireless Sensor Networks. In *Proc. of IEEE WirelessCom*, 2005.

[11] T. Bansal, B. Chen, P. Sinha, and K. Srinivasan. Symphony: Cooperative Packet Recovery over the Wired Backbone in Enterprise WLANs. In *Proc. of ACM MobiCom 2013*.

[12] T. Bansal, K. Sundaresan, S. Rangarajan, and P. Sinha. R2D2: Embracing Device-to-Device Communication in Next Generation Cellular Networks . In *Proceedings of IEEE INFOCOM*, 2014.

[13] T. Bansal, W. Zhou, K. Srinivasan, and P. Sinha. RobinHood: Sharing the Happiness in a Wireless Jungle. In *Proc. of ACM HotMobile 2014*.

[14] Y. Bejerano and R. S. Bhatia. MiFi: A Framework for Fairness and QoS Assurance for Current IEEE 802.11 Networks with Multiple Access Points. *IEEE/ACM Transactions on Networking (TON)*, 14(4):849–862, 2006.

[15] Y. Bejerano, S.-J. Han, and L. E. Li. Fairness and Load Balancing in Wireless LANs Using Association Control. In *Proc. ACM MOBICOM 2004*.

[16] V. R. Cadambe and S. A. Jafar. Interference Alignment and the Degrees of Freedom for the K User Interference Channel. *IEEE Transactions on Information Theory*, 2007.

[17] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[18] R. Y. Chang and et al. A Graph Approach to Dynamic FFR in Multi-Cell OFDMA Networks. In *Proc. of IEEE ICC*, 2009.

[19] A. Cidon, K. Nagaraj, S. Katti, and P. Viswanath. Flashback: Decoupled Lightweight Wireless Control. In *Proc. ACM SIGCOMM 2012*.

[20] K. Doppler, M. Rinne, C. Wijting, C. Ribeiro, and K. Hugl. D2D Communication as an Underlay to LTE-Advanced Networks. *IEEE Communications Magazine*, 47(12):42–49, 2009.

[21] K. Doppler, C.-H. Yu, C. B. Ribeiro, and P. Janis. Mode Selection for D2D Communication Underlaying an LTE-Advanced Network. In *Proc. of IEEE WCNC*, 2010.

[22] Q. Duong, Y. Shin, and O.-S. Shin. Resource Allocation Scheme for D2D Communications Underlaying Cellular Networks. In *IEEE Intl. Conf. on Computing, Management and Telecommunications (ComManTel)*, 2013.

[23] S. C. Ergen and P. Varaiya. TDMA Scheduling Algorithms for Wireless Sensor Networks. *Wireless Networks*, 16(4):985–997, 2010.

[24] P. Festa and *et al*. Feedback Set Problems. *Handbook of combinatorial optimization*, 4:209–258, 1999.

[25] G. Fodor and N. Reider. A Distributed Power Control Scheme for Cellular Network Assisted D2D Communications. In *Proc. of IEEE GLOBECOM*, 2011.

[26] M. Franceschetti, O. Dousse, D. N. Tse, and P. Thiran. Closing the gap in the capacity of wireless networks via percolation theory. *IEEE Transactions on Information Theory*, 53(3):1009–1018, 2007.

[27] D. Gesbert, M. Kountouris, R. Heath, C.-B. Chae, and T. Salzer. Shifting the MIMO Paradigm. *IEEE Signal Processing Magazine*, 24(5):36–46, Sept 2007.

[28] R. Gold. Optimal binary sequences for spread spectrum multiplexing (Corresp.). *Information Theory, IEEE Transactions on*, 13(4):619 –621, 1967.

[29] S. Gollakota and D. Katabi. Zigzag Decoding: Combating Hidden Terminals in Wireless Networks. In *Proc. ACM SIGCOMM 2012*.

[30] S. Gollakota, S. D. Perli, and D. Katabi. Interference Alignment and Cancellation. In *Proc. ACM SIGCOMM 2009*.

[31] P. R. Goundan and A. S. Schulz. Revisiting the Greedy Approach to Submodular Set Function Maximization. *Optimization online*, pages 1–25, 2007.

[32] M. Gowda, S. Sen, R. Roy Choudhury, and L. S. Cooperative Packet Recovery in Enterprise WLANs. In *Proc. IEEE INFOCOM 2013*.

[33] A. Gudipati, S. Perreira, and S. Katti. AutoMAC: Rateless Wireless Concurrent Medium Access. In *Proc. of ACM MOBICOM*, 2012.

[34] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[35] C. Hua and R. Zheng. Starvation Modeling and Identification in Dense 802.11 Wireless Community Networks. In *In Proc. IEEE INFOCOM*, 2008.

[36] IEEE. *IEEE Std 802.11-2007 - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* , 2007. http://standards.ieee.org/about/get/802/802.11.html

[37] S. A. Jafar. *Interference Alignment: A New Look at Signal Dimensions in a Communication Network*. Now Publishers, 2011.

[38] K. Jamieson and H. Balakrishnan. PPR: Partial Packet Recovery for Wireless Networks. In *Proc. of ACM SIGCOMM*, 2007.

[39] P. Jänis and et al. D2D Communication Underlaying Cellular Communications Systems. *IJCNS*, 2(3):169–178, 2009.

[40] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding Congestion in IEEE 802.11b Wireless Networks. In *Proc. of Internet Measurment Conference*, 2005.

[41] S. Johannessen. Time Synchronization in a Local Area Network. *Control Systems, IEEE*, 24(2), 2004.

[42] A. Kamerman and L. Monteban. WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band. *Bell Lab Technical Journal*, pages 118–133, Summer 1997.

[43] S. Katti, S. Gollakota, and D. Katabi. Embracing Wireless Interference: Analog Network Coding. In *Proc. of ACM SIGCOMM*, 2007.

[44] B. Kaufman and B. Aazhang. Cellular Networks with an Overlaid Device to Device Network. In *Proc. of IEEE Asilomar Conference on Signals, Systems and Computers*, 2008.

[45] D. Koutsonikolas and *et al*. TDM MAC Protocol Design and Implementation for Wireless Mesh Networks. In *Proc. ACM CoNEXT 2008*.

[46] M. Kuhn, S. Berger, I. Hammerstrom, and A. Wittneben. Power Line Enhanced Cooperative Wireless Communications. *IEEE JSAC*, 24(7):1401–1410, 2006.

[47] S. Kumar, D. Cifuentes, S. Gollakota, and D. Katabi. Bringing Cross-Layer MIMO to Today's Wireless LANs. In *Proc. of ACM SIGCOMM 2013*.

[48] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *Proc. ACM MSWIM 2004*.

[49] R. Laufer, T. Salonidis, H. Lundgren, and P. Le Guyadec. XPRESS: A Cross-Layer Backpressure Architecture for Wireless Multi-Hop Networks. In *Proc. ACM MOBICOM 2011*.

[50] N. Lee, X. Lin, J. G. Andrews, and R. W. Heath Jr. Power Control for D2D Underlaid Cellular Networks: Modeling, Algorithms and Analysis. *arXiv preprint arXiv:1305.6161*, 2013.

[51] L. Lei, Z. Zhong, C. Lin, and X. Shen. Operator Controlled D2D Communications in LTE-Advanced Networks. *IEEE Wireless Communications*, 19(3):96–104, 2012.

[52] T. Li and *et al*. CRMA: Collision-Resistant Multiple Access. In *Proc. of ACM Mobicom*, 2011.

[53] B. Liu, P. Thiran, and D. Towsley. Capacity of a wireless ad hoc network with infrastructure. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 239–246, 2007.

[54] E. Liu, Q. Zhang, and K. K. Leung. Relay-Assisted Transmission with Fairness Constraint for Cellular Networks. *IEEE TMC*, 11(2):230–239, 2012.

[55] E. Magistretti, O. Gurewitz, and E. Knightly. 802.11 ec: Collision Avoidance Without Control Messages. In *Proc. of ACM MOBICOM*, 2012.

[56] A. Miu, H. Balakrishnan, and C. E. Koksal. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. In *Proc. ACM MOBICOM 2005*.

[57] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill. Designing High Performance Enterprise Wi-Fi Networks. In *Proc. USENIX NSDI 2008*.

[58] B. Nazer and *et al*. Ergodic Interference Alignment. In *Proc. of IEEE ISIT 2009*.

[59] A. Ozgur, O. Lévêque, and D. N. Tse. Hierarchical cooperation achieves optimal capacity scaling in ad hoc networks. *IEEE Transactions on Information Theory*, 53(10):3549–3572, 2007.

[60] J. Paek and et al. Energy-Efficient Positioning for Smartphones using Cell-id Sequence Matching. In *Proc. of ACM MOBISYS*, 2011.

[61] U. Paul and et al. Understanding Traffic Dynamics in Cellular Data Networks. In *Proc. of IEEE INFOCOM*, 2011.

[62] T. Peng and et al. Interference Avoidance Mechanisms in the Hybrid Cellular and D2D Systems. In *Proc. of IEEE PIMRC*, 2009.

[63] A. Rahman and P. Gburzynski. Hidden Problems With the Hidden Node Problem. In *In Proc. IEEE Biennial Symposium on Communications, 2006*.

[64] H. Rahul, H. Hassanieh, and D. Katabi. SourceSync: A Distributed Wireless Architecture for Exploiting Sender Diversity. In *Proc. of ACM SIGCOMM 2010*.

[65] H. Rahul, S. Kumar, and D. Katabi. MegaMIMO: Scaling Wireless Capacity with User Demand. In *Proc. ACM SIGCOMM 2012*.

[66] B. Rankov and A. Wittneben. Spectral Efficient Protocols for Half-Duplex Fading Relay Channels. *IEEE Journal on Selected Areas in Communications*, 25(2):379–389, 2007.

[67] A. Schulman, D. Levin, and N. Spring. CRAWDAD data set umd/sigcomm2008. Downloaded from http://crawdad.cs.dartmouth.edu/umd/sigcomm2008.

[68] S. Sen, R. R. Choudhury, and S. Nelakuditi. No Time to Countdown: Migrating Backoff to the Frequency Domain. In *MOBICOM*, 2011.

[69] S. Sen, R. Roy Choudhury, and S. Nelakuditi. CSMA/CN: Carrier Sense Multiple Access with Collision Notification. In *Proc. of ACM Mobicom 2010*.

[70] S. Sen, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi. Successive Interference Cancellation: Carving Out MAC Layer Opportunities. *IEEE Trans. Mob. Comput.*, 12(2):346–357, 2013.

[71] V. Shrivastava and *et al*. CENTAUR: Realizing the Full Potential of Centralized WLANs Through a Hybrid Data Path. In *Proc. of ACM MobiCom 2009*.

[72] V. Shrivastava and *et al*. CENTAUR: Realizing the Full Potential of Centralized WLANs Through a Hybrid Data Path. In *Proc. of ACM Mobicom*, 2009.

[73] G. Song and Y. Li. Cross-Layer Optimization for OFDM Wireless Networks - Part I: Theoretical Framework. *IEEE TWC*, 4(2), 2005.

[74] Stanford Information Networking Group (SING). SING Datasets. `http://sing.stanford.edu/srikank/datasets.html`.

[75] A. L. Stolyar and H. Viswanathan. Self-Organizing Dynamic FFR for Best-Effort Traffic Through Distributed Inter-Cell Coordination. In *Proc. of IEEE INFOCOM*, 2009.

[76] C. Suh, M. Ho, and D. N. Tse. Downlink Interference Alignment. *IEEE Transactions on Communications*, 59(9):2616–2626, 2011.

[77] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in WSNs. In *Proc. of ACM SenSys*, 2008.

[78] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *Proc. of ACM SIGCOMM*, 2009.

[79] G. R. Woo, P. Kheradpour, D. Shen, and D. Katabi. Beyond the Bits: Cooperative Packet Recovery Using Physical Layer Information. In *Proc. ACM MOBICOM 2007*.

[80] X. Xie, X. Zhang, and K. Sundaresan. Adaptive Feedback Compression for MIMO Networks. In *Proc. of ACM MobiCom 2013*.

[81] W. Zhou, D. Li, K. Srinivasan, and P. Sinha. DOMINO: Relative Scheduling in Enterprise Wireless LANs. In *Proc. of ACM CoNEXT 2013*.