Towards Scalable Design of Future Wireless Networks

by

Krishna Chaitanya Garikipati

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2016

Doctoral Committee:

Professor Kang G. Shin, Chair
Associate Professor Achilleas Anastasopoulos
Assistant Professor N M Mosharaf K. Chowdhury
Professor Mingyan Liu

To my family

ACKNOWLEDGEMENTS

First and foremost I would like to express my gratitude to my advisor, Prof. Kang Shin, to whom I owe the most for teaching me about different aspects of research. Prof. Shin is an outstanding researcher and an excellent advisor. I am deeply indebted to him for imparting his knowledge and wisdom, and most importantly, for giving me the freedom to pursue problems of my liking. His guidance, encouragement, and patience have helped me immensely in my doctorate study.

I am also grateful to my dissertation committee members, Prof. Mingyan Liu, Prof. Achilleas Anastasopoulos and Prof. Mosharaf K. Chowdhury, for devoting their time to my thesis and providing valuable comments.

I also wish to thank my colleagues in the Real-Time Computing Lab (RTCL) for their contributions. Kassem Fawaz has been a great mentor, and I cherish our discussions on research and life in general. Eugene Chai introduced me to software-radio programming that has influenced much of my research work. I also benefited from my interactions with other RTCL members, Seunghyun Choi, Arun Ganesan, Xiaoen Ju, Huan Feng, Dongyao Chen, Yu-Chih Tung, Sihui Han, Michael Zhang and Xinyu Zhang, who gave useful and timely feedback on my work.

I am also grateful to my internship collaborators at NEC Labs and Nokia Research, for giving me the opportunity to work on interesting topics. Also, I like to give credit to the EECS department faculty and staff for providing an exceptional environment for learning and doing research.

I also like to thank all my friends in Ann Arbor, Sai, Deeksha, Ananda, Kon-

stantinos, Maruthi, Raj and others, who made my stay fun and memorable.

Finally, I would like to thank my parents and my sister for their unconditional support through this long journey. To them, I dedicate this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Towards Scalable Design of Future Wireless Networks

by

Krishna C. Garikipati

Chair: Kang G. Shin

Wireless operators face an ever-growing challenge to meet the throughput and processing requirements of billions of devices that are getting connected. In current wireless networks, such as LTE and WiFi, these requirements are addressed by provisioning more resources: spectrum, transmitters, and baseband processors. However, this simple add-on approach to scale system performance is expensive and often results in resource underutilization. What are, then, the ways to efficiently scale the throughput and operational efficiency of these wireless networks? To answer this question, this thesis explores several potential designs: utilizing unlicensed spectrum to augment the bandwidth of a licensed network; coordinating transmitters to increase system throughput; and finally, centralizing wireless processing to reduce computing costs.

First, we propose a solution that allows LTE, a licensed wireless standard, to coexist with WiFi in the unlicensed spectrum. The proposed solution bridges the incompatibility between the fixed access of LTE, and the random access of WiFi, through channel reservation. It achieves a fair LTE-WiFi co-existence despite the transmis-

sion gaps and unequal frame durations. Second, we consider a system where different MIMO transmitters coordinate to transmit data of multiple users. We present an adaptive design of the channel feedback protocol that mitigates interference resulting from the imperfect channel information. Finally, we consider a Cloud-RAN architecture where a datacenter or a cloud resource processes wireless frames. We introduce a tree-based design for real-time transport of baseband samples and provide its end-to-end schedulability and capacity analysis. We also present a processing framework that combines real-time scheduling with fine-grained parallelism. The framework reduces processing times by migrating parallelizable tasks to idle compute resources, and thus, decreases the processing deadline-misses at no additional cost.

We implement and evaluate the above solutions using software-radio platforms and off-the-shelf radios, and confirm their applicability in real-world settings.

CHAPTER I

Introduction

Wireless networks, such as LTE and WiFi, are an integral part of our everyday life. They permeate our daily activities and routines, connecting our smartphones, tablets, and laptops to the online world. However, they face an unprecedented challenge in the next decade from our desire to connect a broad range of devices. Indeed, with the advent of Internet-of-Things (IoT), some 50 billion devices are expected to come online by 2020 [1]. This growth is in addition to a steady increase in mobile data usage that will grow 10× between 2014 and 2019 [2]. The resulting demand for connectivity and throughput will strain the current wireless deployments that already operate near maximum link capacity.

The past few decades has seen remarkable progress in the link layer performance of wireless networks. It is now common for wireless standards to employ capacity achieving codes (e.g. Turbo codes), efficient modulation schemes (e.g., 256-QAM), and multiple-antenna technology (MIMO). These enhancements provide an order-of-magnitude improvement in the wireless throughput over previous standards, for example, from few Mbps in 802.11b to Gbps in IEEE 802.11ac [3]. However, sustaining this trend is becoming increasingly difficult; for instance, the number of antennas is constrained by the small form-factor of the mobile devices.

As link layer performance shows signs of saturation, there are a considerable num-

ber of design bottlenecks at the system level of existing wireless networks. It is well known that interference is the primary source of performance loss in dense networks [4]. This is compounded by the fact that there is only limited bandwidth available due to a small amount of spectrum (licensed or unlicensed) that can be bought or shared. Furthermore, the costs of maintaining and upgrading access points (APs) or basestations (BSs) are increasing manifold, making it uneconomical for cellular operators to expand their networks [5].

To address these issues, we require innovative approaches to design, and in a way, rethink the wireless networks to make them future-ready. The next order of gains will be based on measures that go beyond a single layer (PHY or MAC), and consider the system as a whole. Such measures will invoke cross-layer (e.g., PHY access and application semantics) and inter-disciplinary (e.g., MIMO design and real-time analysis) approaches that have so far received only limited attention in the research community.

1.1 Background

Before we discuss potential solutions, we describe the role of existing design choices in wireless networks and their impact on the system performance.

1.1.1 Existing Design Choices

Spectrum segregation. The FCC divides the wireless spectrum into two broad categories: licensed and unlicensed. The licensed spectrum is owned by an operator who is its sole user, which guarantees no interference from other operators or devices. For example, cellular networks such as 3G, LTE, etc. run on the licensed spectrum. On the other hand, the unlicensed spectrum can be accessed by any device as long as it uses certified radio equipment and complies with technical requirements, including

the power limit specifications[1]. One such example is the ISM band (2.4 and 5 GHz) that is widely used by WiFi, Bluetooth, Zigbee and other wireless protocols.

The division of the spectrum resources, though easy to regulate, does not offer the flexibility to use underutilized channels. The licensed TV bands have low occupancy for long periods [6] . Similarly, license-exempt channels in the 5GHz, such as DFS bands, are sparsely used [7]. This is far from ideal, as bandwidth-starved devices should be able to access unoccupied spectrum without restrictions.

Uncoordinated transmissions. Since the introduction of cellular networks in the 1980s, frequency-reuse, and more recently, fractional frequency reuse (FFR), has been the conventional approach to manage interference in wireless networks [8]. In FFR, the inter-cell interference at cell-edge users is minimized by assigning orthogonal channels to the adjoining cells. Each cell operates independently, and basestations in the network transmit without coordination. However, modern network deployments are extremely dense and have limited number of non-overlapping channels (e.g., 2.4GHz ISM band has three 20MHz non-overlapping channels). In such scenarios, FFR offers limited resistance against interference from uncoordinated transmissions.

Best-effort transport. Upcoming wireless applications, such as Massive MIMO [9, 10], demand new baseband architectures. Massive MIMO involves centralized wireless processing away from the radio hardware, which presumes real-time transport of baseband samples. Current packet-switching networks, however, are designed for best effort traffic flows and are unable to prioritize or distinguish packets based on their delays. For instance, aggregating packet traffic using Ethernet switches can introduce unpredictable queuing delays that can lead to deadline misses [11]. This can negatively impact the wireless throughput as missed transport deadlines result in decoding failures.

[1]FCC Part 15 Rules

Hardware-reliance. Wireless standards and technologies are under a constant state of evolution. Many innovations, such as full-duplex, interference cancellation, etc., are taking place at the PHY (or baseband) layer. To continuously incorporate these changes, the wireless processing at basestations should be fully reconfigurable. On the other hand, existing wireless networks rely on dedicated hardware (e.g., DSP, ASIC) that are difficult to program or even modify. Additionally, while wireless traffic has large temporal and spatial variations (e.g., daytime vs. nighttime), the baseband hardware is provisioned according to the peak usage of the network, which results in underutilization of expensive hardware resources [12]. These factors, combined, have resulted in increasing operating costs to cellular providers, threatening their profitability and ultimately, end-user experience.

1.2 Towards Scalable Design

The increasing demand for throughput, bandwidth, and operational efficiency challenges the design choices of existing networks. Towards this, we propose *scalable* solutions that address the shortcomings of existing approaches. By scalable, we refer to the ability to improve system performance in proportion to the resources used.

1.2.1 Proposed Solutions

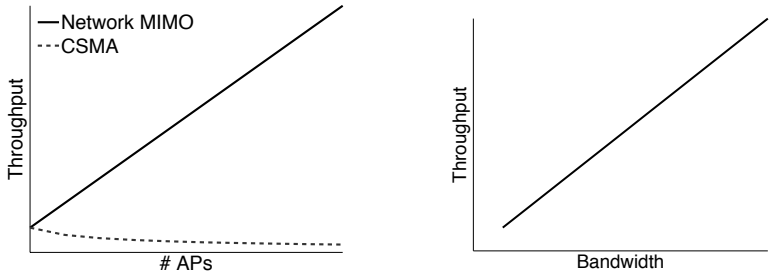
Augmenting bandwidth. Current cellular networks face a huge shortage of spectrum; they utilize channels of bandwidth no more than few tens of MHz. Despite the fact that additional bandwidth can easily increase the network throughput, current designs do not augment system bandwidth with other bands, for example, unlicensed channels. On the other hand, FCC has recently converted vast amounts of the spectrum (e.g., 800 MHz, 3.5 GHz bands) into license-exempt channels that can be freely utilized. These changes are beneficial for licensed networks that are facing severe spectrum shortage. Indeed, there are several proposals for LTE to use

the 5GHz unlicensed band [7, 13, 14] . However, to utilize the unlicensed spectrum requires a new sharing mechanism as licensed users must now co-exist with other unlicensed users. This sharing mechanism should incorporate the notion of *fairness* between licensed and unlicensed users to justify the free usage of unlicensed bands. In this direction, we introduce a new sharing mechanism for licensed users based on CSMA access. Our solution is a practical way of maintaining fair usage of the unlicensed spectrum.

Increasing throughput. The primary cause of interference lies in the lack of common knowledge between transmitters. We propose to use Network MIMO that bridges this gap through a design where MIMO-capable APs coordinate their transmissions to eliminate interference (or cross-talk) by employing a process called precoding [15]. While precoding (also known as beamforming) has been extensively studied in the literature, realizing it in Network MIMO presents additional hurdles; the transmitters require synchronization in both time and frequency [16, 17]. With additional mechanisms in place to mitigate the impact of channel degradation, Network MIMO enables a near interference-free channel even in the presence of other transmitters.

Guaranteed transport. Applications relying on centralized wireless processing require real-time transport of baseband samples. Typically, such applications employ tens and hundreds of antennas. While various architectures (e.g., ARGOS [10], BigStation [18]) have been proposed, their analysis from a real-time perspective is lacking. We introduce a principled approach for the design of baseband transport networks. In particular, assuming a packet-switched topology, we propose a Fat-Tree design that can scale across the antennas/radios while meeting end-to-end delay guarantees. We provide a sufficient condition for schedulability and also study its impact on the wireless capacity.

Efficient processing. Resource pooling is a robust strategy commonly employed in datacenters and cloud infrastructure to save computing costs. It also benefits base-



(a) Transmitter coordination (b) Unlicensed access

Figure 1.1: Performance scaling under ideal conditions.

band processing, since a pool of compute resources in a cloud could replace dedicated basestation hardware. This approach, also known as Cloud-RAN [5], offers tremendous savings on the energy, and the hardware costs compared to dedicated architectures [19]. Previous studies have shown around 22% savings in compute resources [19]. Additionally, Cloud-RAN allows easy upgrades and amenability to implement coordinated processing. While resource pooling relies on the statistical information of basestation loads, it does not account for the load variations at finer timescales. As a result, resource pooling on a multi-core platform leads to underutilized CPU cycles. We propose to utilize these free CPU cycles for parallelizing tasks, resulting in reduced processing times, and fewer deadline-misses.

1.2.2 Achieving Scalability

Scalability of a system can be of two types: scale-out and scale-up. Viewing the system as a collection of modules, scale-out relates to the ability of the system to accommodate more modules to improve aggregated performance while scale-up relates to the ability to improve the performance of each module. Though desirable, scalability is not an inherent feature of current wireless deployments.

Scaling-out. The system capacity in CSMA-based access does not increase proportionally with the number of APs, n, but instead scales as $\Omega(1/\sqrt{n})$ [20]. On the other hand, the capacity increases linearly with the number of transmit antennas (or

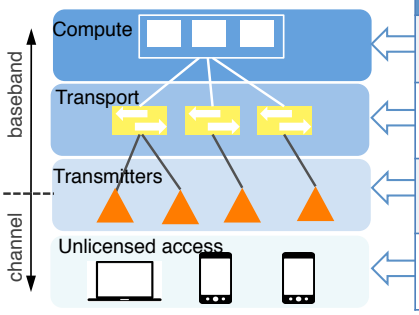
number of transmitters) in a Network MIMO system [21]. To see this, consider a wireless communication channel with n transmit and m receive antennas, and a $m \times n$ channel matrix denoted by $\mathbf{H}$. Let ρ denote the signal-to-noise ratio per antenna, assumed to be equal across the antennas, and let W be the channel bandwidth. The channel capacity (in bits/s) is given as [15, chapter 8]:

$$C = W \log_2 \det(\mathbf{I} + \rho \mathbf{H}\mathbf{H}^H) \tag{1.1}$$

where $\mathbf{I}$ is the $m \times m$ identity matrix. Under ideal conditions of medium to high SNR ($\rho \gg 1$), the capacity, C, scales as the maximum rank of the channel matrix, $\min(m, n)$, while at low SNR ($\rho \ll 1$), the capacity scales as mn. In both cases, the capacity increases with the number of transmit antennas, m. Fig. 1.1(a) illustrates this capacity scaling in Network MIMO compared to CSMA-based access, where each AP is assumed to have a single antenna.

Scaling-out with the number of transmitters also means that the transport network should accommodate an increasing amount of baseband traffic. As the baseband traffic is delay sensitive, the scaling should ensure the transport delay is not adversely affected. We achieve this with a tree-based design that supports a high degree of traffic aggregation while being tractable for end-to-end schedulability.

Scaling-up. Most cellular operators run separate WiFi hotspots on top of their licensed network to offload over-subscribed cellular traffic. Instead, it is more efficient to leverage their existing licensed infrastructure and transmit cellular signals (such as LTE) in unlicensed spectrum. As this impacts the WiFi incumbents, sharing mechanisms must ensure fair co-existence. Once deployed, it allows operators to scale-up their expensive infrastructure across a wide number of unlicensed channels. From Eq. (1.1), the capacity increases linearly with the bandwidth W, as illustrated in Fig. 1.1(b). Hence, unlicensed access can deliver more wireless throughput while



System	Subtopic	Objective	Scaling
RT-OPEX	Baseband processing on GPP	Reducing deadline-miss rate	Scale-up
DISTRO	Real-time baseband transport	Scalable with end-to-end schedulability	Scale-out
netMIMO	Coordinated MIMO transmitters	Mitigating interference from CSI aging	Scale-out
TALOS	Access of unlicensed spectrum	Collision-free and fair co-existence	Scale-up

Figure 1.2: Summary of the thesis contributions showing their applicability to the wireless system stack.

using the same cellular infrastructure.

Centralized architectures such as Cloud-RAN rely on off-the-shelf multi-core compute platforms for baseband processing. The compute resources are provisioned according to long-term cellular load forecasts, typically over the duration of hours [19]. On the other hand, cellular load exhibits significant variability at much finer timescales, often in the order of milliseconds. By making the processing flexible to the minute traffic variations, for instance, utilizing idle compute cycles to offload some of the parallelizable subtasks of baseband processing, can reduce the execution time. As a result, the deadline misses are reduced at no additional cost, increasing the computing efficiency.

1.2.3 Thesis Summary

In this thesis, we start with the question of how to design a scalable wireless network. Towards this, we explore four potential designs that form the basis of such a system. Fig. 1.2 shows the summary of the designs with their scope, objective and the type of scalability. The figure also shows a modularized view of the wireless networking stack spanning channel access, transmitter coordination, baseband transport and baseband processing. Since each of the modules are self-contained, they are treated and evaluated separately in this thesis. For the rest of this thesis, we discuss our

approaches in the context of the current wireless standards, LTE (3GPP LTE) and WiFi (IEEE 802.11), and propose modifications wherever necessary.

Each of our designs targets one particular layer in the wireless system stack. The designs are summarized as follows:

TALOS: LTE in the unlicensed spectrum has been proposed to meet the unprecedented demands faced by operators from ubiquitous, bandwidth-hungry mobile services/applications. However, the access paradigms in LTE (centralized/synchronous) and WiFi (distributed/asynchronous) are fundamentally incompatible. LTE is not designed to yield the channel to WiFi, and may potentially starve WiFi of bandwidth. While simple on-off duty access mechanisms for LTE are considered by the industry as a means to tackle this challenging co-existence problem, we demonstrate several artifacts inherent to LTE transmissions that render such mechanisms ineffective. We design and implement TALOS, a co-existence protocol that departs from LTE's pure synchronous operation to a novel policy of asynchronous access and synchronous transmission (A2TS) to bridge the paradigm gap. TALOS allows LTE to contend asynchronously for access to the channel in a manner that is efficient and fair to WiFi, while preserving the compliance and benefits of synchronous transmissions in LTE. TALOS requires no change to WiFi and LTE protocols, can be easily realized on LTE base stations alone, and delivers a superior co-existence performance for both LTE and WiFi over current approaches.

netMIMO: In Network MIMO, that relies on channel feedback, there are the challenges of interference that arise from the aging of the Channel State Information (CSI). Particularly, in scenarios of high channel mobility–where Network MIMO is expected to be used–the performance degradation from interference can be severe. We design transmission schemes to address the two sources of CSI aging: delay in acquiring CSI feedback, and transmission period over which CSI remains unchanged. We introduce a two-phase training and feedback protocol that balances CSI aging

across users by allowing users with high interference to have a shorter feedback delay. We also introduce an adaptive adjustment to the transmission length to reduce the decoding errors caused from interference. The proposed protocols adapt to the measurements collected from the users and hence applicable to various mobility scenarios.

DISTRO: Upcoming wireless deployments such as C-RAN and Massive-MIMO rely on real-time transport of baseband samples. The radios (front-ends) in such deployments generate baseband packets once every period, which are transported to the backend processing cluster. Therefore, to meet the real-time processing constraints, the baseband transport network must deliver the packets within a fixed end-to-end delay bound. However, computing the queuing delays in a large-scale packet-switched network is intractable, making it difficult to provide end-to-end delay guarantees. We present DISTRO, a novel Fat-Tree-based design for transporting baseband samples. DISTRO's design supports real-time transport as it allows us to bound the maximum transport delay of each packet. As a result, the network switches can implement well-known scheduling policies to achieve end-to-end delay guarantees. DISTRO partitions the transport network into a separate aggregation- and edge-switch network, such that any scheduling policy changes occur only at the edge switch network. This logical division of the network along with the tree-based design enables transport scaling to a large number of radios. We also characterize the maximum wireless capacity that can be achieved while meeting the real-time constraints of baseband transport.

RT-OPEX: Processing in virtualized hardware systems, such as in C-RAN, must respect wireless protocol deadlines, for example, 3ms to transport, decode and respond to an LTE uplink frame. However, the commonly used processing (e.g. parallelism) and scheduling techniques (e.g. partitioned) for wireless processing are inefficient as they result in either over-provisioning of resources or suffer from deadline misses. This inefficiency stems from the large variations in processing times due to fluctuations in wireless traffic. We present a new framework called RT-OPEX, that unlike

state-of-the-art bridges the gap between scheduling and parallelism. RT-OPEX reduces processing times by migrating parallelizable tasks to idle compute resources at runtime, and thus lowers the deadline misses at no additional cost. We implement and evaluate RT-OPEX on a commodity GPP platform using realistic cellular load traces.

1.3 Thesis Organization

The rest of the thesis is organized as follows. Chapter II considers the problem of LTE and WiFi co-existence in the unlicensed spectrum. It introduces, TALOS, a novel and provable mechanism for fair co-existence of LTE and WiFi. Chapter III presents our approach for a measurement-based design of Network MIMO, which addresses the interference from CSI aging. In Chapter IV, we design a Fat-Tree based network for baseband transport, DISTRO, that can enable real-time delivery of baseband samples. In Chapter V, we implement a new processing framework, RT-OPEX, that relies on run-time migration of subtasks to reduce deadline-miss rates. Finally, we conclude and discuss the future directions in Chapter VII.

CHAPTER II

Co-existence of LTE and WiFi in Unlicensed Spectrum

2.1 Introduction

Recently, LTE in the unlicensed spectrum (5GHz ISM bands [7]) has received enthusiastic support from the global mobile industry as a means to boost the capacity of LTE networks. The *carrier aggregation* [22, Chapter 19] feature of LTE allows operators to augment existing licensed carriers with new carriers placed in the unlicensed spectrum (*i.e.*, *unlicensed* carriers). This has the distinct benefit of reusing the same backend infrastructure that has proven to be capable of providing large-scale wireless connectivity, and can address the unprecedented bandwidth demands from ubiquitous Internet-of-Things (IoT) devices and mobile applications.

However, before deploying LTE into the unlicensed spectrum, one must ensure that LTE will co-exist fairly with both the existing WiFi networks and other unlicensed LTE networks. LTE is designed for always-on, synchronous and centrally managed channel access. This is, unfortunately, both incompatible with the CSMA (asynchronous, distributed) model of WiFi, and not conducive for co-existence with competing unlicensed LTE networks.

Current approaches: Two approaches have been currently proposed to support

LTE in the unlicensed spectrum: *LTE-Unlicensed* (LTE-U) and *Licensed Assisted Access LTE* (LAA-LTE). LTE-U can be realized today (*e.g.*, Qualcomm Snapdragon 820 Processor) as it requires no changes to the existing LTE standards, and uses adaptive on-off duty cycling of the LTE channel. LTE-U relies on energy sensing for channel access and dynamically adjusts only the on- and off-durations according to the measured WiFi utilization of the channel. The on-duration, typically on the order of hundreds of milliseconds, is suitable for LTE-to-LTE co-existence, but is too coarse-grained and leads to short-term unfairness to WiFi.[1]

LAA-LTE alleviates this short-term unfairness to WiFi by requiring LTE nodes to employ Listen-Before-Talk (LBT) for Clear Channel Assessments (CCA) like WiFi before LTE subframes are transmitted. LBT also relies on energy sensing but allows the LTE transmissions to be limited to shorter (1–10ms) time durations. Unlike LTE-U, shorter transmissions in LAA-LTE requires modifications to the LTE air interface. At present, LBT requirements are mandatory only in Europe and Japan, but not in the U.S., China and Korea.

Challenges. While LAA-LTE is a welcome step to solving short-term unfairness to WiFi, our in-depth experimental study reveals some fundamental challenges that still remain and undermine the feasiblility of operating LTE in the unlicensed spectrum.

Challenge I: Low Detection Sensitivity. Even with LBT, an LTE node employs only energy-sensing mechanisms [24, 25], which enables only WiFi/LTE signals that are over the CCA threshold (around -62 dBm) to be detected. On the other hand, WiFi uses preamble detection (in addition to energy sensing) and can detect other WiFi (but not LTE) signals as low as -82dBm. This is particularly troubling as LTE and WiFi networks are capable of successfully operating at signal levels of -80dBm and lower. From the prespective of a WiFi (LTE) node, such undetected LTE (WiFi)

[1]Current fairness claims of LTE-U are made at time scales of seconds [23] and not milliseconds.

signals (below the CCA threshold but above the noise floor) can still lead to collisions and severely degrade WiFi (LTE) performance. Our experimental study shows that WiFi (LTE) throughput can degrade by up to 75% (40%) with such LTE (WiFi) interference. Simply decreasing the CCA threshold will not address the problem since it only increases false positive detections, which will reduce the overall channel utilization.

Challenge II: Mis-interpreted Transmission Opportunities. LTE transmissions consist of consecutive subframes, each one millisecond in duration. Each subframe (both uplink and downlink) carries data and control information to/from multiple user elements (UEs). For various reasons we identify in Sec. 2.4, short, non-deliberate transmission gaps arise in these frame transmissions, preventing LTE from providing a continuous source of signal energy. An example of such a gap can be seen in uplink transmissions. Multiple UEs can transmit, one after the other across different symbols (each being 70μs), within the same uplink subframe. From the perspective of a WiFi node, the interference energy seen will be highly time-varying. Hence, a transmission from a distant UE will likely be seen as an energy-gap by the WiFi node. Our experimental profiling has shown these LTE transmit gaps to be typically much larger (median duration of 140 μs) than a WiFi slot duration (9μs). This, in turn, causes WiFi or LTE to mis-interpret these gaps as idle transmission opportunities, thereby resulting in WiFi–LTE and LTE–LTE collisions, impacting performance by as much as 75% in our study.

Our Solution: TALOS

These challenges reveal that current (LTE-U) and proposed (LAA-LTE) mechanisms are not just insufficient for co-existence but can lead to significant performance degradation in both WiFi and LTE. We address these challenges with Talos — a novel, fair and efficient co-existence mechanism between LTE and WiFi in unlicensed

spectrum. The root cause of the aforementioned challenges stems from the interaction of two fundamentally different access technologies — WiFi (carrier-sensed, distributed, asynchronous) and LTE (always-on, centralized and synchronous), that were designed for completely different purposes, in the same spectrum. TALOS bridges this division by making LTE and WiFi speak the same asynchronous language during channel access alone. This is a feature that, we believe, is critical for distributed co-existence between both LTE–WiFi as well as LTE–LTE (different operators). It assigns the co-existence burden to the LTE nodes (only base stations), requiring them to *access asynchronously* instead of synchronously (in a manner that is compatible with, and fair to the WiFi nodes) while allowing them to *transmit synchronously* (retaining compliance and benefits of synchronous LTE) in the un-licensed spectrum. We refer to this novel mode of LTE operation as *A2TS*.

For each unlicensed carrier (channel), TALOS employs a supplementary WiFi sensing module on the LTE base station (also called eNodeB) *solely* for the purpose of contention (access) on that carrier. Once the WiFi sensing module gains access to the channel, it broadcasts a CTS-to-Self with the NAV field set to the duration of the LTE transmission, and relinquishes operation to the LTE stack. The LTE stack then begins synchronous transmissions on the channel to its UEs during this reserved period. A2TS directly addresses the two key challenges identified. The use of the WiFi module for notification and sensing allows preamble-detection-based channel sensing, thereby increasing detection sensitivity to -82dBm [25] and mitigates interference from low-moderate powered transmissions for both LTE and WiFi. Channel reservation (via the CTS-to-Self) prevents WiFi–LTE and LTE–LTE interference, even during the un-intentional transmit-gaps that arise during LTE transmissions. Albeit a simple solution at the outset, instrumenting this in reality faces several obstacles, arising from the disparity that remains between WiFi and LTE in their transmission procedures.

(i) Collision Management. Unlike WiFi, LTE leverages user diversity by transmitting to multiple UEs in the same frame through OFDMA. However, this exacerbates the hidden terminal problem and results in varying interference/collision levels at the UEs (some UEs see collisions while others do not) in the same sub-frame. Managing collisions is no longer straightforward either, as the eNodeB contends on behalf of multiple clients with varying interference levels at the same time. To address this, TALOS uses a simple, yet effective A2TS-aware scheduling policy to manage collisions and minimize the impact of interference from hidden terminals in a multi-user transmission setup.

(ii) Fairness. TALOS relies on the WiFi sensing module (CSMA mechanisms) to contend fairly for channel access. However, WiFi contention window size parameters are appropriately chosen based on typical one-to-one WiFi transmissions. Hence, using the same contention parameters for an extended, one-to-many client transmission in LTE will result in starvation of the WiFi devices. TALOS overcomes this challenge with an A2TS-aware back-off mechanism that is designed to maintain fair sharing of the channel with WiFi in the presence of OFDMA transmissions.

(iii) Efficiency. While LTE's frames carry appreciable control overhead, it is heavily optimized to deliver high spectral efficiencies by relying on synchronous, always-on transmissions in the licensed spectrum. This advantage is diminished when operating at shorter time scales (milliseconds) in the unlicensed spectrum, resulting in a loss in its efficiency. While this is the price LTE needs to pay for a fair co-existence with WiFi, it is unnecessary when there is little or no WiFi activity in the channel (as is the case in some DFS 5GHz channels). In such scenarios, it is desirable for contending eNodeBs to reserve and operate the channel for longer time periods. However, employing a WiFi-based sensing module limits channel reservation to $8ms$ (due to NAV and other practical limitations), which is not sufficient to boost LTE's spectral efficiency. Hence, in the absence of WiFi, TALOS employs a novel *reservation extension*

scheme that allows it to extend the current channel reservation without requiring the LTE interface to stop its on-going transmission.

Realizing TALOS. A key feature of TALOS is that it requires no change to WiFi and LTE. The burden of A2TS co-existence rests entirely with LTE base stations, thus making TALOS immediately compatible with the multitude of LTE-U and LTE-LAA UEs that will soon come into existence. However, there are no LTE devices (eNodeBs and UEs) available yet that are capable of performing LAA-LTE as the standard is still being finalized. Hence, our current prototype realizes TALOS by mimicking the operation of LTE-U using existing commercial LTE eNodeBs and UEs, and off-the-shelf WiFi hardware, together with a novel RF bridge that can replicate WiFi signals into the LTE band, and vice versa. Results from our unlicensed LTE testbed demonstrate that TALOS's mechanisms can deliver both fairness and efficiency to WiFi and LTE in the unlicensed specrtrum.

Contributions. We make two key contributions in this paper: (1) a detailed measurement study to unravel the critical challenges in WiFi and LTE co-existence; and (2) design and implementation of TALOS — a novel, efficient and fair co-existence mechanism for LTE and WiFi in unlicensed spectrum, while requiring no modifications to the WiFi and LTE protocols. Our inferences and mechanisms can contribute to a better understanding of the challenges facing LTE-WiFi co-existence, thereby helping researchers and engineers design and incorporate better co-existence mechanisms into the standard as it evolves, while providing a level playing field in the unlicensed spectrum.

The rest of chapter is organized as follows: we provide the necessary background in Sec. 2.2 and present detailed LTE/WiFi measurements in Sec. 2.4. We describe TALOS in Sec. 2.5 and evaluate it in Sec. 2.6. We discuss related work in Sec. 2.7 and conclude in Sec. 2.8.

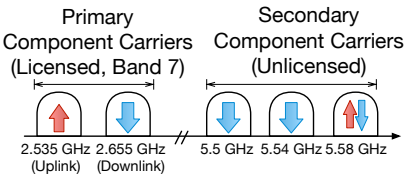


Figure 2.1: An example of LAA-LTE operation.

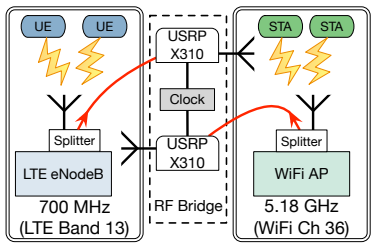


Figure 2.2: RF bridging in the LTE-WiFi testbed.

2.2 Background

2.2.1 LTE PHY Frame Structure

Unlike WiFi, LTE transmissions are based on OFDMA; and all uplink and downlink resource allocations are centralized at the LTE eNodeB (BS). The eNodeB divides spectrum resources along time and frequency into frames spanning $10ms$, with each frame further divided into ten $1ms$ subframes (see Fig. 2.3). Every subframe is then partitioned into a grid of *resource elements*, where each resource element spans a single OFDMA subcarrier (15 KHz), over the duration of a OFDMA symbol ($66.7\mu s$).

An LTE scheduler allocates resources on the granularity of a *resource block* (RB), which corresponds to twelve OFDMA subcarriers (frequency) with a duration of seven OFDMA symbols (time). The RBs are then divided into several types of channels: *(a) control channels* that are used for exchanging control information (e.g., resource assignments) between the eNodeB and the UEs; *(b) data channels* that carry data payloads; *(c) reference signals* that are are used by the UEs synchronization and data decoding. The LTE scheduler does not necessarily fill all resource elements in the control and data channels. For example, if there is only a small amount of payload data in a particular subframe, the unused data RBs are simply left blank. On the other hand, reference signals are always transmitted, even if there is no data and control in the subframe.

LTE can aggregate up to five distinct spectrum bands (component carriers, CCs)

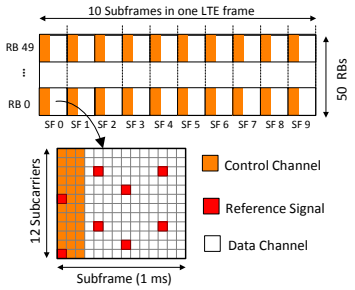


Figure 2.3: 10–MHz downlink LTE frame.

Figure 2.4: eNodeB and UE used in the LTE testbed.

into a single composite channel. Each CC is used for either uplink or downlink traffic, and can be centered in either the unlicensed or licensed spectrum. Fig 2.1 shows an example of an LTE network that uses two primary CCs on LTE Band 7[2], along with two additional downlink CCs and one uplink CC in the 5 GHz unlicensed ISM spectrum. The three downlink CCs are aggregated and used for transmission to each UE. Similarly, each UE transmits to the eNodeB using the two uplink CCs concurrently.

2.2.2 Where Does TALOS Fit In?

TALOS aims to bring LTE networks into the WiFi space using its A2TS protocol design. TALOS is designed to be fully compatible with LTE, and hence compatible with the current industry interest in LTE-U and LAA-LTE protocols [26]. However, LAA-LTE devices do not yet exist while LTE-U is compatible with the existing LTE framework. Hence, our current realization of TALOS is with respect to LTE-U in downlink-only unlicensed channels. We emphasize that TALOS design elements and its A2TS protocol are equally applicable to LAA-LTE.

2.3 Unlicensed LTE Testbed

2.3.1 Overview

There are no commercially available eNodeBs and UEs that operate in the unlicensed spectrum. Nevertheless, our LTE testbed achieves unified WiFi and LTE interaction through a novel bi-directional RF bridge between the two networks. This bi-directional RF bridge sends LTE signals into the WiFi channel, and vice versa, all with negligible forwarding latency. Hence, we can analyze the dynamic interactions between WiFi and LTE when operating in the same frequency bands.

Fig. 2.2 illustrates our testbed architecture. Our testbed has three main components: the LTE network, the WiFi network and the RF bridge. The LTE and WiFi networks operate in separate frequencies — Band 13 and Channel 36 for LTE and WiFi networks respectively.

2.3.2 LTE Network

We use a commercial SISO LTE Release 8 small-cell eNodeB with a transmit power of up to 1W, along with up to five off-the-shelf UE Pantech USB dongles. Fig. 2.4 shows a picture of the LTE equipment. The small-cell eNodeB uses 10MHz FDD LTE uplink and downlink channels in LTE Band 13. We attach a splitter to the active antenna port on the eNodeB: one output port of the splitter is connected to an 3 dBi omni-antenna, while the other port is connected to the RF bridge via an RF cable.

2.3.3 WiFi Network

We use a 20MHz 802.11a SISO WiFi network in the 5GHz band. We make use of two different WiFi network setups. Only one setup is used at any time.

[2] `https://en.wikipedia.org/wiki/LTE_frequency_bands`

WARPv3. LTE measurements in Sec. 2.4 are obtained using WARPv3 and the 802.11 Reference Design v1.2 [27]. The WARPv3 provides more detailed performance statistics than can be obtained from commercial WiFi devices. One of the WARPv3 boards is configured as an AP, while up to five other WARPv3 boards are used as STAs, *i.e.*, WiFi clients. We connect a splitter to the active antenna port of the WARPv3 AP. One of the splitter outputs is connected to a 3dBi WiFi omni-antenna, while the other port is connected to the RF bridge via a RF cable.

TP-Link WiFi. We also employ an alternative WiFi network built using off-the-shelf TP-Link WiFi devices. We use the TP-Link TL-WDN4800 PCIe WiFi card, installed on a desktop PC, as an AP and the other TL-WN821N WiFi dongles as STAs. A splitter is connected to the active antenna port on the AP. This is a drop-in replacement for the WARPv3 network. We use the TP-Link platform to to demonstrate that Talos is feasible over unmodified WiFi hardware.

2.3.4 RF Bridge

I/Q forwarding. The RF bridge uses two USRP X310 devices to forward LTE and WiFi interference between the two networks. Each USRP X310 has two UBX-120 RF daughterboards. In the LTE-to-WiFi bridge, one RF board is connected to the splitter port on the LTE eNodeB via an RF cable, while the other RF board is connected to a 3dBi omni-antenna. The LTE-to-WiFi bridge samples the 10MHz downlink LTE channel from the eNodeB at 46.08MHz. This is exactly three times the sampling rate of a 10MHz LTE channel.[3] This over-sampled data is then immediately transmitted by the other RF daughterboard into the 5GHz WiFi network. We have verified that the spectrum power characteristics (and thus, the interference behavior) of the LTE signal is maintained after bridging into the WiFi network. The bridge will induce phase offsets into the forwarded signal, but this does not affect the testbed

[3] 10MHz LTE channels are actually transmitted at 15.36MHz.

experiments as the forwarded LTE signal is not decoded on the WiFi network. We maintain frequency and time synchronization across both X310s by using a common reference clock source.

A similar design is used in the WiFi-to-LTE bridge, except that a 50MHz sampling rate (2.5x oversampling) is used instead to forward WiFi signals. We emphasize that each bridge (LTE-to-WiFi and WiFi-to-LTE) forwards signals directly obtained from the eNodeB and WiFi AP, respectively, over an RF cable. Hence, *no cyclic RF bridging occurs* — LTE signals forwarded into the WiFi network are *not* subsequently forwarded back into the LTE network.

RF Bridge vs. Analog Frequency Converter. While analog frequency converters are an alternative solution to our RF bridge, the key benefit of the latter is that it enables the forwarded LTE signals to be quickly switched on and off. The on-off switching times is on the order of microseconds, thus allowing us to emulate the fast on-off behavior of both LTE-U and LAA-LTE. Such fine-grained control is difficult with analog frequency converters without custom complex (and hence costly) control circuits.

Bridging Gain. We modify the interference power of LTE on WiFi (and vice versa), by scaling the oversampled I/Q data before transmission over the omni-antenna, and by adjusting the transmit gain of the associated bridge. The transmit power of the eNodeB and WiFi APs over their respective omni-antennas are maintained unchanged. All reported power levels are obtained using an Agilent spectrum analyzer.

2.3.5 Bridging Latency

The WiFi-to-LTE and LTE-to-WiFi bridge each has a bridging latency of $250\mu s$. This latency is an artifact of our LTE testbed, but has a very limited impact on our measurement study, where we let the LTE node transmit continusously and study its

interactions with a conventional WiFi node.

With respect to energy sensing, the bridging latency does not impact WiFi's capability of sensing the LTE signal (only depends on CCA) that is always on. Similarly, the transmit gaps in the LTE transmissions will manifest with a latency of 250 μs in the WiFi channel (as shown in Fig. 2.5), but do not alter WiFi's response (compared to without the latency) since the channel state perceived by WiFi is delayed in its entirety. On the other hand, WiFi's response to such transmit gaps, will be observed at LTE with a latency of 500 μs. This latency can lead to false positives and negatives respectively during measurement of WiFi interference at LTE when (a) WiFi transmission gets completed within the LTE time gap, leading to false interference detection, and (b) LTE transmission duration is small or comparable to bridging latency, leading to potentially not detecting interference. However, as we show in Sec. 2.4, typical WiFi transmission durations (eg. $280\mu s = 232\mu s$ DATA + $16\mu s$SIFS + $32\mu s$ ACK for 1.5KB packet at highest rate of 54 Mbps) are much larger than the median time gaps (under 140 μsecs) observed in LTE transmissions. This coupled with LTE transmissions operating at several millisec time-scales, eliminates such scenarios, thereby allowing us to understand the interactions between LTE and WiFi accurately.

2.3.5.1 Small-Scale Impact

LTE-to-WiFi Latency. Fig. 2.5 illustrates the impact of the $250\mu s$ latency when the downlink LTE channel is forwarded into the WiFi network. Consider, for example, the case where a time gap exists in the LTE channel. WiFi will infer a clear-channel during this gap and begin its transmission. Due to the bridging latency, this gap will appear $250\mu s$ later in the WiFi channel. However, this has no impact on WiFi measurements as the performance of WiFi is dependent on the channel state, which is still preserved here albeit with a fixed delay.

WiFi-to-LTE Latency. Similarly, the WiFi-to-LTE bridge incurs a $250\mu s$ bridging delay. This implies that the effect of WiFi on LTE sees a total delay of $500\mu s$. However, this delay has limited effect on the interference impact that WiFi has on LTE. Consider that a typical 1.5KB 802.11a WiFi transmission has an airtime of $280\mu s$ ($232\mu s$ DATA + $16\mu s$SIFS + $32\mu s$ ACK) at the highest rate of 54Mbps. This airtime will increase at lower bitrates and with frame aggregation. In Sec. 2.4, we show that time gaps in LTE networks are always less than or equal to $220\mu s$ with a median duration under $140\mu s$. Hence, it is highly likely a WiFi transmission that starts in the gap will interfere with the an LTE frame.

2.3.5.2 Large-Scale Impact

The impact of this latency is more significant when LTE is duty-cycled. More precisely, the forwarding latency causes a mismatch between the actual activation time of the LTE CC, and the time at which this activation is seen by WiFi. This delay results in additional and unwarranted LTE-WiFi collisions when the LTE CC is enabled. Similar effects have been observed in the impact of the USRP communication delays on WiFi CSMA behavior [28, 29]. However, this impact on our experiments is insignificant due to two reasons.

First, in our study of LTE time-gaps in Sec. 2.4, we do not duty-cycle the bridged LTE signal, i.e. the bridged LTE signal is always on. Hence, the bridging latency does not result in any anomalous duty-cycling- related effects. Second, in our evaluation of Talos and other LTE protocols in Sec. 2.6, each on-time is $20ms$ and $100ms$ in duration, respectively. This is an order of magnitude larger than the WiFi frame transmission. Hence, an overwhelming majority of LTE-WiFi collisions is due to small-scale time-gaps in the LTE subframes.

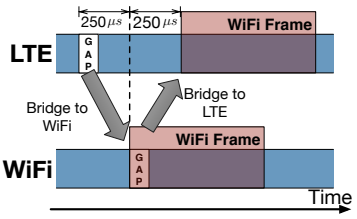


Figure 2.5: Bridging latency.

	LTE	WiFi
Tx. Power	-13dBm	-14dBm
Modulation	4/16/64 QAM	16 QAM
Pkt. Size	960 Bytes	1400 Bytes

Figure 2.6: Testbed parameters.

2.4 A Real-World Study of Unlicensed LTE

To understand the performance of LTE in unlicensed channels, it is important to study the PHY-layer behavior of LTE in the WiFi channel. We conduct two experiments: (a) we use WARPv3 devices to highlight the effects of the CCA threshold, and (b) we study the characteristics of time-gaps using our LTE testbed. Note that the bridged LTE signal is always-on (i.e. not duty-cycled) so that we can focus on the impact of small-scale time-gaps in LTE.

2.4.1 Experiment Methodology

Traffic Generation. We use `iperf` to generate constant bitrate traffic in the LTE and WiFi networks. The bitrate of the `iperf` stream is varied to induce different downlink network loads on the LTE and WiFi networks.

Transmit Load vs. Receive Throughput. The transmit load refers to the offered bitrate of the transmitted stream over the wireless channel. The received throughput is the bitrate received at the destination. The received throughput can be lower than the transmit load due to various wireless channel effects.

Experiment Parameters. Figure 2.6 shows the basic parameters used in the LTE and WiFi networks. The 802.11a WiFi CCA threshold is fixed at the standard -62 dBm [25]. We define Δ_L to be the ratio of the mean LTE signal power to the CCA threshold at the AP. Note that due to the large variations in the LTE signal, the instantaneous LTE power can be higher than the CCA threshold even if $\Delta_L < 0$.

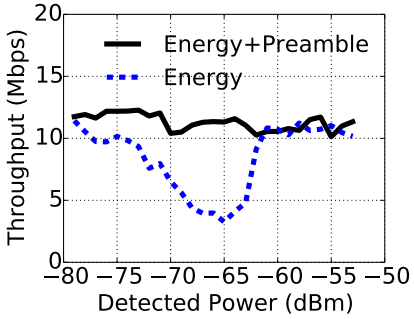


Figure 2.7: Throughput under LBT-like CCA.

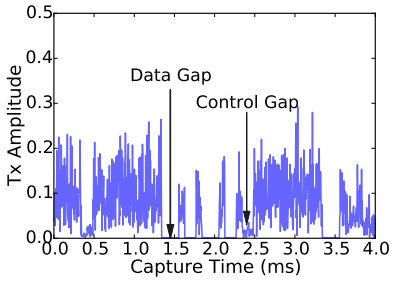


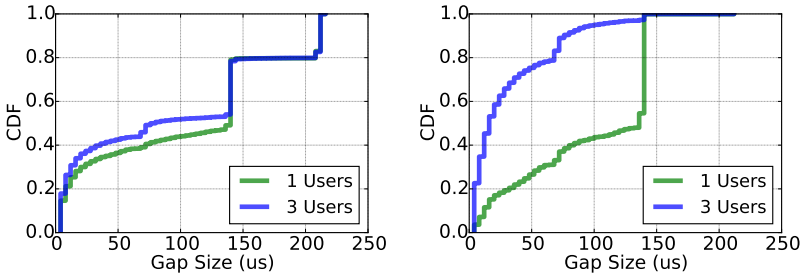
Figure 2.8: Time-domain snapshot of LTE signal.

We also define $SIR_L^{\star}$ to be the ratio of the received LTE signal to WiFi interference power at each UE.

2.4.2 Impact of CCA Thresholds

There are no LTE devices available that can perform LBT since the LAA-LTE standard is yet to be finalized. Instead, we analyze the impact of LBT CCA thresholds using two Tx/Rx pairs of WARPv3 WiFi devices. The devices are placed at multiple locations over an office floor to obtain different received signal levels. At each location, we measure the average throughput when only WiFi CCA is enabled (preamble detection disabled) and when both CCA and preamble detection are active. The CCA threshold used is -62dBm.

Fig. 2.7 shows the throughput of one of the Tx/Rx pairs, w.r.t. the received interfering signal power. Observe that if preamble detection is employed, this pair has a throughput of about 12 Mbps. However, when only energy detection is used, this transmitter cannot detect on-going interfering transmissions, resulting in up to a 66.7% reduction in throughput (to about 4Mbps). The inability to detect such interfering transmissions is even more detrimental to LTE, where multiple (tens of) UEs are scheduled in each subframe. The result also clearly indicates (see region with lower received power) that decreasing the CCA threshold (eg. -75 dBm) will



(a) 5 Mbps LTE Tx load (b) 20 Mbps LTE Tx load

Figure 2.9: LTE gap size distribution for different LTE loads.

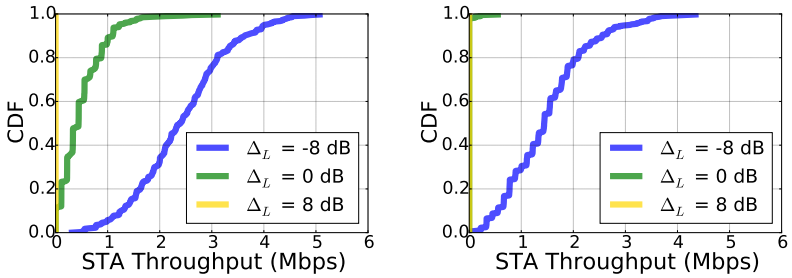
not address the problem either, as this would prevent two legitimate, non-interfering LTE and WiFi transmissions from operating concurrently, thereby reducing channel utilization. Indeed, the role of energy sensing (CCA thresholds) is only to serve as a macro filter for detecting high powered transmissions in the channel. It needs to be supplemented with additional protocol-specific detection mechanisms (eg. preamble detection) that identify legitimate low-moderate powererd transmissions (interference) without jeopardizing channel utilization. Such a mechanism is absent in the case of LTE, which is critical for an efficient LTE-WiFi coexistence.

2.4.3 Characteristics of LTE Time Gaps

We highlight the RF characteristics of LTE I/Q signal measured at the LTE-to-WiFi bridge during normal LTE operation. We follow the established practice and consider the channel to be occupied if its measured signal energy is 10 dB above the noise floor [30]. Our LTE network achieves a peak rate of around 30 Mbps over a 10 MHz downlink channel [4].

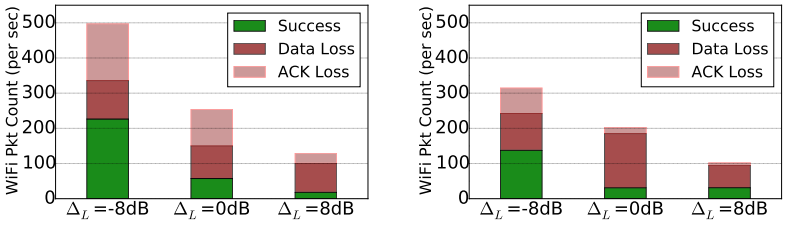
Fig. 2.8 shows an example of an LTE power profile that highlights the time-gaps that arise in both the control and data RBs of an LTE transmission. The presence and distribution of these gaps vary depending on the LTE transmit load, and are not due to any intentional idle periods injected by the eNodeB.

[4]Close to the theoretical bitrate of a Release 8 LTE network operating in SISO mode



(a) 5 Mbps LTE Load (b) 20 Mbps LTE Load

Figure 2.10: Distribution of WiFi throughput at different LTE Loads. The WiFi load is 20Mbps.



(a) LTE Load = 5Mbps (b) LTE Load = 20Mbps

Figure 2.11: WiFi packet loss statistics.

Fig. 2.9 shows the distributions of the time-domain gap sizes as we vary both the total load on the LTE channel, and the number of UEs. Note that at 20Mbps LTE load, all data RBs are utilized. The time-gaps shown here are due to the gaps in control RBs only. We note that WiFi uses a $4\mu s$ window within a $9\mu s$ time slot for energy detection [31], and can have frame transmission time of several milliseconds [32]. Observe that even under a heavily loaded LTE channel, almost all the gaps are larger than the WiFi timeslot. This means that (a) WiFi is highly likely to misinterpret the LTE gaps as transmit opportunities and (b) an erroneous WiFi transmission is likely to result in a collision. Also observe that the size of the time-gaps is even larger under low LTE load of 5Mbps.

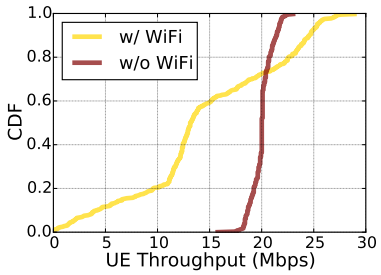


Figure 2.12: Impact of 20Mbps WiFi on UE throughput with LTE load of 20Mbps.

2.4.4 Impact of LTE Gaps

Impact of LTE on WiFi. Fig. 2.10 illustrates the impact of LTE on WiFi. We fix $SIR_L^\star$ at the UE at 10dB and the given WiFi transmit load at 20Mbps. As expected, WiFi achieves higher throughput when the LTE load is lower. More importantly, when the average LTE signal is higher than the CCA threshold ($\Delta_L \geq 0$), we see that WiFi's performance degrades with increasing LTE load. This indicates that although WiFi is able to detect LTE, the transmit gaps in LTE transmission result in interference between WiFi and LTE, whose impact increases with higher LTE load. This is further illustrated in Fig. 2.11, where statistics on the number of transmitted WiFi frames indicates that WiFi generates several packets during LTE's transmission, thereby resulting in a larger fraction of losses (50–75%). This impact increases as the LTE load increases.

Impact of WiFi on LTE. Contrary to several existing measurements [7, 13, 14], we find that the LTE transmission gaps can also result in WiFi interfering with LTE severely. Fig. 2.12 shows the throughput achieved by an LTE UE in the presence of WiFi interferers. Both the WiFi and LTE transmit loads are 20Mbps. It can be clearly seen that when going from isolated LTE operation to one where WiFi can detect LTE (and potentially back-off), the received throughput at the UE can be severely impacted — the median throughput decreases by 40% from 20 to 12 Mbps. This again can be attributed to the numerous transmit gaps that trigger WiFi, thereby

resulting in appreciable performance degradation for LTE.

2.4.5 Can We Eliminate Transmit Gaps?

At a first glance, these gaps seem to arise due to pre-determined control and data allocations that go un-used. Hence, it appears that these gaps can be easily eliminated in two ways: (a) by transmitting random data in all unused/null control and data RBs, and (b) by removing the control portion of the sub-frame completely from the unlicensed channels (keeping them only in licensed channels). However, these approaches suffer several limitations.

(Control) Limited User Multiplexing. Allocation of data transmission resources are indicated to UEs through information in the control part of the frame. When the control part of LTE's transmission is eliminated in the unlicensed channels, resource allocation to UEs in those channels have to be conveyed through the control part of the licensed channels. This will significantly limit the number of UEs that can be scheduled in all the channels, thereby reducing user diversity gains and leading to increased latencies. This affects our ability to accomodate the rapidly growing class of low bandwidth UEs, such as Internet of Things (IoT) devices [33] as well.

(Control) Listen-Before-Talk. Another issue arises in the presence of LBT. While LTE transmissions in licensed channels always begin and end on sub-frame boundaries, this is not the case with unlicensed channels, where LBT is employed. In order to maximize channel utilization, current LAA-LTE proposals allow LTE to transmit immediately after a successful LBT process [34], thereby resulting in partial sub-frame transmissions, unlike in licensed channels. Given the non-deterministic nature of available resources in such partial sub-frames, these cannot be allocated apriori in the control part of licensed channels and needs to be conveyed in the control part of the partial sub-frame (unlicensed channel) itself. This prevents the elimination of control RBs in unlicensed LTE transmissions.

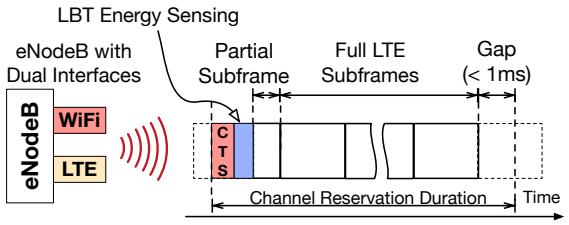


Figure 2.13: Activating the LTE-U component carrier.

(Data) Non-Uniform Interference. It is challenging to ensure that all data RBs are filled in the face of traffic burstiness and client heterogeneity [35, 36]. However, a more significant challenge comes from the fact that LAA-LTE supports both uplink and downlink transmissions in the same frequency band when deployed in TDD (time divisioned dupled) mode. Even if we utilize all RBs in the downlink channel, we cannot prevent gaps during the uplink subframes. Each uplink UE will only transmit on the set of RBs that has been allocated to it. Hence, the aggregate uplink interference power at any WiFi node is time-varying and dependent on the spatial topology of the UEs. Hence, the eNodeB cannot completely prevent transmission gaps at any WiFi/LTE device.

2.5 TALOS: LTE-WiFi Co-Existence

Our coexistence solution, TALOS is built on the observation that it is essential to bridge the fundamental difference between the channel access modes of WiFi (distributed and asynchronous) and LTE (centralized and synchronous) to address the challenges facing their co-existence. TALOS realizes this by enabling the LTE node (base station) *access asynchronously* but *transmit synchronously* (*A2TS*) in the unlicensed spectrum. The WiFi nodes are unchanged. The proposed, hybrid mode of LTE operation (A2TS) allows for the LTE nodes to contend for channel access asynchronously in a manner that is compatible and fair to other WiFi node, while remaining compliant with the synchronous nature of LTE transmissions.

2.5.1 Solution Overview

For each unlicensed channel that a TALOS node accesses, it employs a supplementary WiFi access module on the same channel only for the purpose of channel contention/access. Once the WiFi module gains access to the channel, it reserves the channel using a CTS-to-self frame with an appropriate network allocation vector (NAV) value. The WiFi module then relinquishes the channel to the LTE stack on the node, which then executes LTE transmissions to its UEs on the unlicensed channel.

Fig. 2.13 illustrates the operation of TALOS. Observe that LTE transmissions in the licensed channel can begin from a partial subframe position, but must end on a subframe boundary. Hence, there can be up to a $1ms$ of additional unused reservation time at the end of the reservation duration.

Augmenting LTE with WiFi based sensing and notification, allows TALOS to protect LTE transmission gaps as well as help WiFi/LTE devices detect interference with high sensitivity. Albeit a simple idea at the outset, delivering its benefits in practice requires TALOS to overcome three challenges: collision management, fairness and efficiency. We next describe these challenges and how TALOS 's design helps address them.

Also, while our solution applies to both downlink and uplink LTE transmissions, we present it with respect to downlink for an easier exposition.

2.5.2 Collisions: A2TS-Aware Scheduling

Challenge: Hidden terminals in asynchronous access (e.g. WiFi) networks can result in unexpected collisions at the receiver. Such problems are an even bigger challenge for LTE as multiple UEs can be concurrently scheduled on the downlink channel. If we simply approximate the channel state of all UEs with that at the transmitter, the inferred channel status will be highly inaccurate. The key reason is that only the WiFi nodes that are within CTS-decoding range of the eNodeB will

suspend transmissions in response to the CTS NAV information. Hence, the further away a WiFi node is from the eNodeB, the more likely it will interfere with ongoing downlink transmissions. As a result, different UEs will experience different levels of WiFi interference — those closer to the eNodeB are less susceptible to interference than those at the edge of the CTS coverage range.

2.5.2.1 A2TS-Aware Scheduling

Talos addresses this challenge through intelligent scheduling of UEs in unlicensed carriers. Talos identifies UEs that are more susceptible to interference by measuring their LTE frame loss rates; Talos leverages access to licensed channels by scheduling (moving) the UEs with consistently high loss rates to the licensed channels, where they will not be subject to WiFi or LTE (from another operator) interference. This allows for interference-prone (potentially cell-edge) UEs to receive adequate protection in licensed carriers, while alleviating the interference they cause to other WiFi devices (when LTE uplink also operates in unlicensed carrier). On the other hand, those UEs that are less sensitive to interference continue to be scheduled on the unlicensed carriers.

Data to be transmitted on the unlicensed downlink carrier is partitioned into one or more transport blocks and multiple transport blocks are transmitted in each subframe. The UE ACKs or NACKs each transport block separately, and the eNodeB tracks the acknowledgements of these transport blocks to determine the average data loss rate of each UE in a reservation period.

Talos leverages existing LTE schedulers, which are executed before the transmission of every subframe. Before each subframe transmission, Talos adds a pre-scheduling step that sorts the set of UEs to be scheduled according to their frame loss rates. Subframe resources in the unlicensed channel are then assigned to the UEs in increasing order of their loss rates. The UEs that remain after all RBs are

assigned are the predominantly interference-prone UEs. These are then scheduled in the licensed channels.

2.5.3 Fairness: A2TS-Aware Contention

Challenge: The CSMA behavior in WiFi is influenced by the contention window and backoff policies. However, WiFi's CSMA policies cannot be directly applied to TALOS for two reasons. First, the WiFi channel access probability is constrained by the maximum and minimum contention window sizes, $W_{\max}$ and $W_{\min}$, respectively.

Existing WiFi contention window ranges are appropriate for typical WiFi frame sizes. However, each TALOS reservation duration may be appreciably longer (1-10 ms for LAA-LTE, tens of ms for LTE-U) than a WiFi frame. If we keep the channel access probability unchanged, TALOS will occupy an unfair share of the channel, leading to starvation of the WiFi nodes.

Second, LTE uses a one-to-many transmission model. The probability that at least one UE encounters a collision from WiFi is now significantly higher. Hence, if TALOS backs off even if one UE experiences a collision, then it is very likely to experience a high incidence of backoffs and will thus be starved of throughput. On the other hand, if TALOS only backs off when all UEs experience collisions, then it is likely to be overly-aggressive in contending for the channel, leading to unfairness for WiFi.

2.5.3.1 A2TS-Aware Contention

The goal of TALOS is to achieve fair channel access with WiFi. We define "fairness" as follows. Compare a network with WiFi APs/STAs and LTE eNodeBs/UEs with another network where the LTE eNodeBs/UEs are replaced with WiFi APs and STAs, respectively. If the WiFi APs/STAs from the first network obtains the same throughput as in the second network, then we consider the LTE share of the channel

to be fair. To enable such fair sharing, TALOS modulates its contention mechanism along two fronts as follows.

First, TALOS linearly scales the contention window sizes. Let the reservation duration be $L\times$ the average transmit airtime of a WiFi frame, the latter being tracked through the WiFi access module. TALOS sets its contention window range to $[LW_{\min}, LW_{\max}]$. The random backoff interval, b, is thus increased to $b \in [0,\ \overline{W}]$ where $\overline{W} \in [LW_{\min}, LW_{\max}]$. We show both theoretically in Sec. 2.9 and using experiments in Sec. 2.6 that this linear scaling maintains throughput fairness to WiFi.

Second, doubling the contention window upon collisions is no longer appropriate when UEs see different collision states in the same transmission. Hence, to accurately capture the impact of collisions on multiple UEs concurrently, TALOS increases $\overline{W}$ proportional to the subframe collision rate. The eNodeB maintains a Hybrid-ARQ (i.e. MAC-layer ack) counter (value between 0 and 3) for each data packet (i.e. LTE transport block) that is scheduled during the on-period. Multiple data packets can be scheduled over the same reservation duration for different UEs. The eNodeB increments a HARQ counter by one upon a NACK (or lack of ACK) from the UE. TALOS computes the average HARQ value, $\overline{H}$, of all transmitted data packets in the current reservation period.

If $\overline{H}$ is larger than the average HARQ from the previous period, the backoff window limit for the next reservation duration is increased to $\hat{W} = \min\{(1+\frac{1}{3}\overline{H})\overline{W}, LW_{max}\}$. The $(1 + \frac{1}{3}\overline{H})$ scaling factor ensures that the backoff increment is proportional to the loss rates of the LTE transport blocks across multiple UEs. Otherwise, we set $\hat{W} = LW_{min}$. TALOS then randomly selects $b \in [0, \hat{W}]$.

2.5.4 Efficiency: Extended Channel Access

Challenge: Unlike licensed channels, the distributed contention overhead (backoffs, collisions, etc.) reduces LTE's spectral efficiency in the unlicensed channels.

While this is inevitable to ensure fair sharing with WiFi, it can be avoided during WiFi's absence by allowing LTE transmissions over longer time scales. This requires TALOS to (i) first detect presence/absence of WiFi activity, then (ii), during WiFi's absence, enable channel reservation for longer time periods beyond those allowed by WiFi NAV durations, and finally (iii) disable the use of contention window scaling (Sec. 2.5.3).

2.5.4.1 Detection of WiFi Devices

TALOS uses a CTS-to-Self WiFi frame to reserve the channel. A CTS-to-Self frame that originates from a TALOS node is made to carry a "LTE-source" flag in its payload. Hence, TALOS monitors the WiFi traffic on the unlicensed channel to track the the "LTE-source" flag. If any WiFi packet is detected without the flag, this would indicate the presence of a WiFi source. Hence, TALOS will not use the reservation extension protocol in this case, so as to maintain fairness to WiFi.

2.5.4.2 Extended Channel Access for LTE

Limited by the NAV durations, TALOS uses successive CTS-to-Self frames to continue extending the current channel reservation on the fly. However, the additional CTS-to-Self packets (beyond the first one) from the WiFi access module will interfere with the on-going LTE transmission on the same node. One possible option is to send the CTS packet precisely at the end of the LTE transmission. However, this will require tight PHY-layer synchronization between the two interfaces, which requires PHY-layer changes.

Alternatively, since LTE transmissions must end on sub-frame boundaries, the data resources in the last (potentially partial) sub-frame of the current reservation will be left empty. Hence, TALOS leverages this to extend the reservation without causing interference to its on-going LTE transmission as follows.

To extend a current reservation, TALOS sets the contention window size of the WiFi access module to zero so that it does not backoff when presented with a chance to access the channel. When the WiFi module detects a transmission opportunity during the empty data RBs of the last LTE subframe, it immediately sends three CTS-to-Self packets back-to-back, separated by SIFS duration. TALOS takes advantage of the fact that the smallest gap size between reference signals in the empty downlink LTE subframe is $133.4\mu s$, while the airtime duration of a WiFi CTS frame and SIFS is $48\mu s$ and $16\mu s$ respectively. Hence, a transmitted CTS frame can easily fit completely within the time gaps. By transmitting three CTS-to-Self packets consecutively, TALOS ensures that at least one of them is not interfered by the reference/control signals from the LTE transmission (without synchronizing the WiFi module and LTE interface) and can be received correctly at the WiFi STAs. After the new CTS-to-Self frames have been sent, TALOS continues sending downlink subframes on the same channel un-interrupted. In order to guard against interference to undetected WiFi devices, we limit the maximum total channel reservation to $100ms$ (the typical WiFi beacon interval). This prevent inadvertent WiFi disconnections due to the loss of multiple beacon signals.

Remarks.

(1) While the 802.11 standard allows for channel NAV durations of up to $32ms$, off-the-shelf devices may only allow for a shorter limit as a precaution against NAV flooding attacks [37]. For example, the 802.11ac standard allows up to $5.5ms$ of aggregated frame transmissions [38], and valid 802.11ac NAV values, even accounting for SIFS, DIFS, and other delays, should not exceed this by an excessive amount. This "shortcoming" is easily accomodated in TALOS by utilizing the reservation mechanism more frequently to obtain the desired overall channel reservation time.

(2) The reservation extension mechanism is useful for LTE-LTE coexistence and finds applications in other unlicensed bands such as 3.5 GHz, where WiFi is absent.

2.6 Experimental Evaluation

2.6.1 Implementation

TALOS. Since LTE nodes that operate in the unlicensed spectrum are not available yet, we build a TALOS node by augmenting a commercial LTE node with an off-the-shelf WiFi interface, the latter serving the role of the WiFi channel access module. The WiFi interface is a TP-LINK wireless PCIe adapter built on the Atheros AR9380 chipset. The coordination latency between the LTE and WiFi interfaces on a TALOS node are negligible and hence allow for realization of its mechanisms in real-time. However, with the commercial LTE interface not capable of short time scale transmissions, we execute channel access in TALOS through activation/deactivation of the LTE interface itself at transmission time scales of 20 ms (similar to LTE-U). While these transmission time scales will reduce to few ms in future with LAA-LTE availability, our current set-up is sufficient to validate and evaluate the benefits of the various mechanisms in TALOS .

CTS-to-Self. We take advantage of the built-in CTS-to-self support in the `ath9k` WiFi linux driver. Recall that TALOS scales the contention window size to account for the relatively long channel reservation duration (several ms). In WiFi, the default value of $W_{\min}$ is 15 time slots. However, since the largest contention window size supported by `ath9k` is 1023, the TALOS reservation duration can only be at most $1023/15 \approx 68$ times the WiFi frame duration. If the average WiFi transmission spans, say $280\mu s$, which is the duration of a 1.5KB 802.11a frame at 54Mbps, then the channel reservation duration of TALOS would be limited to $19.1ms$ in our testbed.

Fairness and Collisions. We fix $W_{\min} = W_{\max} = 1023$, and the channel reservation duration to be $20ms$ for all our LTE experiments. This is achieved by using the `hostapd` wireless utility to set the contention windows, and using a modified `ath9k` driver with hard-coded NAV duration. No expoential back-off is performed in TALOS

after any collisions, but the contention window is appropriately adapted (Sec. 2.5).

Unlicensed Carrier Activation and Deactivation. We emulate channel activation and deactivation by switching on and off the bidirectional RF bridge in a coordinated fashion. When the `ath9k` sends a CTS packet after channel contention, TALOS activates its unlicensed channel and the bidirectional RF bridge begins forwarding I/Q signals between the LTE and WiFi networks. When TALOS deactivates its unlicensed channel, the RF bridge will cease forwarding I/Q signals. At this point, the `ath9k` interface will resume its CTS contention. Due to limitations of our eNodeB, we only study the performance of TALOS over a single unlicensed channel. Carrier aggregation with a licensed channel is currently not supported.

2.6.2 Experiment Setup

We evaluate TALOS in a real-world setting using our LTE-WiFi testbed described earlier, while replacing the WARP nodes with commercial TP-LINK WiFi devices. The use of the RF bridge to bring LTE and WiFi signals into each others' band incurs a 250 μs latency either way. This bridging latency causes a mismatch between the actual start time of the LTE (WiFi) transmission, and the time at which this LTE (WiFi) transmission is seen by WiFi (LTE). This delay results in additional LTE-WiFi collisions when the LTE (WiFi) transmission is started and before it is detected by WiFi (LTE). Similar effects have been observed in the impact of the USRP communication delays on WiFi CSMA behavior [28, 29]. Fortunately, this impact is mitigated due to TALOS 's operational time scale of $20ms$, which is an order-of-magnitude larger than the typical WiFi frame transmission duration. Hence, an overwhelming majority of LTE-WiFi interactions will be due to steady state LTE and WiFi characteristics. The impact of the bridging latency is thus negligible. Note that in response to the larger channel reservation durations, the contention windows will be scaled accordingly, as described in Sec. 2.5.3. The experimental results and insights obtained from

our experiments will thus hold for future LAA-LTE-to-WiFi co-existence characteristics. We also augment our evaluations with NS-3 [39] simulations, to compensate for the lack of fine-grained control over the commercial eNodeB scheduler.

Other Protocols (Duty Cycled LTE): We compare TALOS with duty-cycled LTE (eg. Qualcomm's LTE-U [23]). Each on-period has a duration of $100ms$. We vary the duration of the off-period to achieve duty-cycles of 100% (no off period), 50% and 33%. Note that with static duty-cycling, the optimizations in TALOS and CTS-to-Self (for reserving the channel) are not employed. However, the RF bridge is still used to emulate the activation and deactivation of the unlicensed channel.

2.6.3 Performance Measurement

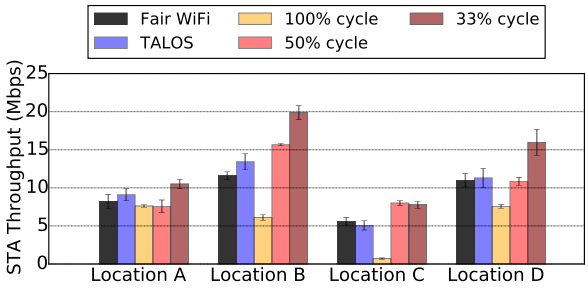
In a given topology, we measure the performance of TALOS, duty-cycled LTE and the WiFi network.

LTE. In order to obtain the performance of TALOS and duty-cycled LTE over the unlicensed spectrum, we keep track of the LTE frames transmitted during the carrier activation period in `tcpdump`. We use only these frames to determine the performance of TALOS and the duty-cycling protocols.

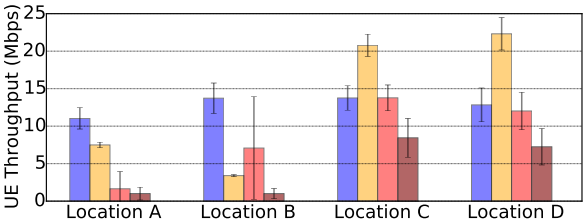
WiFi. We directly measure the performance of WiFi over the same unlicensed spectrum. Note that since the WiFi devices actually suspend and resume transmissions in response to the CTS-to-Self messages, the performance of the WiFi network can be measured directly, without any additional filtering steps. For each topology, we also measure the fair WiFi throughput (as defined in Sec. 2.5.3.1) by replacing the eNodeB and UEs with TPLINK WiFi AP and STAs.

2.6.4 TALOS Fairness and Throughput

In this section, we demonstrate that WiFi will achieve its fair throughput under TALOS, without unnecessarily penalizing the LTE throughput. Note that only the



(a) WiFi



(b) LTE

Figure 2.14: Average throughput at different locations.

Location	A	B	C	D
LOS or NLOS	NLOS	LOS	NLOS	LOS
LTE/WiFi Signal Strength	WiFi > LTE		WiFi < LTE	
LTE/WiFi (STA)	-25dB	-20dB	30dB	25dB
LTE/CCA (AP)	-34dB	-28dB	13 dB	5dB

Table 2.1: Parameters in each scenario.

performance of LTE over the unlicensed bands is considered. Licensed LTE transmissions are ignored. We first consider a simple testbed setup using one UE and one STA, together with an LTE eNodeB and a WiFi AP. From our measurements, we select four different locations (labeled A to D) for the WiFi AP that represent different line-of-sight (LOS) and non-line-of-sight (NLOS) network conditions.

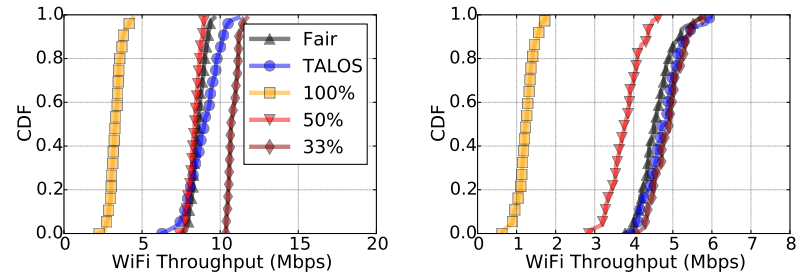
The topology parameters are summarized in Table 2.1 and are representative of the signal strength variations that can be found in a real-world LTE-U network. "LTE/WiFi(STA)" denotes the ratio of the LTE to WiFi signal energy seen at the WiFi STA, while "LTE/CCA(AP)" denotes the ratio of the LTE power to CCA threshold as seen at the AP. Since there is a single LTE and single WiFi link, we expect that the LTE and WiFi will gain equal air-time to the channel. The maximum throughput of the LTE network at all locations is 30 Mbps. Hence, under fair-sharing

of the channel, LTE should achieve a throughput close to 15 Mbps.

Duty-cycling is highly inefficient. Fig. 2.14 shows the mean throughput at the STAs and UEs across all four locations. Observe that under simple duty-cycling, WiFi can achieve, and in some cases even exceed, its fair-throughput. While this is beneficial to WiFi, it comes at a significant cost to LTE. Consider location A as an example. Even though a 50% duty-cycle achieves a fair WiFi throughput, LTE throughput drops to a mere 1 Mbps, which is significantly lower than the expected 15 Mbps. This poor LTE performance is due to significant interference from WiFi. WiFi transmissions are not suppressed during the LTE on-periods, arising from the failure of CCA detection as well as time gaps in the LTE signal.

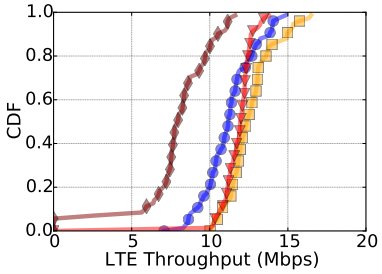
Talos *achieves fair-sharing efficiently.* When Talos is employed at any location, the WiFi network can achieve a throughput that is within 1% (e.g. location D) of its fair throughput. Furthermore, this fairness does not come at a penalty to LTE. For example, in location A, even though Talos achieves approximately the same WiFi throughput as a 50% LTE duty-cycle, it does so with a LTE throughput that is over 10× greater (11 Mbps from 1 Mbps). In all other locations, the mean LTE throughput is within only 25% of the expected 15 Mbps ideal throughput. We emphasize that such adaptation is inherent to Talos and is achieved without any network calibration. Thus, the A2TS contention mechanism of Talos converges to this fair state regardless of the location and configuration of the LTE and WiFi networks.

Optimal duty-cycle is network-specific. We then extend the network to support up to 2 WiFi APs, with 4 STAs each. Figs. 2.15(a) and 2.15(b) shows the sum-throughput of WiFi when one and two WiFi APs are present, respectively. Observe that the optimal duty-cycle in each of these cases (50% and 33% respectively) depends on the number of APs present. This information is typically not explicitly conveyed to the eNodeB. Hence, we will need to adapt the duty-cycle for each network over long-periods in order to converge to the optimal point. Such adaptation has to per-

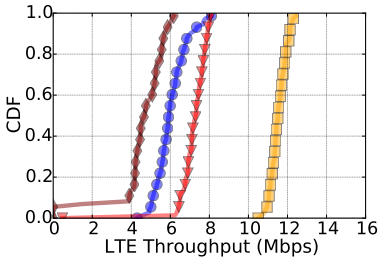


(a) WiFi (1×AP)

(b) WiFi (2×APs)



(c) LTE (1×AP)



(d) LTE (2×APs)

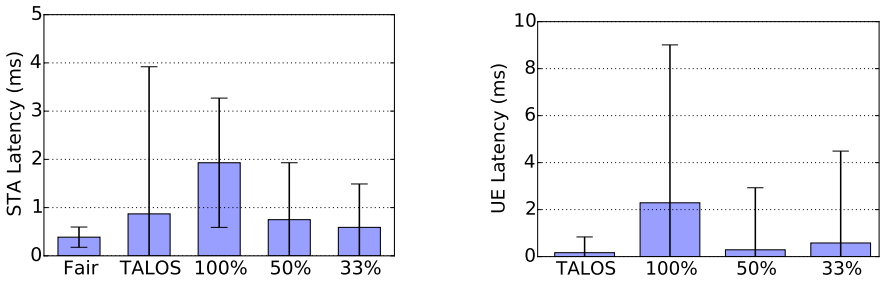
Figure 2.15: Sum throughput distribution in a larger network.

formed continously to track WiFi network dynamics, making it very challenging and inefficient. In contrast, TALOS achieves the fair WiFi throughput in all cases in a highly efficient manner.

TALOS *'s performance scales with contenders.* The fair WiFi operating throughput in TALOS does not come at the expense of LTE — in the 1-AP case, for example, the TALOS LTE throughput is within 10% of the throughput that LTE will achieve without duty cycling. Observe that with two APs, TALOS correctly matches the WiFi throughput achieved under 33% duty-cycle, while achieving 33% (from 4.5 to 6 Mbps) higher corresponding median LTE throughput.

2.6.5 TALOS Link-Layer Latency

While WiFi packet latency is measured over the entire experiment, LTE link-layer packet latency is measured only during periods when LTE is active on the unlicensed channel . This choice is justified since LTE is meant to operates its unlicensed channel



(a) WiFi (b) LTE

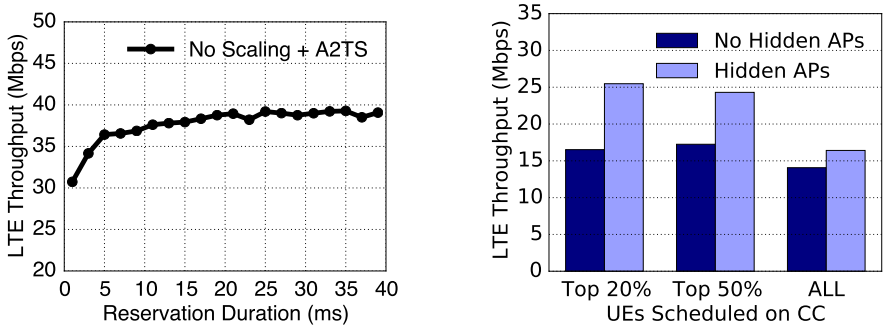
Figure 2.16: Packet latency of the WiFi and LTE networks.

together with a licensed channel. During periods when the unlicensed channel is deactivated, UE transmissions will still be scheduled on the licensed channel. Hence, we are only interested in the LTE latency during periods when the unlicensed LTE spectrum is in use.

Fig. 2.16(a) shows the mean and deviation of the downlink latency (measured from inter-packet arrivals) in WiFi networks with one WiFi link together with one other UE and an TALOS eNodeB. When WiFi exists alongside duty-cycled LTE, its latency increases steadily with an increasing duty-cycle. This is an expected result given that duty-cycled LTE does not respect the asynchronous WiFi channel access policy which will result in co-interference. Even though TALOS increases the mean latency from 0.4 ms to $0.9ms$, compared to the fair WiFi case (which has no interfering LTE nodes), it is on par with the latency achieved under the 50% duty-cycled LTE. We expect the WiFi latency to be reduced even further with LAA-LTE, due to its support for short-timescale transmissions. Fig. 2.16(b) shows that TALOS achieves up to 20× shorter latencies than simple duty-cycling. This reduction is largely due to the fact that TALOS A2TS mechanisms ensure interference-free LTE transmissions.

2.6.6 TALOS Reservation Extension

We study the performance of TALOS in LTE-to-LTE co-existence scenarios (in absence of WiFi) by augmenting our testbed evaluation with moderate-scale NS-3



(a) Reservation extension

(b) Throughput with A2TS-aware scheduling

Figure 2.17: Simulation results in larger deployments.

simulations. Our simulation contains five partially overlapping cells. Each cell has a Talos eNodeB, along with 10 UEs that are uniformly, randomly placed within 20 metres of the eNodeB. Each pair of eNodeBs are separated by at least 50 metres. The downlink transmit power of each eNodeB is fixed at 10 dBm over a 20MHz channel. Recall that the contention window size is not scaled since we do not need to maintain LTE-to-WiFi fairness here.

Fig. 2.17(a) shows the LTE throughput (averaged over all eNodeBs) over different Talos channel reservation durations. If Talos transmits LTE subframes, one-subframe at a time (1 ms), the average LTE throughput is only 30 Mbps. If the reservation duration is extended beyond $32ms$ (i.e. beyond that allowed by the 802.11 specification), the average LTE throughput approaches 40 Mbps — a 33% improvement. The longer the reservation duration, the smaller the impact of the channel access overhead, and the better the LTE resource management algorithms are able to service the downstream UEs. This also highlights the fact that longer channel durations are beneficial to maximize LTE's efficiency in unlicensed channels, when fairness to WiFi is not required, owing to the latter's absence.

2.6.7 TALOS A2TS-Aware Scheduling

We evaluate A2TS-aware scheduling using a simulation that contains one Talos eNodeB with 10 UEs in FDD downlink, along with 3 WiFi APs and a total of 12 WiFi STAs. We consider two scenarios: no hidden terminals, where all APs are located 15m from the eNodeB and thus in its carrier sensing range; and with hidden terminals, where all APs are located 50m from the eNodeB. The transmit power of all devices is set to 10dBm. Both WiFi and LTE use a 20 MHz bandwidth at a fixed modulation rate.

Fig. 2.17(b) shows the UE throughput as we vary the proportion of UEs that are scheduled on the unlicensed channel in the two cases. The reservation duration is $96ms$.

Observe that when there are no hidden terminals (i.e. WiFi devices that cannot receive the CTS-to-Self reservation), A2TS-aware scheduling has little to no impact on the throughput of LTE. This is expected as Talos can always achieve a fair share of the channel. In the face of hidden terminals, Talos achieves the best throughput when it schedules only the top 20% of the UEs (that experience the least amount of interference from hidden terminals) in the unlicensed channel. When multiple UEs are scheduled irrespective of their interference status in the unlicensed channel, there is a large drop (67%) in the overall LTE throughput. This indicates the need for better interference protection (A2TS-aware scheduling) during multi-user scheduling in unlicensed channels.

2.7 Related Work

Unlicensed LTE has received considerable attention from the industry [7, 13, 14]. Contrary to their hypothesis that WiFi is not harmed by LTE[7], we show that LTE transmission artifacts pose a serious challenge for co-existence. The impact of LTE

on WiFi in the unlicensed spectrum has been primarily studied through simulations in [40, 41, 42]. The work in [43] relies on novel DSP techniques, and hence changes to the receiver design, to decode WiFi and LTE frames from overlapping transmissions. LTE–WiFi co-existence can also utilize the subframe blanking support in LTE [44, 45]. However, reference signals are still present in blanked frames and will interfere with WiFi transmissions. A management plane that extends an SDN platform was proposed in [46] to provide coarse-grained spectrum coordination between LTE-U and WiFi. The authors of [47] designed a proportional fair allocation of spectrum between LTE-U and WiFi. However, these two schemes only maintain fairness over a longer time scale, since they are designed primarily for LTE-U. The authors of [48] took an alternative approach and allowed WiFi to take an active role in interference avoidance. Along this line, novel DSP techniques such as [49, 50, 51] enable WiFi to overcome unwanted interference in the unlicensed band. Finally, WiFi offloading [52, 53] is complimentary to our work.

2.8 Conclusion

In this chapter, we reveal several fundamental challenges still facing LTE–WiFi co-existence in unlicensed spectrum. We propose a novel *Asynchronous Access, Synchronous Transmit (A2TS)* policy for LTE base stations to address the source of this problem, namely the channel access divide between LTE and WiFi. We then design and implement Talos , a fair and efficient co-existence protocol that overcomes the challenges in realizing the benefits of A2TS in practice.

2.9 Appendix: TALOS Access Model

Talos follows the principle of random access to co-exist with WiFi links, which implies its probability of access and transmission duration will impact the through-

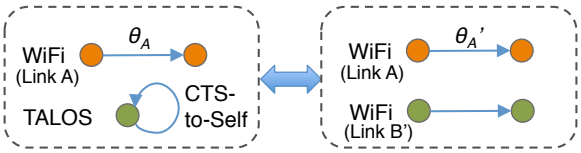


Figure 2.18: TALOS deployed next to a WiFi link and an equivalent representation with WiFi-only links. TALOS is *fair* to WiFi if throughput of WiFi is unaffected when TALOS is replaced by another WiFi link i.e., $\theta_A = \theta_A^{'}$.

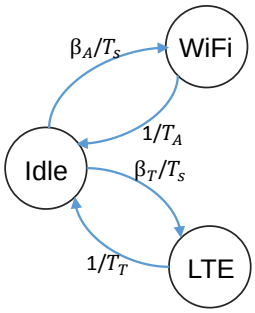


Figure 2.19: State transition diagram.

put of nearby WiFi links. To model this dependence, we consider a simple deployment where a single WiFi transmission(Link A) co-exists with TALOS , as shown in Fig. 2.18. Let β_A denote the access probability of the WiFi link, which is defined as the probability that the WiFi transmitter will transmit in a given time slot (denote slot size as T_s) in steady state. Note that β_A is determined by the WiFi contention window sizes, $W_{\min}$ and $W_{\max}$, and the collision probability [54]. We assume the WiFi packet size is exponentially distributed with mean S_A, while the channel bitrate is fixed. Hence, the packet transmission time is also exponentially distributed, which we denote by T_A. Similarly, we let β_T be the probability that TALOS will send a CTS-to-Self in a given slot, which is again determined by the contention window sizes, $LW_{\min}$ and $LW_{\max}$, and collision probability. Also, assume the channel reservation duration for LTE is exponentially distributed with mean T_T, and is independently chosen at each transmission. To keep the analysis tractable, we further assume the links don't do exponential backoff. This simplifies analysis as the access

probabilities are independent of the collision rate and are given by: $\beta_A = 2/W_{\min}$ and $\beta_T = 2/LW_{\min}$.Under these assumptions, Fig. 2.19 shows the state transition diagram of the transmissions in the medium. Using a Markovian analysis as shown in [55] we calculate the throughput, θ_A, of the WiFi link, which is given by:

$$\theta_A = \theta(\beta_A, T_A, T_T\beta_T) = \frac{S_A\beta_A}{T_s + T_A\beta_A + T_T\beta_T}$$

where $\theta(\cdot)$ is a general throughput function of a WiFi link. To show the fairness between TALOS and WiFi, we now consider an equivalent system representation where TALOS is replaced by a regular WiFi transmission, Link B', as shown in Fig. 2.18. Let $\beta_{B'}$ be its access probability, which is now determined by the regular WiFi contention window size as: $\beta_{B'} = 2/W_{\min}$, and let $T_{B'}$ denote its average transmission time. Using a similar approach as earlier, the throughput, $\theta_{A'}$, of Link A in this case becomes $\theta(\beta_A, T_A, T_{B'}\beta_{B'})$.

From our definition, TALOS is fair to WiFi if the throughput of Link A in both these cases remains unaffected, that is, $\theta_A = \theta_{A'}$. Clearly, this holds when $T_T\beta_T = T_{B'}\beta_{B'}$, which occurs if $L = \frac{T_T}{T'_B}$. Thus, by scaling the contention window size by a factor that is the ratio of the average LTE transmission duration and the average WiFi transmission duration, TALOS ensures fairness to WiFi. Though our analysis ignores the overhead from collisions and exponential backoff, we see from our evaluation and simulation results in Sec. 2.6 that even with those overheads, the fairness concept still holds.

CHAPTER III

Measurement-Based Design of Network MIMO

3.1 Introduction

A distributed multiuser MIMO (MU-MIMO) network, commonly known as *network MIMO* (netMIMO), has the potential of achieving gigabit capacity in dense wireless networks. It is targeted for enterprise networks, where generates and receives the bulk of the wireless traffic. A typical netMIMO setup consists of a central controller that does the baseband processing and a set of distributed Access Points (APs) that concurrently transmit to multiple users or stations (STAs). In contrast to traditional multi-cell wireless deployments where adding more cells increases co-channel interference, netMIMO eliminates the inter-cell interference, or converts it into useful information through tight integration of APs into a giant virtual MIMO transmitter; thus preserving capacity gains from the additional cells. In theory, the network capacity scales linearly with the number of transmitters [21]. Therefore, a netMIMO deployment with a large number of APs could easily address the bandwidth crunch of future wireless networks.

However, there are several challenges in realizing the true gains of a large-scale netMIMO. First, the distributed APs or transmitters must be accurately synchronized in time and frequency through a periodic exchange of synchronization frames. Second, the APs must be connected to the controller through a dedicated backhaul infras-

tructure that supports real-time processing. Last, the fundamental wireless capacity that is under-achieved due to the channel feedback overhead and channel estimation errors has to scale with the size of the network. Note that although reciprocity of the wireless channel precludes the need for channel feedback, the calibration requirements (as proposed in [10]) make the reciprocity assumption untenable for netMIMO deployments.

Nevertheless, recent advances have shown that high-capacity backhaul can be realized with optical fiber technology; while synchronization was achieved with a large number of APs [17]. On the other hand, STA interference that arises from the errors between estimated and observed channel reduces the achievable capacity [56].

A netMIMO derives much of the functionality from an MU-MIMO network. The APs broadcast training frames which are then used by the STAs to compute their respective Channel State Information (CSI). Each STA then transmits its CSI, either compressed or uncompressed, back to the APs during the feedback phase. As a result, the delay in acquiring the CSI feedback from STAs increases with the number of participating STAs.

In certain scenarios, the excessive feedback delay may see the channel undergo significant changes. Subsequently, when data frames are transmitted, one or more STAs may see a channel that is completely different from the channel used for computing the CSI. This phenomenon is technically known as *CSI aging*, and the delay between CSI estimation and actual transmission is called the *CSI delay*. Since netMIMO relies on zero-forcing techniques based on past CSI, improper cancellation of streams due to CSI aging leads to interference at the STAs. This unwanted interference can result in significantly lower throughput than expected from perfect cancellation. In addition, when data frames are sent for an extended period of time using the past CSI, the interference may get accentuated due to further increase in CSI delay. Therefore, both the feedback delay and the length of transmission are important factors that

determine the effect of CSI aging on netMIMO performance.

In this chapter, we propose transmission schemes for netMIMO that address the following two problems associated with CSI aging: Q1) *how to get CSI feedback without adversely affecting one or more STAs' performance?*; Q2) *how to determine the length of transmission over which the CSI remains unchanged?* Since the effects of CSI aging are strongly dependent on the topology (for e.g. mobility), we propose a measurement-based design. Our design adapts to the measurements received from the STAs, and therefore applicable to a wide range of topologies.

In particular, we make the following contributions:

- A non-heuristic, non-threshold-based training and feedback protocol, and an adaptive transmission scheme that effectively overcome the effect of CSI aging.
- A novel measurement-based approach where interference observed at the STAs is used in adapting the netMIMO protocol. We show that interference is a useful metric, and support our claims with analysis and observations.

We implement and evaluate our proposed schemes in a netMIMO testbed with real-world wireless channels. We construct a fully-synchronized netMIMO testbed on WARP Software Defined Radio [57] achieving both time and frequency synchronization. We also develop a new channel-measurement framework with custom hardware and software design [58].

The remainder of the chapter is arranged as follows: Sec. 3.2 motivates the problem of CSI aging. Sec. 3.3 specifies the model for Network MIMO. Sec. 3.4 describes our proposed transmission schemes which are followed by their implementation and evaluation in Sec. 3.5. Finally, we discuss the related work in Sec. 3.6 and conclude in Sec. 3.7.

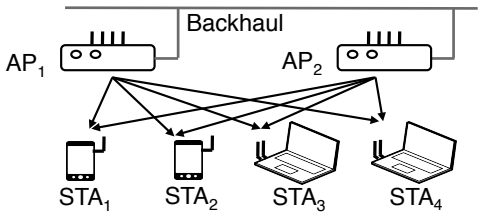


Figure 3.1: APs concurrently transmit to multiple STAs

3.2 Background and Motivation

Consider an example netMIMO deployment in Fig. 3.1, where a group of APs, all of them connected to a controller through a common backhaul link, concurrently transmit data of multiple STAs. To suppress the interference that may occur when a STA receives unwanted data of other STAs (e.g., STA1 receives signals of STAs 2–4), netMIMO uses coordinated transmission like zero-forcing and block-diagonalization [59]. This coordination is possible if the baseband transceiver processing is implemented at the controller, and the APs effectively function as remote antennas.

3.2.1 Protocol Design

In netMIMO, the downlink CSI of STAs that is required for canceling interference is obtained through a training and feedback protocol. We assume a mechanism similar to the IEEE 802.11ac MU-MIMO explicit feedback protocol [3, 60]. Training occurs via a sequence of Null Data Packet (NDP) sounding frames sent by the netMIMO APs. Each STA estimates the channel matrix by listening to the NDP frames and computes its CSI. The format of the CSI, e.g. channel matrix, beamforming matrix, precoding matrix indicator, signal-to-noise ratio, etc., is specified by the controller. The STAs encode this CSI into CSI-feedback frames and transmit them to one or more APs in a pre-determined order. Once the controller obtains the CSI, it generates the precoding weights that are applied to the antennas of the APs to send the netMIMO data packets.

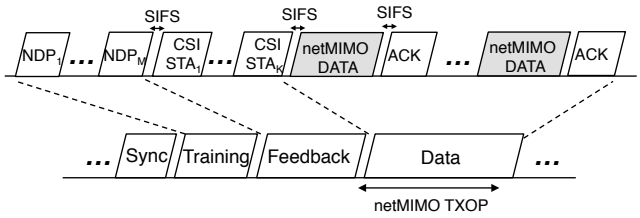


Figure 3.2: Timeline of a netMIMO transmission

CSI compression	Givens	Bits per angle	8
Feedback rate	MCS0	AP,STA Antennas	2,1
SIFS duration	16μs	Tx Power	18 dBm
NDP duration	30μs	Bandwidth	20MHz
CSI Grouping	2:1	Processing	ZF

Table 3.1: netMIMO protocol parameters

Fig. 3.2 illustrates the basic netMIMO transmission protocol. It begins when a sync frame is transmitted by the master AP to synchronize the multiple APs and to reserve the wireless medium [17], followed by a sequence of NDP frames in the training period. This is followed by sequential CSI feedback from the STAs (Poll frames for soliciting CSI feedback are not shown). The data packets of STAs are sent in the netMIMO Transmit Opportunity (TXOP) period, where the TXOP duration is determined during the channel reservation process. Once the STAs receive and decode their data frames, they send ACK/Block ACK frames to acknowledge reception.

Remark: The CSI may be encoded by the STA to reduce the amount of feedback overhead. For example, quantization [61], CSI grouping across subcarriers [3] and differential encoding [62] can be readily applied. In addition, matrix operations like Givens rotation [63] may be applied to reduce the size of channel representation.

3.2.2 Effect of CSI Aging

The wireless channel between two nodes varies with time due to relative motion between them, motion in the surroundings, random fading and other physical impairments. The rate of variation is generally captured through the notion of *coherence*

time—defined as the time period during which the wireless channel remains almost constant. For example, the coherence time is of the order of hundreds of milliseconds for stationary nodes but can be much less ($<$ 10ms) for moving nodes [64]. It therefore follows that aging of CSI is inversely related to the coherence time. To study the effect of this CSI aging, we perform some preliminary measurements of the protocol as described earlier in Section 3.2.1, on a netMIMO testbed (see Section 3.5 for details). We consider three mobility profiles for the STAs—*stationary* (Stat), *moving environment* (MovE) and *moving device* (MovD)—that capture a range of different channel characteristics. The parameters for the netMIMO transmission protocol are chosen from the 802.11ac standard [3] and are shown in Table 3.1. The details of transmit and receive processing are given in Section 3.3. Further, to isolate the effect of channel aging on the performance, we assume a noiseless channel estimation and an error-free feedback channel. This is justified by using the best feedback resolution (8 bits per angle), and the lowest modulation and coding scheme (MCS0) for feedback.

Nonlinear increase in feedback delay. Table 3.2 lists the duration of the CSI feedback frame for each STA and the total feedback delay as the number of STAs is varied. In each scenario, the number of transmit antennas is taken equal to the number of STAs. The calculation, based on the parameters in Table 3.1, is done as follows: Given a total of K transmit antennas and single receive antenna per STA, the CSI feedback is a $K \times 1$ unitary matrix (Sec. 3.3). This implies the total number of angles (both phase and rotation) after parameterization by Givens rotation is $2K-2$ (Eq. (30) in [63]). Therefore, the feedback size (bits) of each STA is given by:

$$\text{Feedback size (each)} = \frac{(2K-2) \times \#\text{BitsperAngle} \times \#\text{Subcarriers}}{\#\text{CSIgrouping}} \tag{3.1}$$

Using Eq. (3.1), the feedback duration is calculated using the MCS0 (6Mbps) feedback

	4 STAs	6 STAs	8 STAs	10 STAs
Each (ms)	0.224	0.288	0.480	0.608
Total (ms)	0.96	1.82	3.96	6.24

Table 3.2: Duration of feedback frames

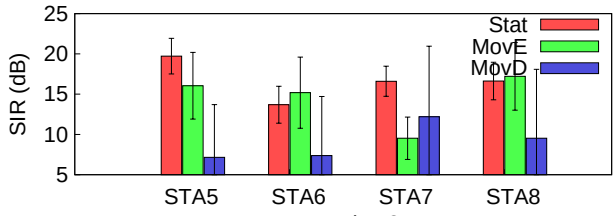
rate. This is then summed for all K STAs including the packetization and the inter-frame overheads to arrive at the total feedback duration. As one can see, the total feedback duration grows quadratically with the number of STAs. For example, the total feedback delay is 0.96ms for 4 users, but increases to 3.96ms for 8 users – a 4× increase when the number of users is doubled.

We measure the Signal-to-Interference (SIR) values for a single STA in various scenarios in a 10–STA netMIMO (Fig. 3.3(a)). The total feedback delay here is 6.24ms, and as one can see, the SIR in the MovD environment is at least 50% less than the stationary environment. Since the SIR in each of the scenarios is generally highest in case of the stationary channel, it suggests that SIR degradation (4–15dB) in non-stationary channels is due to the interference from CSI aging.

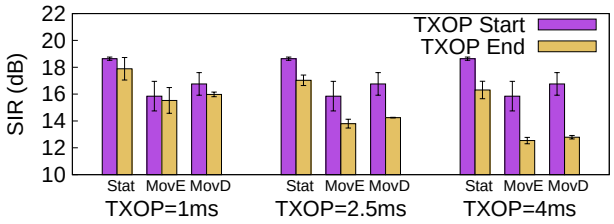
Role of TXOP duration. When the precoding weights based on past CSI are used in sending data packets for a prolonged period, it may also lead to CSI aging. This is confirmed with a netMIMO of 6 STAs as shown in Fig. 3.3(b), where there is a marked difference in the SIR values at the start of TXOP and at the end of TXOP. This is particularly significant (=6 dB) as the duration of TXOP is increased to 4ms, where CSI aging corresponds to a total 5.82ms (including the initial feedback delay of 1.82ms).

3.2.3 Measurement-Based Approach

Overcoming the effects of CSI aging requires changes to the netMIMO protocol. However, given that the indoor wireless channels are difficult to model because of their dependence on the topology and disturbances, it is difficult to quantify the effect of CSI aging. Therefore, we present a passive measurement-driven protocol



(a) Effect of feedback delay



(b) Effect of TXOP duration

Figure 3.3: Measured SIR at the STAs suggests that CSI delay during feedback and transmission leads to performance loss

that inherently accounts for the CSI delay of each STA.

Why netMIMO? Although channel variations are also applicable to a MU-MIMO network consisting of a single AP [60], we emphasize the CSI aging effect in the context of netMIMO for two reasons. First, netMIMO is designed for high-density indoor environments like conference rooms, theaters, shopping malls, etc., that inherently have mobility. Second, as compared to the single AP installation, links between STA and APs in netMIMO are more diverse, which implies lower coherence, and therefore, a greater CSI aging effect.

3.3 Network-MIMO Model

We briefly describe the netMIMO baseband signal processing at the APs and STAs, respectively. The system uses OFDM, and in the description that follows we drop the subcarrier index to indicate that it is applicable to each OFDM subcarrier. Assume there are M APs, each with N_m antennas, and $N_t = M \times N_m$ transmit antennas in total, that together serve K STAs ($K \leq N_t$). Let $\mathcal{K} = \{1, \ldots, K\}$ denote

the set of STAs, each of which is equipped with N_r receive antennas. The STA k estimates the composite $N_r \times N_t$ channel matrix, $\tilde{H}_k$, from the M NDPs in the training phase and computes the right singular matrix, V_k, of $\tilde{H}_k$. Assume that a single data stream is supported per STA and an equal amount of power is assigned to each STA. We consider the case where the CSI of STA k is the strongest right singular vector, v_k, of V_k [65]. Each STA k quantizes, compresses and encodes v_k into the CSI feedback frame, and broadcasts it to the APs. The CSI feedback frame is decoded by the APs and its contents are then passed on to the controller. After collecting CSI from each STA, the controller uses a zero-forcing (ZF) precoding to generate the precoding matrix, $W = \Gamma(\Gamma^{\dagger}\Gamma)^{-1}$, where $\Gamma = [v_1\ v_2 \dots\ v_K]$ is a matrix of singular vectors, and $(\cdot)^{\dagger}$ represents the Hermitian transpose. The precoding weights, $\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_K$, are the normalized column vectors of W used by the controller to modulate the K STA data streams. The APs then collectively transmit these modulated streams which are received and processed by each STA as a single netMIMO frame.

Let x_k denote the data of STA k, then the signal y_k received by STA k is given by

$$y_k = H_k \tilde{w}_k x_k + \overbrace{\sum_{i \neq k} H_k \tilde{w}_i x_i}^{\text{interference}} \tag{3.2}$$

where H_k is the observed channel of STA k in netMIMO data packet. In this equation, the term $R_k = \sum_{i \neq k} H_k \tilde{w}_i x_i$ is the unwanted interference from other STAs' data while the term $H_k \tilde{w}_k x_k$ is the desired signal. In ideal conditions, the interference R_k is 0 because the estimated channel $\tilde{H}_k$ is equal to the observed channel H_k, and the precoding weights $\tilde{w}_i$ satisfy $H_k \tilde{w}_i = 0$ for all $i \neq k$. However, in practice, H_k and $\tilde{H}_k$ are different due to the time-lag between them (CSI aging effect) which results in non-zero interference.

In this chapter, we assume all STAs are equipped with a single antenna, i.e., $N_r = 1$. The case with multiple antennas is left to future work. We also assume the data

symbols, $x_1, \ldots, x_K$, are zero-mean Gaussian i.i.d. with equal average power, P/K, where P is the total input power. Therefore, the interference power, I_k, observed by STA k is

$$I_k = \sum_{i \neq k} \frac{P}{K} |H_k \tilde{w}_i|^2 \tag{3.3}$$

while the SIR is given by

$$\text{SIR}_k = \frac{|H_k \tilde{w}_k|^2}{\sum_{i \neq k} |H_k \tilde{w}_i|^2} \tag{3.4}$$

Unless specified otherwise, the term *interference* refers to the interference power in Eq. (3.3). Each STA k decodes its data $\hat{x}_k$ with a Zero-Forcing (ZF) receiver given by $\hat{x}_k = (H_k \tilde{w}_k)^\dagger y_k / |H_k \tilde{w}_k|^2$. Similar to the 802.11ac MU-MIMO protocol [3], the precoded channel terms, $H_k \tilde{w}_i$, needed for ZF/MMSE estimation are known from the channel estimation symbols of the netMIMO packet. Therefore, from the perspective of a receiver, a netMIMO STA is simply an MU-MIMO STA of existing WLAN deployments.

Observe that the channel noise in the system, albeit present, has been ignored in our model. This follows from the fact that netMIMO deployments are geared towards dense deployments where interference is much stronger than noise.

The interference at a STA arises from the mismatch between the observed channel, H_k, and the precoding weights derived from the estimated channel, $\tilde{H}_k$, and is therefore a good metric to quantify channel fluctuations (Sec. 3.4.3). Furthermore, it can be approximated as: $|H_k \tilde{w}_k|^2 \text{E}[|\hat{x}_k - x_k|^2]$, which can be calculated at the receiver after demodulation. We validate this approximation in our setup by precoding with random precoding vectors at Tx antennas and measuring the average symbol error after demodulation at a receiver.

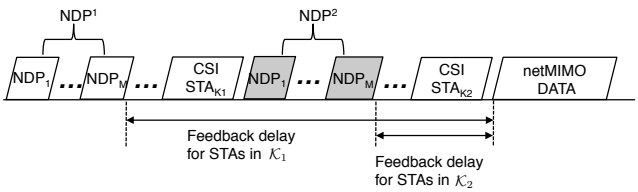


Figure 3.4: The proposed training protocol with additional NDP frames inserted in the feedback sequence

3.4 Measurement-Based Design

We present a netMIMO transmission protocol that combats CSI aging by modifying the CSI delay of STAs. We use the interference experienced by the STAs to quantify CSI aging as the metric of interest. The first part of the protocol balances the interference across STAs through two-phase training in which STAs with mobile channels are accorded a smaller CSI delay. The second part of the protocol adjusts the TXOP duration to meet the decoding threshold required throughout the TXOP duration. We begin by describing our proposed modification of the training protocol.

3.4.1 Two-Phase Training

The idea of two-phase training is to have a second NDP broadcast from APs close to the start of data transmission. STAs with high channel mobility and therefore with large channel variation relative to the first NDP use the new NDP frames to estimate their channel. The training protocol is depicted in Fig. 3.4. Here NDP^1 and NDP^2 refer to the original and the additional set of training frames, respectively. The set of STAs, denoted by $\mathcal{K}_1$, estimate the downlink channel w.r.t. NDP^1, and broadcast their CSI feedback, similar to the standard feedback procedure. The remaining set of STAs, $\mathcal{K}_2$, estimate the channel from NDP^2 and send their CSI feedback after NDP^2 is transmitted. The partitions, $\mathcal{K}_1$ and $\mathcal{K}_2$, therefore identify the two phases of training, and are specified by the controller at the start of the netMIMO transmission.

Let T_{NDP} be the collective duration of the set of NDP frames and let T_k be the fixed

feedback duration of STA k. Given a partition $\mathcal{K}_2$, the feedback delay (CSI delay) of STAs in $\mathcal{K}_1$ is given by $T^1 = \sum_{k\in\mathcal{K}} T_k + T_{\text{NDP}}$, while for STAs in $\mathcal{K}_2$, the delay is given by $T^2 = \sum_{k\in\mathcal{K}_2} T_k$. Clearly, T^1 is fixed while T^2 varies with the size of $\mathcal{K}_2$. For simplicity, the constant overhead terms like SIFS are not shown. Let $\bar{I}_k(\mathcal{K}_2)$ be the average interference of STA k where the average is taken over all OFDM symbols and subcarriers in the first data packet of the netMIMO TXOP. Its dependence on CSI delay, T^2, is indicated implicitly through the set $\mathcal{K}_2$. To keep analysis tractable, the duration of the data frames in netMIMO TXOP is fixed at 1ms. We ignore channel variations within the data frame and define the CSI delay w.r.t. the start of data frame, which is also the feedback delay of the protocol.

Initializing with a few mobile STAs in $\mathcal{K}_2$ results in a smaller feedback delay and, possibly less interference from CSI aging. But as more STAs are included in $\mathcal{K}_2$, the feedback delay, and possibly the interference, of STAs trained with NDP^2 increases. Consequently, there exists an optimal partition that balances interference across all STAs, i.e., minimizes the worst case. Taking the objective function OBJ $= \max_{k\in\mathcal{K}} \bar{I}_k(\mathcal{K}_2)$, we express this as a minimization problem:

$$\min_{\mathcal{K}_2\subseteq\mathcal{K}} \{\max_{k\in\mathcal{K}} \bar{I}_k(\mathcal{K}_2)\}. \tag{3.5}$$

Although straightforward, the problem cannot be solved because the observed channel and interference of STAs are not known *a priori* during training. Therefore, we take a passive measurement-based approach. The current location of NDP^2 is decided based on interference measurements reported by the STAs in the ACK frames from previous netMIMO transmission. We use a greedy approach: At each step the STA with the maximum reported interference is trained with NDP^2 as long as the maximum interference decreases in the next step. Algorithm 1 shows the procedure for NDP placement. It is initialized by sending only NDP^1, i.e., $\mathcal{K}_2 = \emptyset$ and measuring the

interference at STAs.

Algorithm 1 NDP placement

1: *Initialize*: $t = 0$, $\mathcal{K}_1 = \mathcal{K}$,$\mathcal{K}_2 = \emptyset$, $\bar{I}_1(\emptyset) \ldots \bar{I}_K(\emptyset)$, $\bar{I}^0_\star = \max_k \bar{I}_k(\emptyset)$
2: **for** iteration t **do**
3: Find $\bar{I}^t_\star = \max_k \bar{I}_k(\mathcal{K}_2)$, STA $k_\star = \arg\max_k \bar{I}_k(\mathcal{K}_2)$
4: **if** $\bar{I}^t_\star > \bar{I}^{t-1}_\star$ and $\mathcal{K}_2 \neq \emptyset$ **then**
5: Remove the last placed STA from $\mathcal{K}_2$ into $\mathcal{K}_1$
6: **else**
7: Place STA $k_\star$ in $\mathcal{K}_2$ if $k_\star$ was not previously moved from $\mathcal{K}_2$ into $\mathcal{K}_1$
8: **end if**
9: Train STAs in $\mathcal{K}_1$ with NDP^1 and STAs in $\mathcal{K}_2$ with NDP^2. Complete net-MIMO transmission.
10: Obtain average interference, $\bar{I}_k(\mathcal{K}_2)$, of each STA k
11: **end for**

At every iteration of the NDP placement algorithm, the maximum observed interference among STAs is compared with previous maximum interference (line 4). If there is a decrease, the algorithm tries to further decrease OBJ by training the STA with maximum interference with NDP^2 (line 7). If there is an increase in OBJ, the last placed STA in $\mathcal{K}_2$ is shifted to $\mathcal{K}_1$. Intuitively, the NDP placement algorithm addresses CSI aging by reducing CSI delay of mobile STAs.

Overhead. The overhead of additional NDP frames is negligible compared to the feedback duration. For instance, in a 4-AP netMIMO setup with 8 STAs, T_{NDP} is 0.18ms, which is around 4.6% of the 3.96ms total feedback duration.

3.4.2 Adaptive TXOP sizing

The netMIMO transmit opportunity (TXOP) is the common transmission period in which a single or a burst of netMIMO data packets of STAs are transmitted. Since the transmission is based on precoding from past CSI, a limit on the duration of TXOP is required to avoid the interference that arises from excessive CSI delay. A similar TXOP limit has been defined in 802.11 EDCA mechanism, albeit to improve MAC-level efficiency. Also, since different STAs have different channel characteristics,

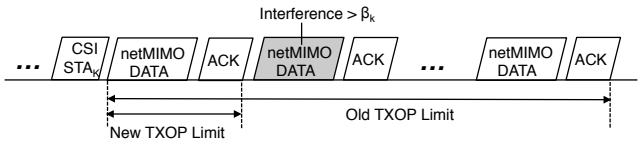


Figure 3.5: Adjusting TXOP based on the interference

the interference in the TXOP can be different for each STA. This suggests that each STA needs to have its own TXOP limit within the netMIMO TXOP according to its tolerance towards interference.

We propose a netMIMO protocol that adapts the TXOP limit of each STA while satisfying its decoding requirements. The basic idea is to observe the interference in the TXOP of a previous netMIMO transmission to set the limit for the next, as shown in Fig. 3.5. Since a TXOP is made up of multiple data packets, which are also utilized in sending the interference reports, the TXOP limit is specified in the integral number of constant sized 1ms data packets. Algorithm 2 shows the procedure for the proposed netMIMO TXOP Adjustment that is run at every netMIMO transmission.

Algorithm 2 TXOP Adjustment

1: *Initialize*: $t = 0$, L_k = TXOP Limit of STA, β_k = interference threshold of STA k
2: **for** iteration t **do**
3: Obtain average interference, $\bar{I}_k^e$, from the end of TXOP of STA k
4: **for** each k with $\bar{I}_k^e \geq \beta_k$ **do**
5: Set L_k to the first time instant ($>$ minimum TXOP limit) in TXOP where interference of STA k exceeds β_k
6: **end for**
7: **for** each k with $\bar{I}_k^e < \beta_k$ **do**
8: $L_k \leftarrow \max(L_1, \ldots, L_K)$
9: **end for**
10: Set netMIMO TXOP Limit, $L = \max(L_1, \ldots, L_K)$
11: **end for**

The algorithm initializes with the default TXOP duration L_k and interference threshold β_k of each STA k. This threshold depends on the fixed modulation and

coding scheme (MCS) used by the STA in the TXOP. Its calculation is given in Section 3.4.3. In lines 4–5, the algorithm decreases the TXOP Limit of STA k when it exceeds its interference threshold. On the other hand, the TXOP limit of STA whose interference is below the threshold is set to the maximum value (line 8) of L which is also the netMIMO TXOP limit. Similar to MU-MIMO transmission in 802.11ac [3], zero-padding is done for STAs whose TXOP limit is less than the transmission duration.

In summary, the two-phase training mitigates the interference at the start of TXOP while the TXOP Adjustment algorithm tries to reduce the interference that shows up towards the end of transmission. Both of these approaches therefore complement each other.

3.4.3 Theoretical Analysis

We now present some key insights into the underlying variables of CSI delay and interference. We back these insights with measurements obtained from our testbed. We then show that the proposed protocol follows directly from these observations.

Insight 1(Independence): Interference at a STA is independent of CSI delay of other STAs.

Explanation: This follows from the fact that given the precoding weights $\tilde{w}_1, \ldots, \tilde{w}_K$ and the estimated channel $\tilde{H}_k$ of STA k, the interference I_k is only dependent on its observed channel H_k, and is therefore independent of CSI delays of other STAs.

Insight 2(Monotonicity): The average interference experienced by a STA is directly related to its channel variations. Moreover, it increases *roughly* monotonically with the CSI delay of the STA.

Explanation: We can express the observed channel of STA k in terms of the estimated channel and the error e_k as:

$$H_k = \tilde{H}_k + e_k \tag{3.6}$$

where e_k is a zero-mean spatially white gaussian error term. The zero-mean behavior was indeed verified in our testbed measurements. Next, consider a single data frame which is made up of a sequence of OFDM symbols. For a single subcarrier, we can express the expected value of interference (using Eq. (3.3)) as follows:

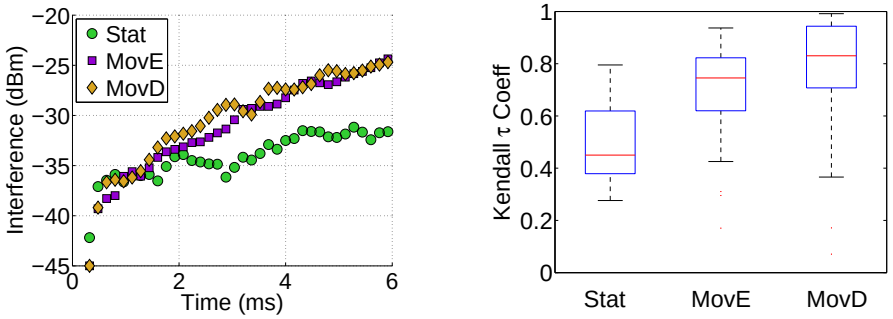
$$\begin{aligned}
&\mathrm{E}[\sum_{i\neq k}\frac{P}{K}|(\tilde{H}_k+e_k)\tilde{w}_i|^2] \overset{(a)}{=} \mathrm{E}[\sum_{i\neq k}\frac{P}{K}|e_k\tilde{w}_i|^2] \\
&=\sum_{i\neq k}\frac{P}{K}\mathrm{E}[(e_k\tilde{w}_i)^\dagger(e_k\tilde{w}_i)] = \sum_{i\neq k}\frac{P}{K}\tilde{w}_i^\dagger\mathrm{E}[e_k^\dagger e_k]\tilde{w}_i \\
&\overset{(b)}{=}\sum_{i\neq k}\frac{P}{K}\tilde{w}_i^\dagger\sigma_k^2 I\tilde{w}_i = \sum_{i\neq k}\frac{P}{K}\tilde{w}_i^\dagger\tilde{w}_i\sigma_k^2 \overset{(c)}{=} C\sigma_k^2
\end{aligned}$$

where $C=(K-1)P/K$ is a constant; (a) follows by using $\tilde{H}_k\tilde{w}_i=0$ for all $i\neq k$, (b) is result of the white gaussian error term e_k, and (c) uses $\tilde{w}_i^\dagger\tilde{w}_i=1$ for all i. This implies the observed average interference[1] that is obtained from multiple OFDM symbols of STA k can be approximated as

$$\bar{I}_k \approx C\sigma_k^2 \tag{3.7}$$

Therefore, the average interference at a STA is directly proportional to the power of channel variations, σ_k^2. This suggests that as channel variations grow larger with the CSI delay, the interference should also increase. To investigate if this indeed holds, we experimentally measure the interference as a function of STA's CSI delay under different scenarios. Fig. 3.6(a) gives an example of the observed STA interference in a 10-STA netMIMO setup. As expected, the interference in general increases with CSI delay, and more so in case of non-stationary environments. We run through these observations with the Kendall τ rank correlation test [66] which is non-parametric test to measure monotonicity between two variables. A correlation coefficient close to $+1$ indicates a monotonically increasing relationship while a value close to -1

[1]Average from a single sample path is used as approximation of mean across all paths



(a) Example of STA interference (averaged over 10MHz) vs. CSI delay

(b) Kendall τ coefficient for interference vs. CSI delay with 0.01 significance level

Figure 3.6: Examples and results of statistical test show that interference increases with CSI delay

suggests a dependency that is monotonically decreasing. Fig. 3.6(b) plots the range of Kendall τ rank correlation coefficient values calculated for the interference with respect to the CSI delay observed over 10 runs. The significance level was set to 0.01. In the majority of runs and under different channel conditions, we observe a rank correlation coefficient, $\tau > 0.7$, which confirms our insight on the monotonic nature of interference as a function of CSI delay.

In what follows, we make use of the assumptions on independence and *strict* monotonicity, and make another simplifying assumption that channel behavior remains constant across multiple iterations.

Proposition 1. The NDP placement algorithm finds the optimal solution to Eq. 3.5 within K iterations.

Proof. We first show that NDP placement does converge. By construction, at every step, a STA from $\mathcal{K}_1$ is moved to $\mathcal{K}_2$, unless the maximum interference among STAs increases. The CSI delay of STAs in $\mathcal{K}_1$ is given by $T^1 = \sum_{k\in\mathcal{K}} T_k + T_{\text{NDP}}$ which remains constant, so interference of STAs in $\mathcal{K}_1$ also remains constant by assumption. On the other hand, from the monotonicity assumption, the interference of STAs in $\mathcal{K}_2$ increases as the CSI delay $T^2 = \sum_{k\in\mathcal{K}_2} T_k$ increases which happens when a STA is moved to $\mathcal{K}_2$. Clearly, if at any point $k_\star \in \mathcal{K}_2$, then partitions don't change further.

Also, if at any iteration t, $\bar{I}_\star^t > \bar{I}_\star^{t-1}$, STA $k_\star$ form iteration $t-1$ will be moved back to $\mathcal{K}_1$. Subsequently, in iteration $t+1$ STA $k_\star$ will continue to remain in $\mathcal{K}_1$. Therefore, the partitions converge in both cases.

To show that the converged partition, $\mathcal{K}_1, \mathcal{K}_2$, is optimal, assume there exists another partition $\mathcal{G}_1, \mathcal{G}_2$ that does strictly better. Let STA $k_\star$ and STA $r_\star$ have the maximum interference in the two scenarios, respectively. It follows that $I_{r_\star}(\mathcal{G}_2) < I_{k_\star}(\mathcal{K}_2)$. For any $k \in \mathcal{K}_2$, it is true that $k \in \mathcal{G}_2$ because otherwise if $k \in \mathcal{G}_1$, then $I_k(\mathcal{G}_2) \leq I_{r_\star}(\mathcal{G}_2) < I_{k_\star}(\mathcal{K}_2)$ from assumption. Further, $I_{k_\star}(\mathcal{K}_2) < I_k(\mathcal{G}_2)$, because at some point, T', in NDP placement STA k would be moved from the first partition and the maximum interference strictly decreases at each step, and also because the CSI delay of STA k before T' is same as with the partitions $\mathcal{G}_1, \mathcal{G}_2$. As a result, $I_k(\mathcal{G}_2) < I_k(\mathcal{G}_2)$ which is a contradiction. Therefore, we have $\mathcal{K}_2 \subset \mathcal{G}_2$ since the two partitions cannot be the same. Now, consider two cases when $k_\star \in \mathcal{K}_1$ and $k_\star \in \mathcal{K}_2$. In the first, it follows that $k_\star \in \mathcal{G}_2$ because otherwise if $k_\star \in \mathcal{G}_1$ implies $I_{r_\star}(\mathcal{G}_2) \geq I_{k_\star}(\mathcal{G}_2) = I_{k_\star}(\mathcal{K}_2)$, which is not true. Now, consider, $I_{k_\star}(\mathcal{K}_2 \cup k_\star) > I_{k_\star}(\mathcal{K}_2) > I_{r_\star}(\mathcal{G}_2) > I_{k_\star}(\mathcal{G}_2)$, from the stopping condition of the algorithm. This is a contradiction since $\mathcal{K}_2 \cup k_\star \subseteq \mathcal{G}_2$. For the second case, since $k_\star \in \mathcal{K}_2 \subset \mathcal{G}_2$, it follows $I_{r_\star}(\mathcal{G}_2) \geq I_{k_\star}(\mathcal{G}_2) > I_{k_\star}(\mathcal{K}_2)$ which is also contradiction to our assumption. Consequently, there cannot exist another partition with does strictly better than the NDP placement algorithm.

Proposition 2. If the interference threshold β_k of STA k satisfies $\beta_k = \frac{P}{K}(\tilde{\Lambda}_k^{\min})^2/\alpha_k$, where α_k is the SIR decoding threshold, and $\tilde{\Lambda}_k^{\min}$ is the minimum largest singular value of $\tilde{H}_k$ across subcarriers, then the average SIR after TXOP Adjustment satisfies $\overline{SIR}_k \gtrapprox \alpha_k$ for each data frame.

Proof. The TXOP adjustment proceeds by doing a series of adjustments to the TXOP limit of each STA, and sets it either to the minimum TXOP limit or increases to the maximum of the TXOP limits of all STAs. Therefore, after a finite number of steps the TXOP limit of each STA k converges to a constant value such that the average

interference, $\bar{I}_k$, throughout the TXOP is less than β_k (if not the case then it is set to the minimum possible TXOP limit and there are no performance guarantees). Consider a subcarrier of a data frame within the TXOP, we can express:

$$\begin{aligned}
\mathrm{E}[SIR_k] &= \mathrm{E}[\frac{P}{K}\frac{|(\tilde{H}_k + e_k)\tilde{w}_k|^2}{I_k}] \overset{(d)}{\approx} \mathrm{E}[\frac{P}{K}\frac{|\tilde{H}_k\tilde{w}_k|^2}{I_k}] \\
&\overset{(e)}{=} \mathrm{E}[\frac{P}{K}\frac{\Lambda_k^2}{I_k}] \overset{(f)}{\geq} \mathrm{E}[\frac{P}{K}\frac{(\Lambda_k^{min})^2}{I_k}] \overset{(g)}{\geq} \frac{P}{K}\frac{(\Lambda_k^{min})^2}{\mathrm{E}[I_k]} \\
&\overset{(h)}{\approx} \frac{P}{K}\frac{(\Lambda_k^{min})^2}{\bar{I}_k} \geq \frac{(\Lambda_k^{min})^2}{\beta_k} = \alpha_k
\end{aligned}$$

where (d) is an empirical approximation when $|H_k w_k| >> |e_k w_k|$, (e) uses the fact that $\tilde{w}_k$ is normalized vector of $W = (\Gamma^{\dagger})^{-1}$ and H_k has a single non-zero singular value, (f) follows by using the minimum singular value across subcarriers, (g) follows from Jensen's inequality and (h) is obtained by replacing expectation with the average sum. Therefore, for any data frame in a TXOP, we may approximate the average measured SIR measured across the frame's OFDM symbols and subcarriers as $\overline{SIR}_k \gtrapprox \alpha_k$.

3.5 Evaluation

In this section, we provide the details of our netMIMO testbed, the evaluation methodology, the implementation of the proposed algorithms and their performance evaluation.

3.5.1 netMIMO Testbed

We implement a prototype of netMIMO using the WARP SDR platform [57]. The setup consists of a maximum of 5 APs (with 2 Tx antennas each), and 10 STAs, each with 1 Rx antenna. The controller function is implemented on a back-end server connected to the WARP boards via Ethernet with a Gigabit switch. We make use of the WARPLabv7.3 drivers with their efficient WARP-PC interface to interface with

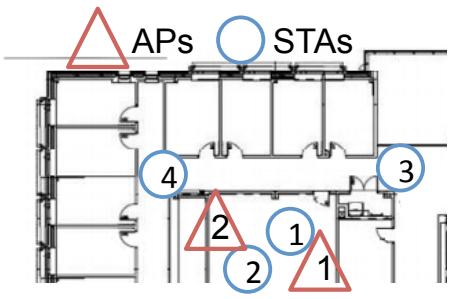


Figure 3.7: Deployment of net-MIMO testbed

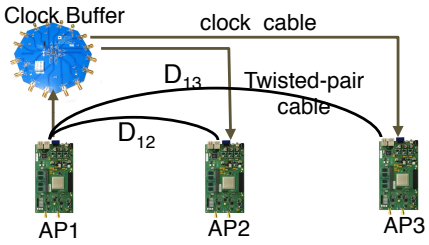


Figure 3.8: Time and frequency synchronization

the WARP boards.

Fig. 3.7 shows the layout of the netMIMO testbed deployed in our Department building. To manage the large number of WARP boards we grouped them into separate AP and STA clusters. We have 2 AP clusters, and 3 STA clusters, with a combined total of 10 Tx and Rx antennas. The STA WARP boards are placed on a movable cart while carrying out the experiments.

3.5.1.1 Signal Processing

Following the 802.11n/ac standards, we have implemented a full-fledged netMIMO OFDM system. We use the basic 20MHz 64-fft OFDM (48 data subcarriers) design and upsample it by a factor of 2 to get an effective bandwidth of 10MHz. We use the channel 14 in the 2.4GHz ISM band that is unused by nearby devices.

At the start of every netMIMO transmission, a *sync* signal is sent by the controller to trigger the APs and the STAs to start their transmit and receive chain, respectively. Each AP broadcasts an 802.11 NDP packet composed of a short training field (STF) and a long training field (LTF). Each STA runs an auto-correlation algorithm to detect the STF and extracts the start time of the received packet. It also uses the STF to estimate the frequency-offset and corrects it accordingly. The LTF of an AP consists of OFDM training symbols on each Tx antenna of the AP, and is used

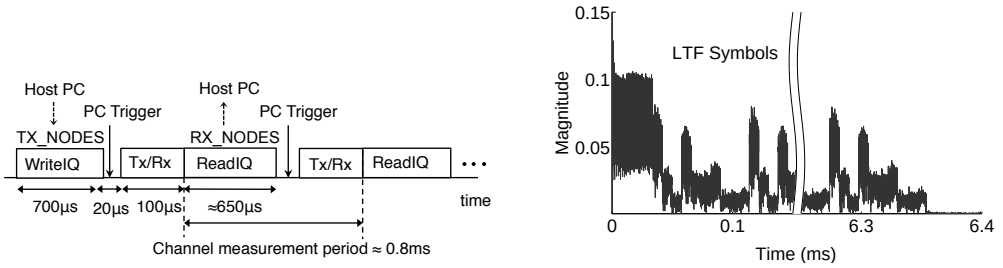
by the STA to estimate channel w.r.t. each Tx antenna. The CSI feedback frames are not sent over-the-air and made implicitly available to the controller. Finally, a netMIMO data frame consisting of cyclic-shifted STFs, and precoded training and data is transmitted.

3.5.1.2 Synchronization

For the netMIMO signal processing to be valid, we require that all AP nodes in netMIMO are completely synchronized. By default, we take AP_1 as the master AP and synchronize the rest of the APs with AP_1.

Time Synchronization: At a high level, time synchronization means that all APs transmit simultaneously. However, in the context of netMIMO which uses OFDM, it requires that for each STA, the precoded OFDM frames transmitted by the APs as part of a single netMIMO packet must arrive within a cyclic prefix (CP) window. By ignoring the small differences in the propagation delay between APs and STAs, this implies that time synchronization in netMIMO is achieved when all APs begin transmission within a CP period. We ensure this in two steps.

First, we use a twisted-pair cable assembly and connect the debug headers on the WARP board of AP_1 with that of other APs. This ensures that the rest of the APs can be triggered by AP_1 while AP_1's input trigger is set to a PC trigger that is sent over Ethernet. The twisted-pair cable which extends up to 30m in our setup has a propagation delay of the order of 100ns (typical value of CP is 400 or 800ns). As a result, we observe a significant time-lag in triggering from AP_1. In our second step, we address this lag by introducing transmission offsets among the APs, i.e., we delay the start of packet transmission at an AP by a fixed duration. Let D_j denote the delay of the twisted-pair cable between AP_1 and AP j. Note that $D_1 = 0$. Also,let O_j be the transmission offset of AP j. We find offsets O_1 to O_M that satisfy the



(a) Timeline of function calls in CWARP

(b) LTF symbol stream for channel estimation

Figure 3.9: Details of the channel measurement framework that allows real-time evaluation of netMIMO

following constraint.

$$\max_{1\leq j\leq M}\{O_j + D_j\} - \min_{1\leq j\leq M}\{O_j + D_j\} \leq \text{CP}. \tag{3.8}$$

As the above constraint is under-specified, the offsets are found easily through a simple search. To allow for tolerance, we use CP/2 as the window size in our setup.

Frequency Synchronization: As illustrated in Fig. 3.8, we frequency-synchronize all netMIMO APs by sourcing their RF and sampling clock from the 40MHz reference clock of AP_1. This is implemented with the CM-MMCX module of WARPv3 board that provides external interface to the reference clocks. We use the RG-174 coax cables that have an attenuation rating of 6.6dB/100ft at 50MHz for connecting the clock sources. Further, we use ADCLK954 boards as external clock buffers to prevent clock drift that may arise from the attenuation.

Phase Synchronization: Two WARP boards that are frequency synchronized may still differ in phase due to a random phase introduced by the Amplifier (LNA) circuit. Fortunately, this random phase remains constant once the circuit is powered on. Thus, a constant phase difference between two APs is easily corrected.

3.5.2 Evaluation Methodology

3.5.2.1 Trace-based evaluation

Our netMIMO setup uses the WARPLAB 7.3 library written in MATLAB to do signal processing. However, the execution time of MATLAB for netMIMO processing and WARP read/write exceeds 100ms and thus renders it useless for real-time evaluation. Therefore, we resort to a trace-based evaluation of netMIMO. The idea is to continuously collect the channel measurements between every pair of antennas in real time. This is achieved by repeatedly sending LTF symbols from the APs and processing the received symbols at the STAs to estimate the channel. We then play back the channel traces with the netMIMO transmission protocols through virtual timers which allow us to do an offline yet realistic evaluation.

3.5.2.2 Channel-Measurement Framework

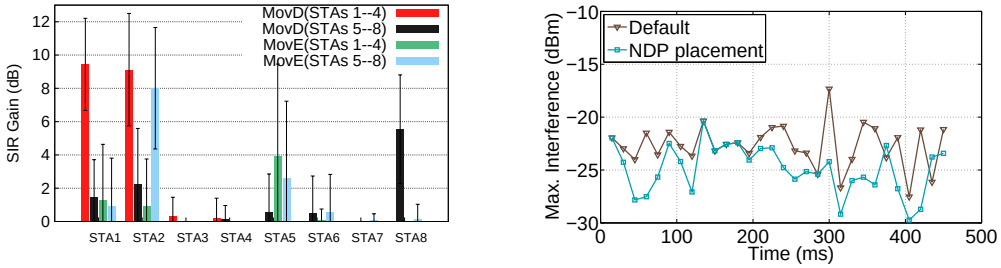
To do an accurate trace-based evaluation of netMIMO, we require the $N_r \times N_t$ channel measurements to be done as frequently as possible. This implies that we need to either send/receive a single long packet (>10ms) of LTF symbols or send/receive the LTF symbols at a very fast rate (every 1ms or so). The WARP design, however, has a maximum buffer size of only 2^{15} samples (airtime of 0.8ms at 40MHz). Moreover, reading just a single buffer into WARPLab incurs an average delay of around 3ms. Therefore, it is difficult to obtain accurate channel traces using the current WARP framework.

We overcome these limitations by designing our own hardware and software solutions. We modify the WARP FPGA design by halving the sampling clock and increasing the buffer size to 2^{16} samples (beyond 2^{16} is not possible due to FPGA Bus width limitations). This allows us to send/receive signals up to 6.4ms long—an 8× increase in the transmission airtime. Consequently, we are able to send/receive

LTF symbols to accurately track the channel as shown in Fig. 3.9(b). We have also written a multi-threaded C driver that directly interfaces with the WARP hardware [58]. This lightweight driver completely replaces the much slower WARPLAB driver that is written in MATLAB. On comparing the performance of the basic readIQ routine which reads the IQ baseband buffers into the PC, we witness an almost 3× to 10× reduction in read delay as the number of buffers is increased from 4 to 10. In particular, the average readIQ delay for reading 2^{11} samples in case of 6 STAs is only 0.8ms, i.e., smaller packets can be sent and received within 1ms (Fig. 3.9(a)), which allows us to do channel measurements with high accuracy. The long transmission design is used for validating the STA interference behavior, e.g., Section 3.4.3, while the WARP driver is used in the evaluation of our algorithms. We would like to point out that measurements with WARP boards are highly sensitive to noise, external interference and other physical impairments. Therefore, we apply a locally-weighted smoothing on the time variations of channel magnitude and phase, similar to the channel tracking procedure of OFDM receivers.

3.5.2.3 Mobility Experiments

In order to account for the channel conditions in real netMIMO deployments, we consider three mobility scenarios in our evaluation: (1) *Stationary*(Stat)—This corresponds to the scenario when the location of STAs is fixed, and there are no disturbances from the motion of people or nearby objects. (2) *Moving environment*(MovE)—Environmental mobility is simulated through human actions such as arm movements and walking while the nodes remain fixed. (3) *Moving device*(MovD)—Device mobility is simulated by moving the cart on which STA nodes are placed while taking measurements. This motion closely resembles the pedestrian mobility of smartphones and other mobile devices.



(a) Observed gains in four experiments

(b) Maximum interference among STAs

Figure 3.10: Evaluation of NDP placement in 8–STA netMIMO. Results show significant gains for mobile STAs.

3.5.3 NDP Placement

We implement and run the NDP placement given in Algorithm 1 over a period of 450ms where a netMIMO transmission occurs every 15ms. A number of scenarios for STAs' mobility are played out while the channel measurements are taken. Fig. 3.10(a) shows the SIR gains of STAs with the NDP placement compared to the default training protocol in four different scenarios: MovD where one among the group of STAs 1–4 and STAs 5–8 are placed and moved on a cart, and repeated again in MovE where instead external disturbances are present around one of the group of STAs. The gain is calculated by measuring the SIR of a single netMIMO packet that is transmitted following the feedback process. As one can see, the gains for some STAs can be significant (5–10dB) and is due to the smaller CSI delay resulting in reduced interference. Most importantly, the gains are strongly correlated to the channel mobility of the STAs. For example, when STAs 1–4 are moved, all of STAs 1–4 see gains while STAs 5–8 see almost none. When STAs 5–8 are moved, STAs 6 and 8 have noticeable gains while STA3's and STA4's performance remains unchanged. This can be explained from the two different CSI delays—a few STAs' delay is reduced (from NDP^2) while for others it remains the same (NDP^1). Note that due to the coarseness of the measurement readings, the actual gains may deviate slightly than what has been reported.

Fig. 3.10(b) shows the results for the MovD (STAs 1–4) scenario of the 8–STA netMIMO system which according to Table 3.2 incurs a feedback delay of 3.96ms with single NDP training. The average readIQ delay of reading each WARP buffer was observed to be 1.07ms. Therefore, channel readings are available at 4 time points within the feedback period. In Fig. 3.10(b), we plot the maximum average interference of STAs ($\bar{I}_{\star}^{t}$ in Algorithm 1) as a function of netMIMO iteration. As expected, the NDP placement strives to lower the maximum interference compared to the default scheme.

3.5.4 TXOP Adjustment

Similar to the NDP placement, the TXOP Adjustment (TxAj) procedure in Algorithm 2 uses the past interference to adapt every netMIMO transmission. The CSI delay of STAs is fixed while the netMIMO TXOP duration is varied. Since the objective here is to meet the SIR threshold (β_k) of a STA according to its MCS, we evaluate the performance of our proposed approach with all possible MCS values. We use the widely adopted MCS-threshold lookup table for 802.11 wireless environments[67]. Fig. 3.11 shows the evaluation results of a single STA in an 8-STA netMIMO setup with TXOP limit of 4ms. The minimum TXOP limit of each STA is set to 1ms. The TXOP limit value is the initial TXOP duration of all netMIMO STAs which remains constant in the default scheme but is adjusted for each STA in our proposed scheme. Note that the CSI delay here consists of the TXOP duration in addition to the feedback delay of 3.96ms. For each MCS, we calculate the percentage of STA's total transmissions that do not meet the corresponding average SIR threshold and hence would result in packet errors. Since the accuracy of measurement is limited to 1ms, the SIR is measured only for a few packets in the TXOP. A higher MCS means requires larger SIR (e.g., MCS7 requires SIR $>$ 23dB); the plots in Fig. 3.11 therefore exhibit an increasing trend. With our TXOP Adjustment which changes

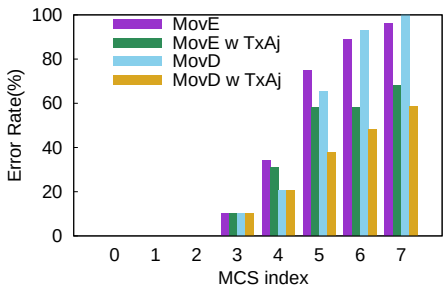


Figure 3.11: Error rate measured for each MCS before and after TXOP Adjustment

# STAs	$K = 4$		$K = 6$		$K = 8$	
Location	(1,2)	(3,4)	(1,2)	(3,4)	(1,2)	(3,4)
Default (Sec. 3.2.1)	134	113	199	108	279	150
Proposed (Sec. 3.4)	142	119	220	123	292	174

Table 3.3: Average total netMIMO-throughput (Mbps)

the transmission length according to the interference, we see a noticeable reduction in error rate – up to and 42%.

3.5.5 Field Test

We conduct a field test of the netMIMO setup by running both NDP placement and TXOP Adjustment simultaneously for a period of 450ms. We consider the ambient wireless environment of our testbed and further orchestrate all STAs in motion together with the motion of people/objects around. We consider two different locations of the STA clusters, with Location (3,4) being in NLOS environment w.r.t. the APs, and placed farther than Location (1,2). The initial TXOP limit was set to 4ms. The average throughput of each STA is calculated by averaging over all possible MCS data rates, which is then added for all STAs to arrive at the average netMIMO throughput, as shown in Table 3.3. It can be concluded that the throughput gains are modest in the case of 4–STA netMIMO, but become more prominent as the system is scaled—more than 15% in case of 8 STAs. Note that the throughput is per unit time within a netMIMO TXOP.

3.5.6 Discussion

Theoretical basis: While the monotonic behavior of interference is loosely observed in our measurements, the channel state across netMIMO iterations can be quite different. Consequently, the theoretical analysis in Sec. 3.4.3 does not necessarily apply, but rather provides a guiding principle.

Rate adaptation and Decoding: We have used fixed MCS rates for each STA for determining the transmission duration. In addition, decoding is assumed to fail completely if the decoding threshold is not met. However, our TXOP adjustment can be further improved if rate adaptation is applied within and across TXOPs, whereas soft decoding at STA will reduce the TXOP error rate.

3.6 Related Work

Distributed MU-MIMO systems have been implemented through centralized [17, 16] and decentralized WLAN architectures[68]. Furthermore, a downlink MU-MIMO base station with 64 antennas was implemented in [10]. Each of them assumed perfect CSI, or rather stationary wireless channel, which is justified given that their experiments were carried out in a controlled environment. However, we find that user mobility has a significant effect on the performance of such systems. While the other alternative of implicit CSI feedback, as employed in [10], effectively mitigates the CSI aging effects, its calibration requirements make it less practical for WLAN devices, and as such has not been adopted in the 802.11ac standard.

In theory, the results on MU-MIMO capacity with the overhead of training and feedback, imperfect channel estimation, and channel aging are well known [21, 69, 61]. However, few systems have validated these results in practice. The user mobility profile was considered in [70] to select a transmission strategy. The authors claimed that netMIMO architecture was only suitable for stationary clients. In [64] and [60],

experiments were performed to show the relationship between feedback delay and capacity of 802.11n SU-MIMO and MU-MIMO users, respectively. The authors in [60] observed that the performance of MU-MIMO is highly sensitive to CSI delay and aging, and the sensitivity is dependent on the number of Tx/Rx antennas and the interference cancellation ability of the STAs. It is stated that without interference cancellation, capacity decreases considerably within 5ms, which is also consistent with our findings. However, their results are limited to just 4 STAs, assumes multiple Rx antennas, and employs fixed training interval for each STA irrespective of its mobility state. The recent work in [71] on MU-MIMO feedback is closely related to ours. It proposes an adaptive feedback compression scheme based on frequency and time-domain compression techniques, and evaluates them in realistic indoor channels. The setup, however, considers single AP and 2 STAs, and does not take into account the scaling of feedback overhead or the diversity from having multiple APs.

3.7 Conlusion

Network MIMO has the potential to meet the capacity requirements of future wireless networks. In this chapter, we consider how CSI aging affects the performance of a large-scale netMIMO in real and accurately measured indoor wireless channels. We propose a two-phase feedback protocol and an adaptive transmission scheme, which adapt from the interference measured by the STAs. Our approach exploits the underlying relationship between interference and CSI delay, which is validated in our experiments. Our evaluations demonstrate significant gains in the case of mobile wireless channels.

CHAPTER IV

Scalable Real-Time Transport of Baseband Traffic

4.1 Introduction

In upcoming wireless architectures such as C-RAN [5, 12], and Massive-MIMO [9, 10], a baseband transport network connects the antenna/radio deployments to the backend processing infrastructure. For instance, in a C-RAN deployment, a fronthaul network carries the baseband samples between the remote radio heads and the baseband processors located in a datacenter. Such architectures provide numerous cost advantages: lesser power consumption from resource pooling, quicker upgrade and replacement cycles, support for advanced signal processing, and flexibility in the management of radio infrastructure.

Since the baseband transport network is the key enabler of such architectures, its design must meet the following requirements to maximize the cost benefits.

Guarantees. The real-time nature of the wireless processing imposes stringent constraints on the transport network; the network must provide end-to-end (e2e) guarantees for delay and jitter. For instance, the transport delay bound can be as little as few microseconds for WiFi samples [72], to hundreds of microseconds for LTE samples [73]. In addition, the transport behavior of the radio samples must be predictable, that is, given a network topology and the traffic sources, one must be able to model the delivery of the baseband traffic. This model is necessary for an

e2e schedulability analysis — determining whether the given network can meet the requested delay bounds. In the real-time systems literature, however, it is known that modeling packet traffic in the network core is intractable due to the non-periodic nature of arrivals [74, 11]. Therefore, in a general baseband transport network consisting of multiple switches, it is not entirely clear what packet scheduling policies (implemented by switches) will achieve e2e schedulability.

Aggregation. The baseband transport design should also support traffic aggregation from the radios, for scenarios such as MIMO processing, or, for resource pooling in C-RAN, where a common compute platform decodes multiple base-stations. Inherently, this can be achieved from a tree-based design using aggregation switches. However, traffic aggregation without proper scheduling introduces variable queuing delays, and moreover, cannot differentiate between traffic flows based on their delay bounds.

Scalable. Considering the size of future radio deployments, the baseband transport design should be scalable. It must be extensible to any number of radios while preserving its predictable behavior. Also, it must require only few additional resources (e.g., cables, switches) to add a large number of radios to the network. The scalability of baseband transport is desirable, if not necessary, for massive MIMO systems which are equipped with tens or hundreds of antennas/radios. Existing datacenter designs [75] that are optimized for scalability, are likely candidates for baseband transport. However, they are primarily designed to handle bisection traffic between the compute clusters whereas traffic flows in a baseband network are exclusively in the north–south direction.

Optimal. Whether the baseband traffic is schedulable or not depends on the transport rate of the radios, which in turn depends on the sample quantization widths used at the radios. As the selected quantization widths affect the wireless capacity through quantization noise, there is an indirect dependence between schedulability

of the traffic and the wireless capacity of the network. Ideally, the network should operate at a point that ensures schedulability but also maximizes wireless capacity.

Based on the above requirements, we introduce DISTRO, a design for real-time baseband transport networks (such as fronthaul networks) that can potentially scale to a large number of radios. DISTRO utilizes a logical tree structure of radio front-ends and network switches; the radios represent the leaves of the tree, the network switches represent the intermediate nodes, and the root of the tree is the common aggregation point that connects to a pool of baseband processors. DISTRO's design supports real-time transport as it allows us to bound the maximum transport delay of each baseband packet. Specifically, using a constrained tree design, the upper bound on the waiting time at any switch can be obtained irrespective of the input arrival sequence, using which, one can get the maximum total delay of a baseband flow. As a result, the network switches can utilize schedulability results from the real-time systems literature and implement scheduling policies (such as EDF[76], fixed-priority[77] etc.) to achieve e2e delay guarantees.

Since scheduling policies are subject to the baseband traffic parameters, any addition of radios or changes in them requires policy changes in the entire network, making the design unscalable. Therefore, the tree structure in DISTRO is partitioned into an aggregation- and an edge- switch network; the schedulability analysis is done only at the edge-switch network whereas the default packet scheduling (e.g., FIFO) is implemented in the aggregation switch network. This logical division along with the tree-based design enables transport scaling to a large number of radios.

The quantization widths in DISTRO are selected to maximize the wireless capacity while ensuring e2e schedulability. A brute-force search of optimal quantization widths has exponential complexity in the number of radios. However, using the monotonic dependence of the wireless capacity and the schedulability on the quantization widths we propose a greedy-based approach that has much less complexity in practice.

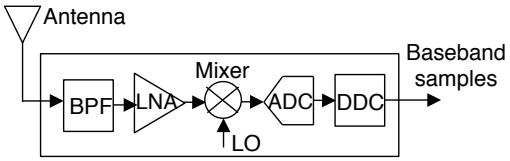


Figure 4.1: Radio front-end design for baseband conversion.

```
90213 3.112460000   1514 UDP
90214 3.112463000   1514 UDP
90215 3.112467000   1514 UDP
90216 3.112470000   1514 UDP
90217 3.112474000   1514 UDP
90218 3.112477000   1514 UDP
90219 3.112481000   1514 UDP
```

Figure 4.2: USRP2 transport log at 25MHz sampling rate.

In summary, we make the following contributions:

- Proposal of a Fat-Tree architecture for scalable deployment of baseband radios;
- Calculation of the maximum delay bound of a baseband packet in the network and its use to achieve e2e schedulability; and
- Characterization of the wireless capacity under the schedulability constraint and development of an efficient search algorithm to maximize the capacity.

The rest of this chapter proceeds as follows. Sec. 4.2 provides the background on the baseband transport. Sec. 4.3 presents our proposed transport design and its evaluation results in a simulated network scenario. In Sec. 4.4, we obtain the maximum wireless capacity with end-to-end schedulability. Finally, Sec. 4.5 presents the related work and Sec. 4.6 concludes the chapter.

4.2 Background

This section provides the background on baseband transport and processing, and describes how real-time scheduling is applicable to baseband transport.

4.2.1 Wireless Baseband Transport

The radio front-ends in a baseband network (also known as RRHs in C-RAN) act as converters between RF samples and complex (I and Q) baseband samples. Fig. 4.1 shows the components of such a radio. In the receive mode, the RF signal is down-converted, filtered and passed through an analog-to-digital converter (ADC) that gives out a digitized (e.g. 16-bit) stream of baseband samples. This stream is broken into fixed-size blocks, which are then transported as payloads in special-purpose packets generated with appropriate headers and tags.

Suppose there are n radios in the network, where each radio is denoted by index $i \in [1, n]$. Let f_i denote the desired sampling frequency from the ADC (achieved through decimation by digital-down converters), and let Q_i denote the number of bits used to represent each I (and Q) baseband sample. Then, the transport data rate (in bits/s) of radio i can be expressed as :

$$R_i = 2Q_i f_i. \tag{4.1}$$

Further, let B denote the fixed payload size (in bits) of a transport packet. The inter-packet arrival time (in seconds) at radio i, assuming negligible packet overhead, is given by:

$$T_i = \frac{B}{R_i}. \tag{4.2}$$

Since the ADC operates at a fixed frequency, and fixed-size blocks are used, the packet inter-arrival time is a constant at each radio, which we refer to as the *period* of the arrival process.

Example:USRP is a common software-radio platform that uses the UDP protocol for baseband transport. Fig. 4.2 shows the timestamps from the transport log of a USRP2 running at 25MHz sampling rate and payload size of 1492 Bytes. Using Eqs. (4.1) and (4.2), the packet inter-arrival time with 8–bit quantization is calculated

to be 2.98μs. This is indeed close to inter-packet arrival time $\in [3,\ 4]\mu$s observed from the USRP2 logs (Fig. 4.2). Note that the logs are not exactly periodic because of the minimum 1μs resolution and random lag in the packet capture.

Wireless protocols have a fixed e2e processing deadline for PHY-layer primitives such as channel sensing and decoding. Assuming a fixed (or worst-case) processing time at the baseband processors, the e2e PHY deadlines impose a maximum transport delay. Therefore, in order to support real-time processing, the generated baseband packets from the radios must be transported to the baseband processors within a fixed amount of time. That is, each radio i has an e2e transport delay bound, D_i, that the transport network must satisfy.

The traffic specification of radio i is thus given by a 2–tuple $\tau_i = (T_i, D_i)$, which represents the inter-arrival time, and the e2e delay bound, respectively. The radio traffic is said to be *schedulable* if for all $1 \leq i \leq n$, the maximum delay experienced by a packet of radio i is not greater than the requested delay bound D_i.

The transport network in large deployments of C-RAN runs over a fiber infrastructure such as dark fiber and WDM [12]. While in indoor environments, the radios can be connected using high-capacity Ethernet or Infiniband links [10]. In both scenarios, baseband samples are exchanged through packet transmissions (most optical networks now offer packet switching for increased flexibility). Thus, the baseband transport network can be modeled as a packet-switch network with one or more network switches. Every packet in the network passes through multiple links, switches, and routers before reaching its destination. While many routes can exist for a packet, for simplicity, we assume fixed routing in the network, which is necessary for e2e delay guarantees [78].

Despite its advantages, packet-switching introduces various delays at each link in a selected route. The e2e packet delay is the summation of delays over links and switches along the selected route, which is composed of:

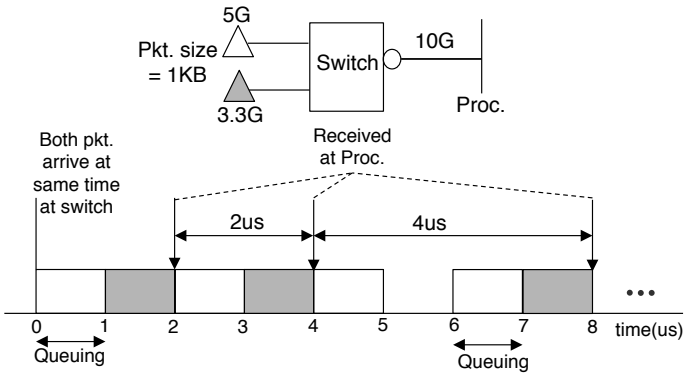


Figure 4.3: Switch FIFO output for two periodic flows. The output sequence is non-periodic.

- Propagation delay (t_p): the time taken for the packet to reach the next switch;
- Switching delay (t_s): the time taken for the packet to move from the ingress to egress port of a switch;
- Transmission delay: the time needed to transmit the packet, which is a function of the link capacity; and
- Queuing delay: the waiting time of packet in a switch's egress queue.

Among the different delays, the queuing delay is the only unknown that can be different for each packet in the network. In general, it is a function of the switch's scheduling policy and the input arrival sequence. One needs to model these delays at each switching stage, which becomes intractable for large networks as the output sequence from a switch is non-periodic even though packets arrive periodically. For example, consider a simple baseband network (Fig. 4.3) with periodic baseband traffic from 2 radios having inter-arrival times of 2μs and 3μs, respectively. Assume the processing link capacity is 10Gbps and the packet size is 1000 bytes. That is, the output transmission time of both flows is 1μs. Assume packets from the two radios arrive in the queue at the same time instant in the beginning. Fig. 4.3 shows the timeline of the queue output. As one can see, the inter-arrival time of the 3.3Gbps

flow at the output is non-periodic (inter-arrival times of 2μs and 4μs) that is induced from the waiting in the queue. This non-linearity of queuing makes it difficult to model e2e packet arrivals, a fact well-known in the literature [74].

Common packet-switching techniques such as First-In-First-Out (FIFO) and Round-Robin (RR), which are designed for best-effort traffic flows, are not suitable for the real-time traffic that requires e2e delay guarantees. To support real-time baseband traffic, the switches can use various scheduling policies that were developed by real-time systems researchers [11, 74]. Among them, the deadline scheduling is a natural approach where each arriving packet is assigned a deadline according to the requested delay bound. The packet with the earliest deadline is transmitted first. This scheduler is optimal in the sense that if packets meet their deadlines using any scheduling policy, so will they using deadline scheduling. While the original deadline scheduling considered implicit deadlines (deadline is the same as the period) and preemption, one can generalize it further to obtain both necessary and sufficient conditions for schedulability with arbitrary deadlines. Theorem A.1 formally states these conditions for both cases, with and without preemption.

Deadline scheduling uses dynamic prioritization and thus difficult to realize in practice. A more feasible approach is the fixed-priority scheduling where each traffic flow is assigned a static priority [77], and incoming packets are transmitted in the order of their priority. As shown in Theorem A.2, there exists a schedulability test to determine whether the set of traffic flows with given priorities meet their delay bounds. Furthermore, one can do an iterative or offline search and use the schedulability criteria to arrive at the feasible priority assignment policy if one can be found [79]. The necessary conditions for schedulability, however, are known only under special circumstances (e.g., when the deadline is equal to the period).

4.2.2 Transport Delay Bound

The wireless processing design depends on the wireless protocol and the target architecture. For WiFi signals, where slots are 9μs long, baseband samples are streamed and decoded on the fly [72]. In contrast, LTE has $1ms$-long subframes and decoding is carried out on an accumulated buffer of baseband samples [73]. The time required to decode the baseband samples (or frames) depends on the capability and the optimizations of the platform. In this chapter, we assume the processing time is fixed, and focus on the transport delays.

The transport delay bound of radio i is computed by subtracting the maximum processing time, T_{proc}, from its end-to-end protocol deadline, T_{prot}, as:

$$D_i = T_{prot} - T_{proc}. \tag{4.3}$$

In case of WiFi signals, the protocol deadline, T_{prot}, can be 4μ (since CCA assert should occur within 4μs during energy sensing [3]). Assuming $T_{proc} = 2\mu$s to perform sample summation, this results in a 2μs delay bound for the transport network. On the other hand, for LTE signals, $T_{prot} = 2$ms, as there is no channel sensing, and the HARQ process governs the protocol deadline. Consequently, the delay bound is much larger (0.5–0.7 ms) than the WiFi case.

It is worth noting that the transport delay bound, D_i, is not always fixed but can vary with the number of radios, n, since the processing time, T_{proc}, typically increases with n. For instance, $T_{proc} \propto O(n^3)$, when decoding n spatially multiplexed signals [18].

4.3 DISTRO

This section describes the construction, requirements, and analysis of our proposed real-time transport design for baseband traffic.

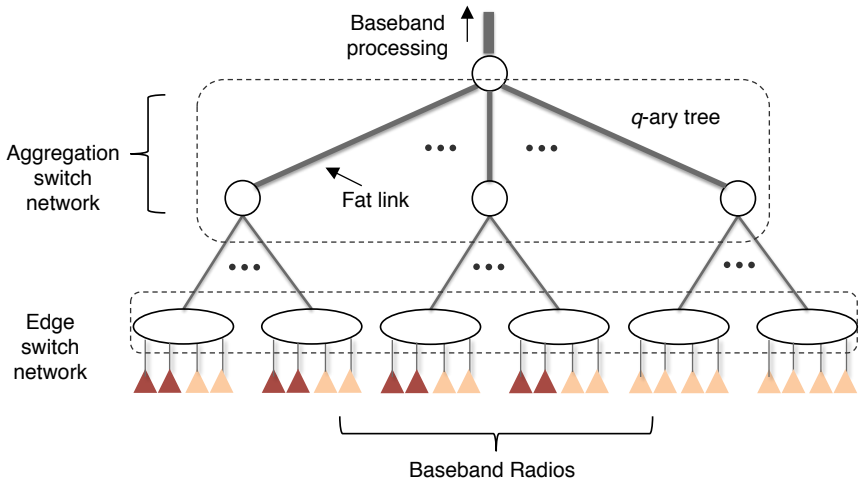


Figure 4.4: Fat-Tree architecture of DISTRO with heterogeneous radios.

4.3.1 Design Philosophy

The design of a baseband network is driven by two key observations. First, the baseband traffic flows exclusively between the radios and the processor pool (cross-traffic between radios is negligible). Second, depending on the application, baseband traffic is aggregated into one or more links for processing.

A tree-based design, therefore, is a good fit for baseband transport. Moreover, the baseband transport operates in real time. Specifically, given the traffic flows and delay bounds, we must be able to give a sufficient, if not necessary, condition to check their e2e schedulability.

We propose DISTRO, a baseband network design that combines the tree structure with real-time scheduling. Fig. 4.4 illustrates the proposed design in a deployment of heterogeneous radios. The radios are connected to edge switches, and the links from the edge switches are aggregated at multiple levels until the root switch. The destination is assumed to be located at the root switch, which is the common aggregation point for the baseband packets. The destination could is assumed to be a physical point that is connected to a pool of baseband processors.

DISTRO's design accommodates an increasing number of radios without severely

affecting their delay performance, and makes the schedulability flexible to the addition and removal of radios. It partitions the baseband network into two components: edge- and aggregation- switch network. The edge switch network contains the edge switches that form the first entry point of a baseband flow. From a deployment standpoint, each edge switch could connect a group of radios that are in physical proximity of each other, for instance, a basestation site in a cellular network. The aggregation network contains all the remaining switches except the edge switches, and its purpose is to aggregate traffic from the edge towards the destination.

4.3.2 Design Requirements

The aggregation network in DISTRO is a logical tree of links and switches. To simplify the schedulability analysis, we place the following restrictions on its design.
1) Fat-Tree. A tree is a Fat-Tree if for every switch in the tree, the switch's uplink capacity is greater than or equal to the sum capacity of the incoming links.
2) Symmetric. A tree is symmetric if for every switch, interchanging the incoming links results in the same tree.
3) Non-preemptive. The network switches always use a non-preemptive scheduling policy, i.e., an ongoing packet transmission is never preempted.
4) Equal packet sizes. The baseband packet sizes in the network are always equal irrespective of their source.

4.3.3 End-to-End Guarantees

As noted earlier, a packet can experience a queuing delay at any of the switches in the network. Beyond the edge switch, characterizing the queuing behavior becomes difficult as the packet arrivals are no longer periodic [74]. However, under our symmetric Fat-Tree construction, we can easily bound the maximum queuing time.

Assume K edge switches, and let $S_i, i = 1, 2, \ldots K$, denote the set of radios con-

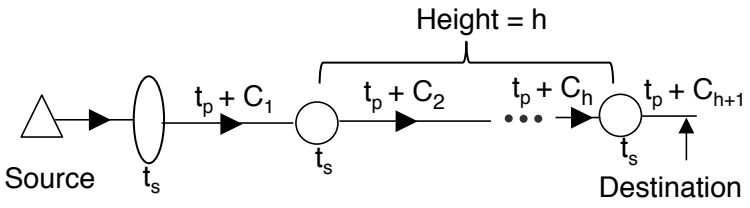
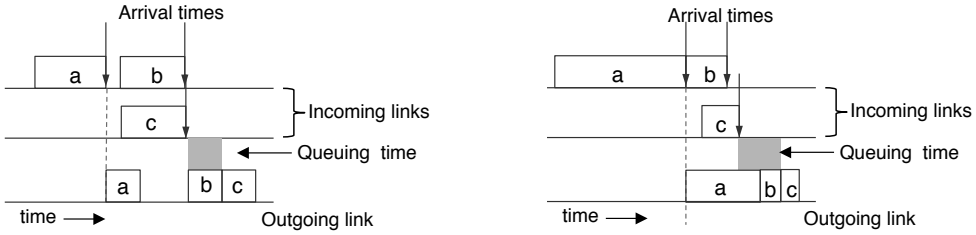


Figure 4.5: Path of baseband packet from the source to the destination.



(a) Equal transmission times

(b) Unequal transmission times

Figure 4.6: Blocking of c by at most one packet (left) and more than one packet (right).

nected to edge switch k where $S_k \subseteq \{1, 2 \ldots, n\}$. For simplicity, assume at time $t = 0$ packets from radios of each edge switch are queued at the switch, and the packets arrive periodically at the switch thereafter. Further, assume no transmission or propagation delay from the radio to the edge switch. Let C_1 denote the transmission time of a packet on the link connecting the edge switch and the next aggregation switch. Since packet size, B, is fixed for the network, every baseband packet has the same transmission time going through the edge switch.

For simplicity, assume a binary aggregation tree. Let the transmission time sequence as a packet traverses from the edge to the destination be represented as $C_1, C_2, \ldots, C_{h+1}$, where h is the height of the aggregation tree. We define $h = 1$ if there is only one aggregation switch.

From the symmetric Fat-Tree definition, it holds that if C_j is the transmission time on the incoming link, then the transmission time on the aggregation link, C_{j+1}, for all $j, 1 \leq j \leq h$, satisfies:

$$C_{j+1} \leq C_j/2. \tag{4.4}$$

Let us assume the edge-switch uses a non-preemptive scheduling policy whereas the aggregation switches use FIFO scheduling. Under non-preemptive scheduling, the inter-arrival time of the incoming packets from the edge is equal to or larger than C_1. Consider the first aggregation switch: the outgoing transmission time is $C_2 (C_2 \leq C_1/2)$. The packets in each of the two incoming links have at least C_1 time separation. Therefore, any packet will wait for at most C_2 time in the queue, which occurs when two packets arrive at the same time. This is illustrated in Fig. 4.6(a), where packet c arrives at the same instant as packet b, and is blocked by the transmission time of packet b.

Similarly, for every j, packets in the two incoming links with transmission time C_j and inter-arrival time greater than C_j, will have a maximum queuing delay of C_{j+1}. Therefore, for the path as shown in Fig. 4.5, the maximum total delay of *any* baseband packet across the aggregation network is bounded by:

$$\begin{aligned} T_a = \sum_{j=2}^{h+1}(t_s + C_j + C_j + t_p) &= h(t_s + t_p) + \sum_{j=2}^{h+1} 2C_j \\ &\leq h(t_s + t_p) + \sum_{j=2}^{h+1} \frac{C_1}{2^{j-2}} \\ &= h(t_s + t_p) + 2(1 - 2^{-h})C_1 \end{aligned}$$

where we use the inequality $C_j \leq C_{j-1}/2 \leq \ldots \leq C_1/2^{j-1}$ using Eq. (4.4).

Now, in order for a baseband packet of radio i to meet its e2e delay bound D_i, the total packet delay at its edge switch must be less than D_i less the maximum aggregation delay. Adding the switching and propagation delays of the edge switch, we arrive at the e2e schedulability of all baseband flows in the network by checking the schedulability of baseband traffic at each edge switch.

Theorem IV.1. *In a q-ary symmetric fat-aggregation-tree of height h, the radio traffic* $\tau_i = (T_i, D_i), i = 1, 2, \ldots, n$, *is schedulable, if for every* $k, 1 \leq k \leq K$, *the set*

of traffic flows $\tau_i = (T_i, d_i^{'}), i \in S_k$*, with transmission time* C_1 *and no preemption, is schedulable at edge switch* k*, where* $d_i' = D_i - \frac{1-q^{-h}}{1-q^{-1}} C_1 - (h+1)(t_s + t_p)$.

Proof. By extension of the binary tree analysis to q-ary tree and noting that the maximum queuing delay of a packet on a link with transmission time C_j is $(q-1)C_j$. □

Theorem IV.1 provides only a sufficient condition for schedulability, but does not affirmatively tell us if a given set of traffic flows are schedulable. Nevertheless, the result is still useful as it allows us to construct a transport network that guarantees to meet the requested e2e delay bounds. Furthermore, it gives the scheduling policies that the network must implement: non-preemptive scheduling at edge switches, and regular FIFO scheduling at aggregation switches.

The reason for restricting our construction to a symmetric Fat-Tree and non-preemptive scheduling is simple: the packet transmission times are equal for the incoming links, and therefore, the maximum queuing delay in the aggregation network is easy to obtain. This is not the case, however, if we assume unequal transmission times, which happens if unequal links are used or when preemption is allowed. For instance, Fig. 4.6(b) shows the blocking of a smaller packet, c, due to the transmission of two previous larger packets. This example is akin to the head-of-line (HOL) blocking problem that occurs when subsequent transmissions are held up by the first transmission [80]. Then, to arrive at the maximum queuing time in the network, one must examine different arrival sequences and their transmission times at each switch, which can quickly become intractable.

4.3.4 Scalability

The design of DISTRO is scalable on three fronts. First, as the number of radios in the network, n, grows, and the number of edge switches, K, increases, the

maximum total delay of a packet increases only logarithmically. To see this, one can upper bound the total aggregation delay as follows:

$$T_a < h(t_s + t_p) + 2C_1. \tag{4.5}$$

The delay bound grows linearly with the height, h, which is always less than or equal to $\lceil \log_2 K \rceil$. For large packets (e.g., Jumbo Ethernet frames), the transmission time is much larger than the switching and propagation time. In this case, the delay scaling factor, $t_s + t_p$, becomes even less significant.

Second, DISTRO supports heterogeneous baseband radios with differing periods and deadlines. There is no restriction on the type of radio protocol. This aids the deployment of a radio access network over multiple wireless standards, such as LTE and WiFi, at the same physical location.

Finally, each edge switch, k, implements a schedulability test locally that is dependent only on the traffic generated from the set of radios, S_k, connected to it. Any addition or removal of radios requires one to check only local schedulability, without disturbing the performance of other flows. This feature enables incremental deployment of the baseband network.

4.3.5 Run-time Scheduling

The aggregation network in DISTRO implements the default FIFO scheduling. On the other hand, each edge switch implements a non-preemptive scheduler which can either be EDF or fixed-priority. The fixed-priority scheduler is easy to realize with multiple outbound queues at a switch. Each incoming packet is first classified and placed in its corresponding queue, and the queues are then dequeued in the order of their priority.

Background Traffic. Background traffic such as control messages can disrupt the

t_s	50ns	Edge–src.	10Gbps
t_p	10ns	Agg.–Edge	10Gbps
Pkt. size (B)	1KB	Agg.–Core	40Gbps
q-ary tree	2,3,4	Core–Dest.	200Gbps
height (h)	2	Flow rate (R_i)	1, 1.5, 2, 2.5 Gbps
Radios/edge	4	Simulation Time	1sec

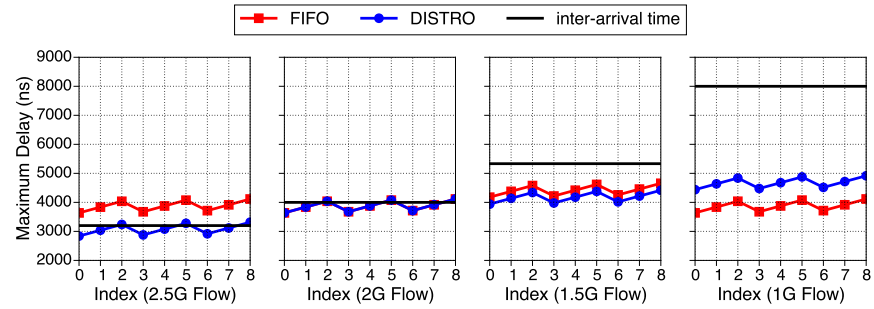
Table 4.1: Simulation parameters for schedulability analysis

real-time performance of the network. To ensure minimum disruption, we segregate the baseband traffic from the rest of the traffic through flow prioritization. Background and rest of the traffic are placed in a separate queue, which has lower priority than baseband traffic. Since background traffic adds to the non-preemptive delay at each switch, we update the delay bound in Eq. (4.5) accordingly.

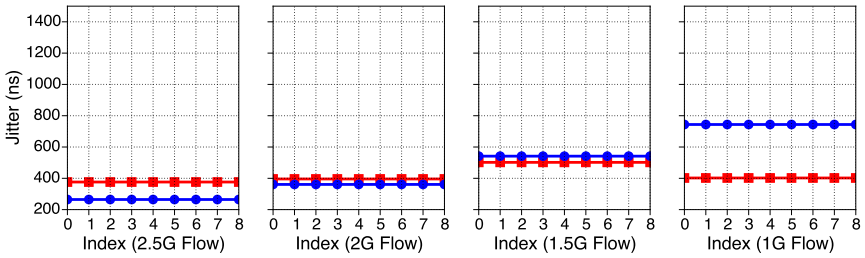
4.3.6 Evaluation

We implement and evaluate DISTRO's Fat-Tree architecture using NS-3[39]. NS–3 is a discrete-event network simulator that accurately simulates network traffic in large deployments. Table 4.1 shows the simulation parameters used in our setup. Each edge switch connects 4 heterogeneous radios that have different flow rates (1, 1.5, 2 and 2.5G) but fixed packet size of 1000 bytes. We simulate a fat-aggregation-tree with three levels: 1) a core switch that is connected to the destination through 200Gbps link; 2) q aggregation switches connected to core switch with 40Gbps links; 3) q edge switches connected to each aggregation switch with 10Gbps links. Thus, the total number of radios in our setup is $4q^2$.

We use the NS–3 packet tagging mechanism to tag each baseband packet with a priority level. The priority levels are chosen to achieve e2e schedulability (Sec. 4.3.3). We then implement a packet-classifier at the edge switch that classifies the incoming packet and places it in one of the 4 egress queues. For dequeuing, the queues are searched in decreasing order of their priority. For simplicity, we assume the transport delay-bound is same as the inter-arrival period of each flow. Under this assumption,



(a) Maximum Delay



(b) Jitter

Figure 4.7: Delay metrics for flows in a 36-radio setup. DISTRO meets the end-to-end guarantees of all flows.

the priorities are determined by the inverse of the inter-arrival period (equivalent to a rate monotonic scheduler).

Fig. 4.7 shows the maximum e2e transport delays and jitter observed in a 3-*ary* tree with 36 radios. Here, we compare DISTRO's scheduling with the basic FIFO packet scheduling. As seen from Fig. 4.7(a), DISTRO meets the e2e guarantees (delay $<$ inter-arrival time) of each flow while the FIFO scheduling misses the delay bound of the 2.5G flow. In prioritized scheduling, the 1G flow has the least priority and therefore sees an increase in its transport delay.

In Fig. 4.8, we show the maximum observed flow delays as we increase the number of radios in the network. We assuming a fixed number of radios per edge switch. The scaling is achieved by using q^2 edge switches, and q aggregation switches for a q-ary aggregation tree. The core switch capacity, however, is fixed at 200Gbps. Despite quadrupling the number of radios (from 16 to 64), the maximum delay in the network increases not more than $300ns$ (a less than 10% increase). This can be explained by

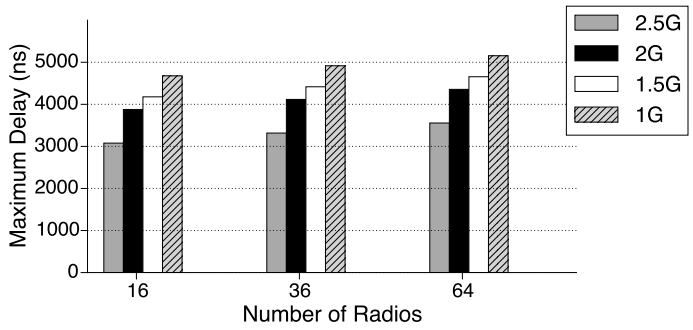


Figure 4.8: Maximum end-to-end delay of flows with increasing number of radios.

the maximum delay bound that is dependent only on the height of the tree, and not on the number of radios. Note that this constant increase is not always the case, for example, in daisy-chained radio network in ARGOS [10], the maximum delay grows linearly with the number of radios.

4.4 Achievable Capacity Under E2E Schedulability

In this section, we characterize the achievable wireless capacity under the constraint of e2e schedulability. We consider the MIMO processing of baseband samples from n radios. The specifics of synchronization, buffering and decoding are omitted but are assumed to manifest through the requested delay bounds.

Fig. 4.9 shows our system model radios receive samples over a wireless channel and transport them using a single aggregation switch. Let $\mathbf{x} \in \mathbb{C}^{m\times 1}$ be the signal vector sent by the transmitter, and let $\mathbf{y} = \mathbb{C}^{n\times 1}$ be the received signal vector, where m is the number of transmit antennas, and n is the number of radios. Let $\mathbf{H} \in \mathbb{C}^{n\times m}$ represent the wireless channel between the transmitter and the radios.

Quantization. Assume radio i has ADC quantization width, Q_i, that takes an integer value from the set $\mathcal{L} = \{L_1, \ldots, L_d\}$, where L_1 and L_d are the minimum and the maximum quantization widths, respectively. Further, let $\mathbf{Q} = [Q_1, \ldots, Q_n]$, and let $\gamma(Q_i)$ be the average quantization noise power injected into the baseband samples,

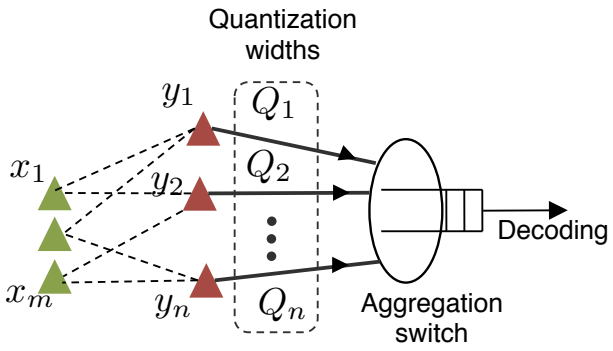


Figure 4.9: The system model for processing and scheduling.

where $\gamma(\cdot)$ is the quantization noise function.

Transport. Assume a fixed size B of baseband packets. From Eq. (4.1) and Eq. (4.2), the inter-arrival time at radio i is $T_i = \frac{B}{2Q_i f_i}$, which is a function of the radio quantization width, Q_i.

Model. Assuming a narrow-band channel, the received signal can be expressed as:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z} + \mathbf{z}_Q, \tag{4.6}$$

where $E[\mathbf{x}\mathbf{x}^H] = \rho\mathbf{I}_m$ and ρ denotes the transmitted power, that is equal across all symbols, and $\mathbf{z} \sim \mathcal{CN}(0, \sigma^2\mathbf{I}_n)$ is the additive complex white gaussian noise, and $\mathbf{z}_Q$ represents the quantization noise vector. We assume the effect of quantization manifests through the additive term $\mathbf{z}_Q$, which is approximated as a zero-mean complex Gaussian noise with covariance matrix $\mathbf{\Sigma}_Q = \text{diag}(\gamma(Q_i), \ldots, \gamma(Q_n))$. Let $\mathbf{\Sigma} = \sigma^2\mathbf{I}_n + \mathbf{\Sigma}_Q$ denote the equivalent noise covariance matrix.

The ergodic wireless capacity (in b/s/Hz) for a fixed $\mathbf{Q}$, with imperfect channel knowledge at the transmitter, is [81, Eq. 20]:

$$R(\mathbf{Q}) = \text{E}_{\mathbf{H}}[\log_2 \det(\mathbf{I} + \rho\mathbf{\Sigma}^{-1}\mathbf{H}\mathbf{H}^H)] \tag{4.7}$$

where the expectation $\text{E}_{\mathbf{H}}[\cdot]$ is taken over all channel realizations of $\mathbf{H}$.

4.4.1 Problem Formulation

We want to select the quantization, $\mathbf{Q}$, that ensures e2e schedulability, and also maximizes the wireless capacity. This is expressed through an optimization problem (OP):

$$\begin{aligned} \text{OP:} \quad & \max_{\mathbf{Q}\in\mathcal{L}^n} R(\mathbf{Q}) \\ & \text{s.t.} \tau_i(\mathbf{Q}) = (T_i, D_i), i = 1, \ldots, n, \text{is schedulable} \end{aligned}$$

where D_i is the transport delay bound of radio i. Next, we show the following properties of our optimization problem.

Proposition IV.2. *If $\gamma(\cdot)$ is monotonically decreasing, $R(\mathbf{Q})$ is monotonically increasing in $\mathbf{Q}$.*

Proof. Let $\mathbf{Z} = \mathbf{I}+\rho\mathbf{\Sigma}^{-1}\mathbf{H}\mathbf{H}^H$, $\mathbf{\Sigma}^{-1} = \text{diag}(\mu_1, \ldots, \mu_n)$, where $\mu_i = 1/(\sigma^2+\gamma(Q_i))$ for $i = 1, \ldots, n$, and let $\lambda_1, \ldots, \lambda_n$ denote the eigen values of $\mathbf{Z}$ such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. We follow an approach similar to [82, Theorem 10.3] for the proof. For a fixed i, we claim that λ_j, for every j, is monotonically increasing in μ_i. On the contrary, suppose this is not true and there exists an interval of μ_i, and k, $k \in \{1, \ldots, n\}$, such that λ_k increases and decreases in that interval. Consequently there exists $\lambda^{'}$ such that $(\lambda^{'} - \lambda_k)$ vanishes for at least two values of μ_i. Now write $\det(\lambda\mathbf{I}-\mathbf{Z}) = \Pi_{j=1}^{n}(\lambda-\lambda_j)$, Then, for $\lambda = \lambda^{'}$, $\det(\lambda\mathbf{I} - \mathbf{Z})$ vanishes for more than one value of μ_i, which is impossible, since $\det(\lambda\mathbf{I} - \mathbf{Z})$ is a linear polynomial in μ_i. Therefore if $\gamma(Q_i)$ strictly decreases with Q_i, μ_i strictly increases with Q_i, thus, $\forall j$, λ_j monotonically increases with Q_i, for any $i = 1, \ldots, n$. Since $\log_2 \det(\mathbf{Z}) = \sum_{j=1}^{n} \log_2 \lambda_j$, and expectation is a monotonic operator, therefore, $R(\mathbf{Q})$ is monotonically increasing in $\mathbf{Q}$. □

Proposition IV.3. *For quantization $\mathbf{Q}$, if the traffic $\tau_i = (T_i, D_i)$, $i = 1, \ldots, n$, is schedulable with the sufficient conditions of scheduling given in Theorems A.1–A.2,*

then the same traffic is schedulable by decreasing $\mathbf{Q}$.

Proof. For simplicity, we consider a non-preemptive EDF scheduler and show that the proposition holds. For any i, changing Q_i to $Q_i^{'}$ decreases the inter-arrival period from T_i to $T_i^{'}$. Since $T_i^{'} < T_i$ implies $\lceil (t-d_i)/T_i^{'} \rceil^+ \leq \lceil (t-d_i)/T_i \rceil^+$, $\forall t$, substituting in Eq. (4.9), it follows that $\tau_i(\mathbf{Q}^{'}) = (T_i^{'}, D_i)$, $i = 1, \ldots, n$, is schedulable if $\tau_i(\mathbf{Q}) = (T_i, D_i)$, $i = 1, \ldots, n$ is schedulable. □

4.4.2 Search Algorithm

The search space in OP is $\mathcal{L}^n$, hence, finding the optimal $\mathbf{Q}$ through a brute-force search has an exponential complexity. Note that we cannot relax the integral constraint because of the discreteness of the schedulability. However, from the monotonic dependence on $\mathbf{Q}$ (Propositions IV.2–IV.3), we can construct a greedy search algorithm.

The idea is to use breadth-first search (BFS) on the enumeration of the search space. Starting from the highest quantization, $[L_d, \ldots, L_d]$, we enumerate the next highest quantizations, $[L_{d-1}, L_d, \ldots, L_d]$, …, $[L_d, L_d, \ldots, L_{d-1}]$, and then enumerate the next highest quantization for each of them, and so on. More generally, let us define the function $enum(\cdot)$, for quantization, $\mathbf{Q} = [L_{k_1}, L_{k_2}, \ldots, L_{k_n}]$, as follows:

$$enum(\mathbf{Q}) = \{[L_{k_1-1}, L_{k_2}, \ldots, L_{k_n}], [L_{k_1}, L_{k_2-1}, \ldots, L_{k_n}], \ldots, [L_{k_1}, L_{k_2}, \ldots, L_{k_n-1}]\}. \tag{4.8}$$

As we enumerate each possible quantization, we check its shcedulability (according to Theorems A.1–A.2). If it is schedulable, we calculate the corresponding capacity, but do not enumerate its further. Finally, from the calculated capacities, we find quantization $\mathbf{Q}^\star$ that achieves the maximum capacity.

Algorithm 3 gives the pseudo-code for implementing our proposed solution. It uses

Algorithm 3 BFS search

1: *Initialize*: $\mathbf{Q}^\star \leftarrow \phi$, $C^\star \leftarrow 0$, U $\leftarrow [L_d, \ldots, L_d]$
2: *Returns*: $C^\star$, $\mathbf{Q}^\star$, optimal capacity and quantization
3: **while** U is not empty **do** ▷ breadth-first traversal
4: $\quad \mathbf{Q}^{'} \leftarrow$ U.pop()
5: $\quad$ **if** $\mathbf{Q}^{'}$ is schedulable **then** ▷ check e2e schedulability
6: $\quad\quad$ **if** $R(\mathbf{Q}^{'}) > C^\star$ **then**
7: $\quad\quad\quad C^\star \leftarrow R(\mathbf{Q}^{'})$, $\mathbf{Q}^\star \leftarrow \mathbf{Q}^{'}$ ▷ maximum capacity till now
8: $\quad\quad$ **end if**
9: $\quad$ **else**
10: $\quad\quad$ **for** $\mathbf{T}$ in $enum(\mathbf{Q}')$ **do**
11: $\quad\quad\quad$ U.push($\mathbf{T}$) ▷ next highest quantization
12: $\quad\quad$ **end for**
13: $\quad$ **end if**
14: **end while**

B	1 KB	$\mathcal{L}$	$\{2, 4, 8\}$
ρ	0dB	σ^2	1
m	2	channel	rayleigh
$\gamma(x)$	$10\frac{\pi\sqrt{3}}{2}2^{-2x}$	f_i	20MHz

Table 4.2: Simulation parameters for capacity evaluation

a queue data structure, U, for breadth-first traversal. We show that this algorithm finds the optimal solution to OP. Though in the worst case it has the same $O(\mathcal{L}^n)$ complexity, the running time in practice is much less than a brute-force search.

Theorem IV.4. *Algorithm 3 gives the optimal solution to OP.*

Proof. We use the the construction of Algorithm 3 to prove that $\mathbf{Q}^\star$ is schedulable and achieves maximum capacity. The schedulability of $\mathbf{Q}^\star$ holds from Line 5. From Proposition IV.2, the capacity from enumeration (in Line 10) always decreases, and hence $C^\star$ will always be larger than the capacity of unenumerated quantizations (all of which are schedulable from Proposition IV.3). Also, from Line 7, $C^\star$ is the maximum of all enumerated quantizations that are schedulable. Therefore, $C^\star$ is the optimal capacity that is schedulable. □

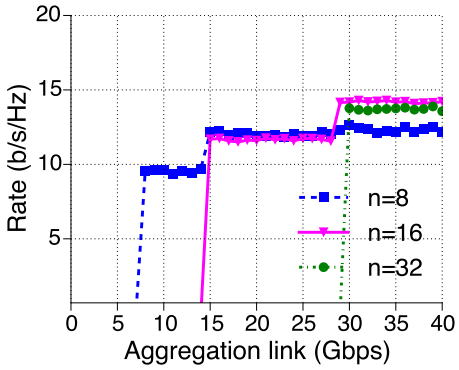


Figure 4.10: Wireless capacity as the aggregation link rate is varied.

4.4.3 Simulation Results

We use simulations to evaluate the wireless capacity and its dependence on e2e schedulability. Table 4.2 shows the simulation parameters used in our setup. Since we are interested in the queuing delays, we ignore the transmission time from radio to the switch, the propagation, and the switching delays. We consider a Rayleigh channel model, with two transmit antennas and SNR (without quantization) of 0 dB. Note that the quantization noise function, $\gamma(x)$, is monotonically decreasing in x. For ease of comparison, we set the delay bounds of all flows to be equal to their periods, i.e., $D_i = T_i$ for all $i = 1, \ldots, n$. We use a rate monotonic scheduler as the fixed priority scheduler.

Fig. 4.10 shows the maximum sum wireless rate averaged over 1000 channel realizations. Observe that as the aggregation link capacity increases, the transmission times decreases, and thus the schedulability supports higher quantization rates resulting in increased capacity. For example, when $n = 8$, the rate increases (from 2-bit to 4-bit quantization) at 14Gbps. Also, as we increase the number of radios from 8 to 32, we require at least 30Gbps, up from 14Gbps, link rate for the the baseband traffic with 4-bit quantization to meet their delay bounds.

4.5 Related Work

The concept of a Fat-Tree topology originated from the work on non-blocking switch networks. The topology is now widely used in datacenters for building scalable transport networks [75].

A large-scale MU-MIMO was realized using a distributed architecture of servers in [18]. The authors claim that bandwidth needed for baseband transport is not an issue as modern switches support up to 40Gbps links. However, they do not consider the real-time guarantees of transporting baseband samples. ARGOS [10] is another practical multi-antenna setup that suggests daisy-chained radios with Tree-based aggregation. However, no real-time analysis was provided there.

To eliminate transport jitter in fronthaul networks of a C-RAN, [83] proposes scheduled Ethernet using global scheduling. On the other hand, the authors of [84, 85] study the compression of baseband signals. They propose quantization schemes for lossy compression of the baseband samples. In [84], the authors propose prioritization of baseband frames, however, the priorities do not account for packet delays.

Real-time transport of Ethernet packets is a well-known problem in real-time systems literature [78, 74, 11]. While [78] provides a general approach for schedulability, the authors propose the use of flow regulators at each switch, which is difficult to realize in practice.

4.6 Conclusion

In this chapter, we have designed a baseband transport network based on Fat-Tree topology that is scalable to a large number of radios while meeting the e2e delay bounds. We have provided sufficient criteria for e2e schedulability, which is validated via simulations. While we only consider one aggregation link per switch, the design can be generalized to multiple aggregation links and cross-links for improved fault-

tolerance. The only requirement is that one should be able to bound the queuing time at each switch.

We also characterize the wireless capacity with the schedulability constraint, and provide an efficient search algorithm to maximize the capacity. Overall, our design can enable processing of wireless signals away from hardware while ensuring no performance loss from the underlying transport network.

4.7 Appendix

Theorem A.1. Under deadline scheduling, a set of traffic flows $\tau_i = (T_i, d_i), i = 1, 2, \ldots, n$, is schedulable on a link with transmission time C, and preemption if and only if [76, Theorem 1]:

$$\forall t > 0, \quad \frac{C}{t}\left(\sum_{i=1}^{n} \lceil (t - d_i)/T_i \rceil^+\right) \leq 1, \tag{4.9}$$

and without preemption if and only if [76, Theorem 6]:

$$\forall t \geq d_{min}, \quad \frac{C}{t}\left(1 + \sum_{i=1}^{n} \lceil (t - d_i)/T_i \rceil^+\right) \leq 1 \tag{4.10}$$

where $d_{min} = \min\{d_i : 1 \leq i \leq n\}$ and the function $\lceil x \rceil^+ = max(0, \lceil x \rceil)$.

Theorem A.2. Under fixed-priority scheduling, a set of traffic flows $\tau_i = (T_i, d_i), i = 1, 2, \ldots, n$, and priorities π_i, $i = 1, 2, \ldots, n$, such that $\pi_1 \geq \pi_2 \geq \ldots \geq \pi_n$, is schedulable on a link with transmission time C, if the function $W_m(k, x)$ satisfies:

$$\max_{1 \leq m \leq n} \max_{k \leq N_m} W_m(k, (k-1)T_m + d_m) \leq 1 \tag{4.11}$$

where $N_m = \min\{k : W_m(k, kT_m) \leq 1\}$, and function $W_m(k, x)$, for preemption, is

defined as [77]:

$$W_m(k,x) = \min_{0<t\leq x} \frac{C}{t}\left(k + \sum_{i=1}^{m-1} \lceil t/T_i \rceil\right), \tag{4.12}$$

and without preemption, $W_m(k,x)$ is defined as [86]:

$$W_m(k,x) = \min_{0<t\leq x} \frac{C}{t}\left(k + 1 + \sum_{i=1}^{m-1} (1 + \lfloor (t-C)/T_i \rfloor)\right) \tag{4.13}$$

CHAPTER V

Parallelism Meets Scheduling in Cloud-RAN Processing

5.1 Introduction

The baseband architecture of today's wireless networks is highly inefficient. Basestations use hardware that is usually proprietary, expensive, and difficult to upgrade. More importantly, the hardware resources (such as CPUs, DSPs, etc.) at each basestation are provisioned for its peak usage. This often results in severe resource-underutilization, as wireless traffic is known to exhibit significant spatial and temporal variations within a network [12]. Moreover, as network density increases from smaller cell sizes, the operating costs for the maintenance of the hardware (cooling, site visits, etc.) increases rapidly. Consequently, wireless operators are decoupling baseband processing from basestations and implementing it in a centralized pool of compute resources. The idea is to implement radio access network (RAN) functions in a cloud or a datacenter, where resources can be managed more efficiently. This approach, also known as *Cloud-RAN* (C-RAN), has received considerable attention from the industry as a way to reduce network costs [5, 12].

C-RAN attributes most of its advantages to resource pooling in which the aggregate load of a group of basestations is processed together. Previously, resource pooling

was shown to achieve more than 22% reduction in compute resources [19]. However, the biggest challenge in a C-RAN comes from the timing constraints of frame processing. For example, in LTE, a basestation must process a received subframe within a hard deadline of $3ms$ in order to send an acknowledgment.

The problem of meeting deadlines is fundamental tied to the design of the wireless frame processing, particularly, the scheduling of frame processing and the degree of parallel processing. A baseband scheduler must handle wireless frames that arrive periodically at a fixed rate (every $1ms$ in LTE), where each frame has a hard processing deadline. Besides, parallelism is another design dimension that lowers processing times and can enable real-time frame processing. A typical baseband chain consists of signal processing blocks, such as FFT, equalizer and decoder; each of these blocks can be broken down into independent (sub)tasks that can execute concurrently. For instance, FFT operations can run in parallel across antennas and symbols, while well-known parallel algorithms can be applied to Viterbi and Turbo decoding.

State-of-the-Art. Existing solutions utilize different scheduling schemes to meet the C-RAN's timing constraints; these schemes fall under *partitioned* scheduling [19, 87, 88]. Partitioned schedulers employ an upper bound on the frames' processing time (a.k.a. the worst-case execution time (WCET)) as a fixed processing time, which enables the design of optimal scheduling of basestations on multiple processors — a problem known to be NP-complete [89, 90, 91, 92, 93, 94, 95].

On the other hand, one could also use a *global* scheduler which maintains a shared queue and assigns each incoming frame to the next available core according to a priority mechanism, such as FIFO or earliest-deadline-first (EDF). Global schedulers are flexible in that they adapt to the available number of cores and the processing time variations [91, 92, 93].

Other existing systems [96, 18] exploit parallelism by splitting the baseband processing into parallel subtasks that can be executed on a large number of CPU/GPU

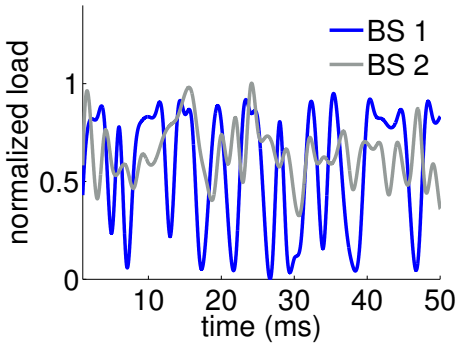


Figure 5.1: Variations in cellular load of two basestations.

cores. By achieving fined-grained parallelism and thus very small processing times, the problem of scheduling incoming frames becomes straightforward. For instance, parallelism can be used with existing partitioned schedulers through a variant, called *partitioned-parallel*, where a scheduled task makes use of additional cores for parallel processing. This approach, however, leads to over-provisioning of compute resources and increases the operating costs, thereby negating the benefits of C-RAN.

Shortcomings. The assumption of fixed processing time, albeit simplifying assumption, does not hold in practice. Both basestation traffic and wireless channel exhibit large temporal and spatial variations that result in varying subframe processing times. For example, Fig. 5.1 shows the load variations of Band–13 and Band–17 LTE basestations (in downlink) measured over a $50ms$ interval in a metropolitan region. As shown in the figure, the load varies considerably between two consecutive subframes that are transmitted every $1ms$. The load variations are also seen across the two basestations. Therefore, a partitioned scheduler that relies on WCET (corresponding to peak load), will over-provision compute resources [97]. A global scheduler avoids the pitfalls of the partitioned scheduler by adapting to the variable processing time and available compute resources. Nevertheless, such schedulers are more complex to implement, and suffer from high runtime overhead [98].

Proposed Approach. To address the shortcomings of existing schedulers, we propose RT-OPEX (*Real-Time OPportunistic EXecution*), a new framework that com-

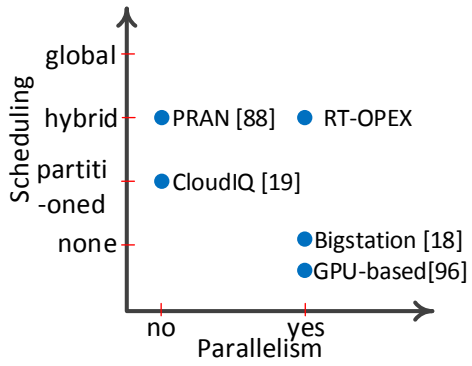


Figure 5.2: Existing approaches for baseband processing compared with ours.

bines offline scheduling with runtime parallelism. It minimizes deadline-misses of wireless frame processing by utilizing free CPU cycles at runtime to migrate parallelizable tasks to idle cores. The design of RT-OPEX is based on the premise that partitioned scheduling is unable to exploit other cores' free CPU cycles, while designing a scheduling algorithm with parallel processing is highly intractable. Hence, RT-OPEX treads the middle path and combines the flexibility of global schedules with determinism of partitioned schedules. Fig. 5.2 illustrates this comparison over existing approaches, which have either relied on scheduling, or on parallelism, but not both. We claim that RT-OPEX is highly effective in reducing deadline misses in C-RAN where frame execution times are highly variable across basestations.

RT-OPEX vs. Resource Pooling. Existing C-RAN literature [19, 12] has suggested resource pooling in which the statistical information of basestation loads is utilized to aggregate processing. We highlight that RT-OPEX is another variant of resource pooling, except that it consolidates processing at much finer timescales. Particularly, it utilizes the load variations of the order of subframes ($1ms$) to migrate processing on the compute platform, and as a result, maximizes the utilization of the available resources. However, unlike resource pooling, it has the advantage of making no assumptions about the prior knowledge of the load and traffic variations.

We evaluate the performance of RT-OPEX with other well-known schedulers:

partitioned, partitioned-parallel, and global. For accurate evaluation, we implement a medium-scale cloud-RAN-type platform comprising 16 radios, off-the-shelf GPP platform, and Ethernet infrastructure. We profile our implementation to develop an end-to-end (e2e) model for processing that includes transport and processing latency of a wireless frame. We also develop and release an open-source tool, *SchedTool*[1], that enables cellular operators to compare the performance of different C-RAN schedulers. It empowers the operators to conduct an extensive evaluation of deadline-misses and system capacity in addition to the computational and memory usage for each scheduler.

Our results reveal that RT-OPEX, compared to existing partitioned and global scheduling schemes, reduces the deadline-miss rate by more than orders-of-magnitude and increases the system capacity by 15%. As a result, RT-OPEX strictly outperforms partitioned scheduling schemes while offering the flexibility of a global scheduler.

Following are the main contributions of this chapter:

- development of an e2e model for characterizing the wireless processing times and the deadline-miss event;
- implementation and evaluation of different schedulers for a medium-scale C-RAN under realistic workloads and scenarios;
- design, implementation, and evaluation of a novel scheduling algorithm, RT-OPEX, which reduces deadline-misses by combining scheduling and parallelism; and
- an open source tool, SchedTool, that assists C-RAN operators in their quest for a scheduler.

The rest of the chapter is organized as follows. We present an end-to-end model of

[1]https://github.com/gkchai/garud

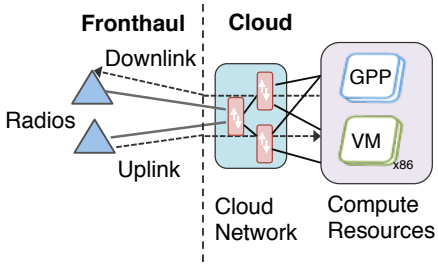


Figure 5.3: C-RAN system model for wireless processing.

C-RAN, including processing and transport in Section 5.2. Section 5.3 describes the design and implementation of possible schedulers including our proposed RT-OPEX scheduler. Section 5.4 presents the evaluation results of our implementation along with the details of the evaluation platform. In Section 5.5, we provide a qualitative comparison of the implemented schedulers. Finally, we discuss related work in Section 5.6 and conclude the chapter in Section 5.7.

5.2 End-to-End Model

In this work, we assume a pure software-based C-RAN where the entire baseband processing (or L1) is carried out on general-purpose processors (GPPs) or virtual machines (VMs). Fig. 5.3 shows the main elements in a C-RAN deployment where baseband (or IQ) samples from the radios are transported back and forth over a fronthaul network. In this section, we present a model to calculate the uplink processing time that allows us to characterize the deadline-miss event from an end-to-end perspective. This model is then used to develop C-RAN scheduling algorithms. We restrict our attention to uplink processing as it is significantly more time-consuming and varying than downlink [19, 73].

5.2.1 Uplink processing

A basestation, in the uplink, processes wireless signals from multiple antennas to decode user information. The basic unit of processing in LTE is a subframe ($1ms$

long). Each subframe is a sequence of 14 OFDM symbols, which is divided into multiple physical resource blocks (PRBs). These PRBs are then assigned to one or more users. Each user encodes information using a modulation and coding scheme (MCS) and transmits it in the allocated PRBs.

The computational load, and therefore, the time to process a subframe, is determined by the number of users, the number of antennas, allocation of PRBs to users, MCS assignment, and the number of decoder iterations required to decode user information. To capture this relationship, we present a linear model that accurately approximates the processing time. We first establish the mapping between the MCS and the number of data bits.

Let us consider the transmission of a single user and let the subcarrier load, D, denote the ratio of the number of data bits (packet size) in a subframe to the number of resource elements (REs) available in a subframe, where RE is the basic data carrier unit in an LTE subframe. The packet size as a function of number of PRBs and MCS is determined by a lookup table specified in the LTE standard [99]. For 10MHz bandwidth, which has 8400 REs, D varies from 0.16 to 3.7 bits per RE (for 50 PRBs), corresponding to MCS 0 and MCS 27, respectively. The maximum subcarrier load is 6 bits per RE when using 64-QAM modulation. For MCS 27, which uses 64-QAM, the load is much lower due to the overhead of coding, pilots and CRC bits.

LTE's uplink chain consists of commonly used signal processing blocks. To calculate the total processing time, one must model the dependence of each block on the number of antennas, subcarriers and other block-specific parameters. In general, this can be daunting as the uplink chain contains numerous blocks: FFT/IFFT, channel estimator, equalizer, demapper, descrambler, rate dematcher, and turbo decoder. However, we can construct a simple yet accurate model by making the following observations: (i) processing time of blocks that operate on OFDM symbol level (e.g.,

FFT, equalization), including the memory copy, varies linearly[2] with the number, N, of antennas; (ii) processing time of blocks using the constellation symbols (e.g., demapper, dematcher) is a function of the modulation order; (iii) processing time of decoder is determined by the number of iterations, denoted by L, and subcarrier load D, and noting that the decoder processes D bits per subcarrier in each iteration.

Thus, the total processing time can be written as:

$$T_{rxproc} = w_0 + w_1 \cdot N + w_2 \cdot K + w_3 \cdot D \cdot L + E, \tag{5.1}$$

where w_0, w_1 and w_2 are constants; E is the error term that includes modeling error and the variability of the execution environment; K is the modulation order of the MCS used. The constants in Eq. (5.1) are largely implementation and platform specific, as they depend on type of optimizations (e.g. vectorization) as well as the architecture.

The processing time depends indirectly on the wireless channel through the number of iterations, L, that are required to decode a packet, i.e., pass the CRC checksum. To avoid excessive delay, receivers typically limit decoding to at most L_m iterations. Therefore, irrespective of the channel condition, we obtain a WCET bound on processing time by substituting L with L_m in Eq. (5.1). Note that the number of iterations, L, is in general non-deterministic (even for fixed SNR) and may take any value in $[1,\ L_m]$.

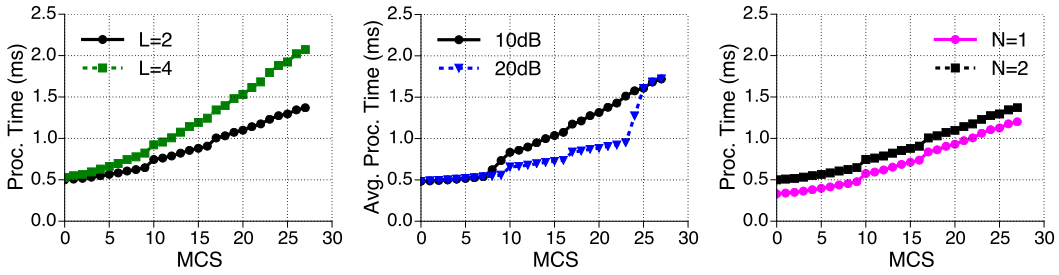
To validate our linear model, we collect data on the total uplink processing time of a 10MHz LTE system (50 PRBs) for different MCS (0–27), SNR values (0–30dB), and different number of antennas. The maximum number of turbo iterations, L_m, is set to 4. For each measurement, we note the load, D, and the iteration count, L, and then apply a linear regression to determine the model parameters.

Table 5.1 shows the model parameter estimates obtained from 4×10^6 measure-

[2]Assuming MRC equalization. With spatial multiplexing, the equalization complexity is N^2[18].

	w_0	w_1	w_2	w_3	r^2
GPP	31.4	169.1	49.7	93.0	0.992
KVM	31.1	186.6	56.5	101.3	0.982

Table 5.1: Model parameter estimates (in μs).



(a) vs. Iterations ($N = 2$) (b) vs. SNR ($N = 2$) (c) vs. Antennas

Figure 5.4: Plots showing the variations in processing time.

ments on a GPP and KVM [100] environment, respectively. We also show the goodness-of-fit metric, r^2, in each scenario. The fitness metric is observed to be close to 0.99, indicating the high accuracy of our model. Based on the model parameter estimates, we observe that each additional antenna adds 169μs while each turbo iteration at MCS 27 adds 345μs.

Fig. 5.4 shows the total processing time as we vary the number of iterations, SNR and number of antennas. From the plots, it is evident that the processing time exhibits high variability. For instance, it increases by a factor of 3 (from 0.5ms to 1.4ms) as MCS changes from 0 to 27 (Fig. 5.4(a)). Further, the total time with four iterations increases the total processing time by more than 0.5ms. This is consistent with observations made in previous studies [73, 96]. Note that in Fig. 5.4(b), decreasing the SNR from 20dB to 10dB increases the processing time by more than 50% between MCS 13 and 25. Similarly, for a fixed post-processing SNR, increasing the number of antennas to 2 adds a fixed 200μs to processing time.

In summary, we find that wireless processing is a dynamic workload that varies with data rates, channel conditions and the number of antennas used.

Platform Error. Fig. 5.5 shows the distribution of the error between the model

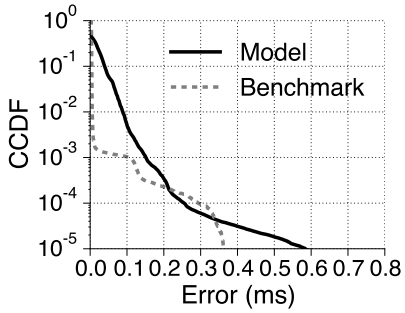
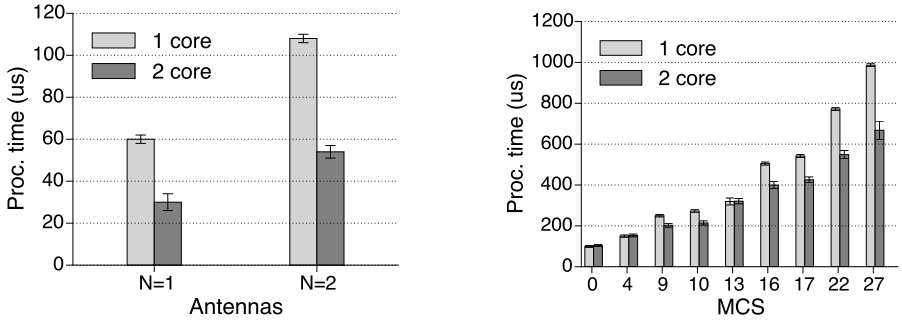


Figure 5.5: Distribution of model error (E) vs. benchmark error.

and the actual processing time. In 99.9% of observations, the error is less than $0.15ms$. However, for a few measurements the error can be as high as $0.7ms$. Since the processing runs on a soft-real time system (§5.4), the processing could be disrupted due to kernel tasks such as interrupt handling. The error term could thus be significant for some observations. Nevertheless, this could also be attributed to a large model error. To confirm otherwise, we perform a separate stress test on the processing platform. We ran the cyclictest [101] latency measurement tool alongside a benchmark load generated using hackbench [102]. The cyclictest was run with the highest system priority, and was expected to show a near-constant latency. Fig. 5.5 shows the latency distribution from the benchmark. The mean latency is $0.2ms$, but some of the measurements have a latency above $0.4ms$. We also observe that the *order statistics* of the modeling error is roughly similar to the benchmark latency. For example, 1 in 10^5 measurements had a latency of more than few hundred microseconds. This confirms that the distribution of error term in Eq. (5.1) is mostly influenced by the platform and not by the model error.

5.2.2 Parallelism

While the model in Eq. (5.1) provides the total processing time, it does not show the processing times of individual blocks. For simplicity, we assume the processing chain comprises of three sequential tasks: FFT, *demod*, and *decode*, where the demod



(a) FFT (b) Decode

Figure 5.6: Task execution times on multiple cores.

task comprises of channel estimation, channel equalizer and constellation-demapper; and decode task comprises of rate-dematcher, de-scrambler and Turbo decoder. Similar to Eq. (5.1), one can obtain a model for the processing time of each block.

So far, we have modeled processing time for a single thread running on a single core. However, it is possible to exploit different levels of parallelism, e.g., antenna-level, symbol-level and subcarrier-level parallelism, by making use of multiple CPU/GPU cores[96]. The FFT task that runs on each of the 14 OFDM symbols of each antenna, is easy to parallelize. Similarly, channel equalization that runs on each OFDM symbol can also be parallelized. Turbo decoding which is the most time consuming operation can be parallelized over code-blocks, where decoding and CRC check can be done independently on each code-block [99]. For instance, at MCS 27, LTE specifies 6 code-blocks all of which can be decoded concurrently.

Fig. 5.6 shows the processing times of FFT and decode tasks when it is parallelized over 2 cores. We are able to run FFT on 7 OFDM symbols on each core, and nearly halve the processing time (note the maximum overhead of 6μs). In the decode block, as seen in Fig. 5.6(b), parallelizing the Turbo decoding reduces the processing time by almost 320μs, from 980μs to 670μs.

Based on these observations, Fig. 5.7 shows a general breakdown of the processing of a subframe into tasks, and further into subtasks. For clarity, each task is shown

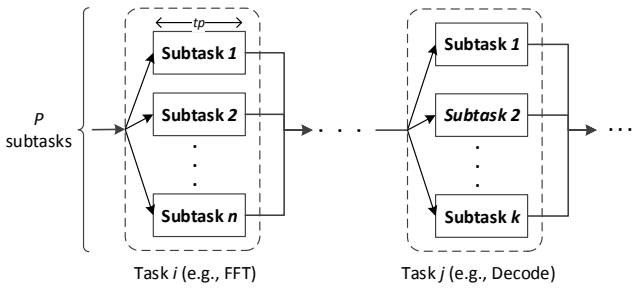


Figure 5.7: Breakdown of subframe processing into tasks and subtasks.

to be completely parallelizable, as one can always add another task block that contains serial tasks. Although a certain task might be parallelized, all of its subtasks must complete execution before moving on to the next stage. This establishes a dependency/precedence constraint between the different tasks involved in the subframe decoding. For the rest of the chapter, we assume that the execution time of tasks(subtasks) is deterministic (except for the decoder).

5.2.3 Transport Latency

The transport of IQ samples from the radios to the cloud involves two separate networks as shown in Fig. 5.3. The fronthaul network, deployed using an optical fiber network, connects the radios to an optical switch located in the cloud. Various standards such as CPRI [103] have been proposed for transport over fronthaul networks. In addition, a cloud network connects the optical switch to the pool of GPPs and VMs. The architecture of the cloud network is similar to a datacenter network (e.g., fat-tree topology) and includes aggregation and top-of-rack switches [12].

A wireless subframe incurs both fronthaul and cloud latencies. The fronthaul latency is a function of the length of the fiber (propagation time of light in fiber is approximately 5μs/Km) and the overhead of optical switching. While the exact fronthaul specifications are still under consideration, it is expected that the distance between remote radios and the cloud can be up to 20–40Km [5], resulting in a one-

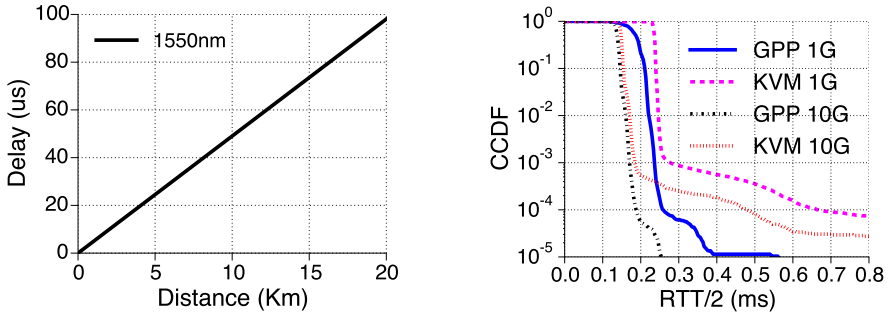


Figure 5.8: Fronthaul latency (left) and distribution of cloud network delay (right).

way propagation delay of 0.1–0.2ms excluding the overheads of (de)-packetization and cloud transport delay. For example, Fig. 5.8 shows the fronthaul latency as a function of distance for a single-mode fiber with 1550nm wavelength. Being circuit-switched, the fronthaul network has a fixed delay and a negligible jitter [12]. On the other hand, the cloud transport latency is less deterministic as it involves a mix of hardware, software and virtualized interfaces. To see the impact of the cloud network, we measure the one-way latency (measured from the round-trip time) between an external host and cloud resource. The host and the cloud resource are connected over 1/10 GbE Ethernet through a switch. We obtain the measurements by sending 1000 packets per second (LTE processes 1000 subframes per second) between the host and the cloud resource.

In Fig. 5.8 we show the distribution of one-way cloud latency for GPP and KVM platform with 1 and 10Gbps Ethernet network. The KVM uses a para-virtualized network driver, `virtio`, which is known to offer close to physical performance. However, the similarity between KVM and GPP driver is only seen in 99.9% of the packets. For instance, consider the performance with 1GbE connection. The mean latency for GPP is 0.16ms while it is 0.24ms for KVM. For around 1 in 10^4 packets, the latency with GPP increases to 0.25ms, while the latency for the KVM increases to 0.7ms — which is more than twice the GPP latency. Similar behavior is also observed in the 10GbE network. These observations imply that using the mean statistic of the trans-

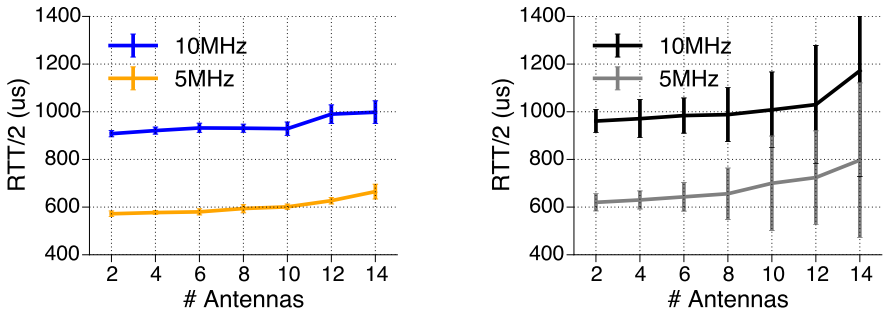


Figure 5.9: One-way transport latency for GPP (left) and KVM (right) vs number of antennas for 10GbE port.

port latency is not enough to provide latency guarantees. A KVM network driver may cause a fraction of the subframes to miss their processing deadline even if its mean transport latency has been accounted in the processing budget.

To emulate C-RAN's transport network, we build a medium-size testbed of 16 WARPv3 radios that are connected using Ethernet to an off-the-shelf GPP. The radios are connected via 1 GbE port, and then aggregated using a 1/10 GbE switch into GPP's 10GbE port. We use the CWARP transport library [104] to implement the read and write operations. Fig. 5.9 shows the one-way transport latency as we vary the number of antennas/radios and the sampling bandwidth. In the GPP scenario: for the 5 MHz case, we observe that the maximum latency is 620μs while it is exceeds 1000μs (or 1ms) for 10MHz bandwidth. Since LTE frames arrive every 1ms, to prevent queuing delay, at most 8 antennas at 10 MHz can be supported on the GPP. On the other hand, for the same bandwidth, we can only support 2 antennas on the KVM. Also, note the high transport jitter on the KVM, which is attributed to high latency of virtualization.

5.2.4 Deadline-miss

Once an uplink subframe is received at the radio frontends, an ACK or NACK response must be included in the downlink subframe that is transmitted exactly 3ms later. However, as illustrated in Fig. 5.10, not all of 3ms is available for Rx processing.

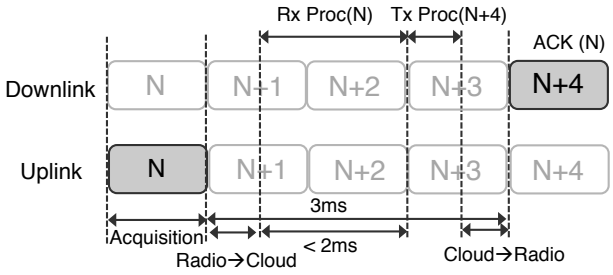


Figure 5.10: Example sequence of subframes in LTE showing the Rx and Tx processing timelines.

For example, subframe N received by the basestation (after acquisition) in the uplink needs to acknowledged by subframe $N + 4$ in the downlink. The Tx processing that encodes the response subframe cannot wait indefinitely for the Rx to finish. We assume the Tx processing starts 1ms before the actual over-the-air transmission of downlink frame [3]. As a result, only 2ms is effectively available for Rx processing and this includes the transport delay from radios to the cloud.

Formally, we can express the end-to-end timing requirements as:

$$T_{rxproc} + \overbrace{T_{fronthaul} + T_{cloud}}^{\text{RTT}/2} \leq 2ms \tag{5.2}$$

where $T_{fronthaul}$ is the fixed fronthaul latency and T_{cloud} is the cloud network latency. For ease of notation, the combined transport latency is denoted by RTT/2. One can rewrite Eq. (5.2) using the model given in Eq. (5.1). Furthermore, we can use the resulting equation to calculate the probability of a deadline-miss event. Assuming the fronhtaul latency is fixed, the calculation requires the underlying distribution of the network latency and platform error. Since the true distribution is difficult to model, we can use the empirical distribution obtained from a separate network and stress tests.

[3]Consistent with OpenAirInterface[105] implementation

5.3 C-RAN Scheduling

In what follows, we discuss the different approaches to scheduling subframes in C-RAN. We also present the design and implementation of RT-OPEX, a novel scheduling approach that reduces deadline-miss rate in wireless frame processing.

A typical C-RAN consists of two main components: transport and processing. The transport component makes an LTE subframe available every $1ms$ for processing. The processing component, on the other hand, receives these subframes and attempts to decode each of them within the available processing time budget. Specifically, the processing time of each subframe, T_{rxproc}, should be less than L_{max}, such that

$$T_{rxproc} \leq L_{max} = 2ms - (RTT/2). \tag{5.3}$$

One could view a typical C-RAN on a multiprocessor host as executing a set of processing threads, with each thread continuously running on a single core. An underlying scheduler assigns and schedules tasks for each processing thread, where each task represents a subframe decoding process.

5.3.1 Original Scheduling Approaches

There are two types of schedulers: *partitioned* and *global*. Below, we describe the design and implementation of partitioned and global schedulers.

5.3.1.1 Partitioned Scheduler

Generally, partitioned schedules are determined offline; each incoming subframe is assigned to a core based on a predetermined schedule. Such schedules have the advantages of providing statistical real-time processing guarantees and generating deterministic schedules.

Our focus here is not to generate an optimal partitioned schedule (minimizes the

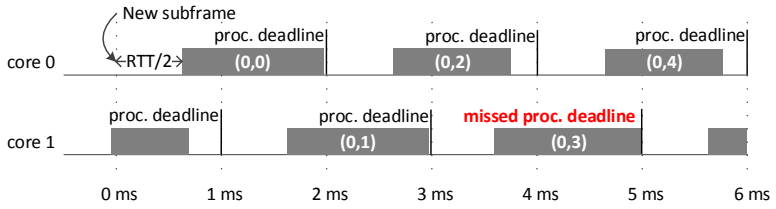


Figure 5.11: An example of a partitioned schedule on two cores. Notation (i,j) refer to processing of j^{th} subframe on the i^{th} basestation.

number of cores), but one that is feasible (meets processing deadlines). As is common in partitioned scheduling, we utilize an upper bound on execution time as an input (such that $L_{max} < 2ms$). The scheduler assigns $\lceil L_{max} \rceil$ cores for each basestation. For a basestation of id bs_{id} ($0 \leq bs_{id} \leq M-1$), it maps the subframe of index i to core $bs_{id} * \lceil L_{max} \rceil + i \, mod \, \lceil L_{max} \rceil$.

As a basestation receives a new subframe each 1 ms, the partitioned scheduler schedules two subframes on the same core each $\lceil L_{max} \rceil$ ms. This guarantees that each subframe has $\lceil L_{max} \rceil$ of available processing time on the core it is assigned to, which is larger than its upper bound of L_{max}. It can be easily shown that when $L_{max} = \lceil L_{max} \rceil$, this partitioned scheduler is optimal. Fig. 5.11 shows an example partitioned schedule on a 2-core host. In this example, the $\lceil L_{max} \rceil = 2$, so the partitioned scheduler assigns the subframes in a round robin fashion on the two cores each 1ms.

5.3.1.2 Partitioned Parallelized Scheduler

A partitioned scheduler can exploit parallelism to improve its performance and reduce deadline misses. If we consider the task model of Fig. 5.7, several stages of the processing chain can be parallelized. A scheduler can split each processing stage into subtasks, and then run these subtasks simultaneously over a set of cores. The main input to such scheduler is the degree of parallelism that indicates the performance gains.

Nevertheless, constructing an optimal parallelized partitioned schedule (one that minimizes the number of needed cores) is NP-complete with a much larger search space to cover than the original partitioned schedule [90]. As we are more interested in characterizing the performance gains of parallelizing a partitioned schedule, we resort to increasing the number of cores. In particular, our parallelized variant takes the original partitioned scheduler and the degree of parallelism (d_p) as inputs. It then dedicates $d_p - 1$ shadow cores for each processing core so that the total number of utilized cores is $d_p.\lceil L_{max} \rceil$. Shadow cores are dedicated for processing the parallelizable subtasks.

The scheduler works as follows: it starts processing each subframe according to the original partitioned schedule until it reaches a parallelizable stage (of P subtasks). It then runs on each of the processing and shadow cores P/d_p subtasks. Further, it uses barrier logic on the main processing cores to ensure that all the parallelizable subtasks have completed execution before moving to the next stage.

5.3.1.3 Global Scheduler

We utilize a single queue shared across basestations to implement the global scheduling. The queue is realized with a fixed-size ring-buffer that holds the incoming subframes from the basestations. The main scheduling thread runs on an independent core, performs book-keeping of the deadlines, and dispatches subframes from the queue to the available cores for decoding according to earliest-deadline-first (EDF) scheduling. Note that EDF is equivalent to FIFO scheduling when all basestations have the same transport delay.

Each core has a resident processing thread that receives the subframe dispatched from the main scheduling thread. A core will process at most one subframe at a time. If the processing does not end before the deadline, the processing thread terminates the ongoing task and goes to an idling state. It then waits for the next dispatched

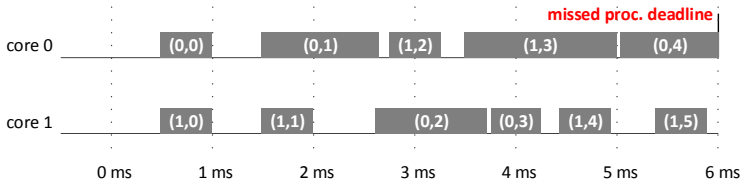


Figure 5.12: An example of a global schedule of two basestations on two cores.

subframe. Fig. 5.12 shows an example global schedule of two basestations on a 2-core host. In this example, when the processing finishes before the next subframe arrives, no queuing takes place, as seen at $t = 1ms$. On the other hand, at $t = 4ms$, the 4^{th} subframe of basestation 0, is queued and only dispatched at $t = 5ms$, so that it misses its processing deadline (at $6ms$).

Below, we present RT-OPEX, which builds on top of a partitioned scheduler and utilizes idle processor cycles to reduce deadline-misses, and consequently improve performance of a C-RAN.

5.3.2 RT-OPEX

Depending on the partitioned scheduler design, there will be time intervals during which the processing thread might not be actively processing a subframe. This occurs frequently as the processing time is dynamic and often not completely deterministic, especially due to turbo decoding. The processing time depends on external factors that can be completely unanticipated, such as the channel condition and network traffic load. Even if an optimal scheduler achieves tight packing of subframes, the dynamic processing times might still open up gaps (core being idle) or result in missing deadlines (longer than expected processing time).

For example, as shown in Fig. 5.11, the processing time of a frame, T_{rxproc}, might take time less than $\lceil L_{max} \rceil$, so that the core will be idle for the amount of time equal to $\lceil L_{max} \rceil - T_{rxproc}$. On the other hand, in extreme cases, T_{rxproc} will be larger $\lceil L_{max} \rceil$. This will happen as $\lceil L_{max} \rceil$ is usually chosen as the a percentile of the processing times

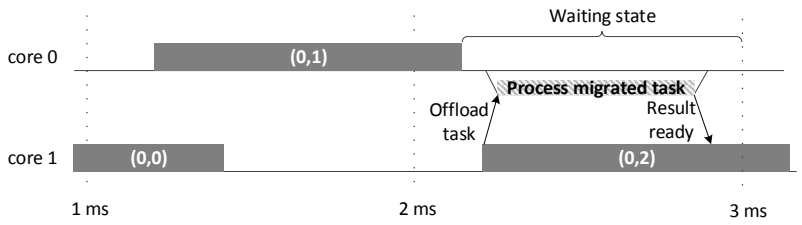


Figure 5.13: An example scenario of RT-OPEX showing migration between two cores.

distribution, but can't guarantee each processing time being less than $\lceil L_{max} \rceil$ [19]. This event will force a scheduler to drop the subframe to ensure the schedulability of incoming subframes, and consequently a deadline miss.

The processing thread alternates between two states: *active* and *waiting*. The active state corresponds to the case when it is actually processing a subframe (darkened portions of Fig. 5.11). The waiting state, on the other hand, corresponds to the case when it is not performing any active processing (empty portions of Fig. 5.11).

5.3.2.1 RT-OPEX Design

The design of RT-OPEX is inspired by an intuitive observation: if the processing thread of core 2 in Fig. 5.11 were able to utilize the idle cycles of core 1, then it wouldn't have missed its deadline.

RT-OPEX **op**portunistically **ex**ecutes a portion of a processing task on another idle core, which we refer to in the rest of this chapter as "migration". RT-OPEX is independent of any partitioned scheduler employed underneath. As long as multiple processing threads are running on different cores, there will be time intervals during which the active and waiting states of these threads will overlap. RT-OPEX exploits this phenomenon to decrease processing time, reduce the deadline-miss probability, and improve performance.

A. High-Level Description: From a high-level perspective, RT-OPEX *migrates* a subtask from a processing thread in its active state to another processing

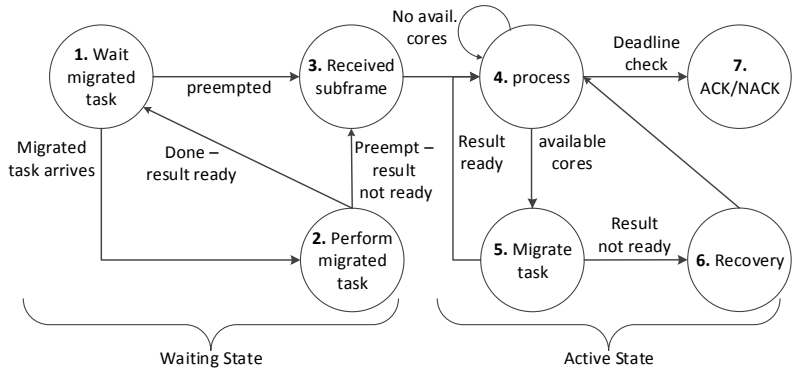


Figure 5.14: The state diagram of the processing thread in RT-OPEX.

thread (running on a different core) in the waiting state. We refer to the subframe processing task assigned to a processing thread by the scheduler as the *processing task*, and the subtask (part of the processing task) migrated by RT-OPEX to another core as the *migrated subtask*.

Fig. 5.13 shows an migration example. At $2.3ms$, RT-OPEX finds that the processing thread on core 1 is in its waiting state. So, it migrates a subtask from the processing task of core 2 to run on core 1. At that point of time, the processing task will be executing in parallel on both cores. After the migrated subtask finishes execution, it makes its result ready for the processing thread to consume it. The processing task then completes execution on core 2, and the processing thread on core 1 returns to its waiting state.

B. Migration Mechanism: Fig. 5.14 shows the state diagram of RT-OPEX. A processing thread alternates between two states, active and waiting. In its active state, it executes the processing task, and might execute migrated subtask(s) in its waiting state.

1. Waiting State: When the processing thread is in its waiting state, it waits for a migrated subtask for execution (from another processing thread – state 1). When such subtask arrives, the processing thread starts executing the migrated subtask immediately (state 2), where two of the following events might happen.

Algorithm 4 The Migration Algorithm in RT-OPEX.

Input: P subtasks, each subtask has tp proc. time.
Input: R cores, each core has $fc_j > 0$ of free time.
Input: δ, cost of migrating a subtask to another core.
$S \leftarrow P$ ▷ # of left subtasks (not migrated)
$max_{off} \leftarrow 0$ ▷ max # of migrated subtasks per core
while $S > 1$ **and** $j \leq R$ **do**
    $lim_{off} = \lfloor \frac{fc_j}{tp+\delta} \rfloor$ ▷ # of subtasks can be migrated
    $n_{off} \leftarrow min(S - max_{off}, lim_{off}, \lfloor \frac{S}{2} \rfloor)$
    $max_{off} \leftarrow max(n_{off}, max_{off})$
    Migrate n_{off} subtasks to j^{th} core
    $S \leftarrow S - n_{off}$
    $j \leftarrow j + 1$
end while

1. The migrated task completes before a new processing task is available, i.e., before it gets preempted. RT-OPEX sets a *result ready* flag for the "remote" processing thread (the one migrating a subtask) to consume the result. The thread then returns to waiting for a migrated subtask (state 2 → state 1).

2. The migrated subtask is preempted at the deadline before it is completed. This indicates that a new processing task is available for the processing thread. RT-OPEX sets a *result not ready* flag and switches the processing thread to the active state (state 2 → state 3).

While waiting for a migrated subtask (at state 1), the transport component can preempt the processing thread to indicate that a new processing task is available (state 1 → state 3).

2. Active State: When the processing thread receives a new processing task, it switches to the active state, and starts processing the subframe (state 4).

The processing thread starts processing the subframe until it reaches a parallelizable task which offers an opportunity for migration to idle cores (such as FFT or decoder). As the arrival of subframes is deterministic, the underlying scheduler should be able to inform when each idle core will be preempted and switched to active

processing. As such, the scheduler can compute the potential available time budget for migration on each idle core.

RT-OPEX uses this knowledge along with the model of the subtask execution time to decide how many subtasks to migrate to each core (state 4 $\rightarrow$ state 5) through applying Alg. 4. RT-OPEX follows a greedy approach; it tries to migrate subtasks as much as possible with one caveat. The number of subtasks left to be executed locally (i.e., those that are not migrated) should be larger than the maximum number of migrated subtasks to each idle core. In particular, the number of migrated subtasks to a core j, n_{off}, should satisfy the following requirements.

R1. It must be less than the maximum number of subtasks core j can accommodate (given in line 7 as lim_{off}), such that $n_{off} \leq lim_{off}$. lim_{off} includes the subtask execution time as well as the subtask migration cost, δ.

R2. The number of subtasks left after migrating $S - n_{off}$ should be larger than the maximum number of subtasks already allocated to any other core such that $S - n_{off} \geq max_{off}$. It follows that $n_{off} \leq S - max_{off}$.

R3. The number of un-migrated subtasks, $S - n_{off}$, should be larger than the number of subtasks migrated to core j. We need this condition as the previous step does not count in the subtasks to be migrated to core j. We then have $S - n_{off} \geq n_{off}$ so that $n_{off} \leq S/2$.

At line 8, Alg. 4 combines the three requirements so that $n_{off} = min(S - max_{off}, lim_{off}, \lfloor \frac{S}{2} \rfloor)$. After calculating n_{off}, RT-OPEX migrates n_{off} subtasks to core j. It repeats the same process until either the number of subtasks for migration or the number of cores is exhausted.

Due to Alg. 4, the processing thread does not have to wait for any migrated subtask for completion. By the time the processing thread finished the local subtasks, all migrated subtasks must have completed in the ideal case. The processing thread

can use their results and move on with the execution (state 5 → state 4). On the other hand, if the local processing is complete and at least one migrated subtask is not completed, the processing thread goes to recovery state (state 5 → state 6). The recovery state simply involves computing the results for the incomplete migrated subtasks. This ensures that the performance of RT-OPEX will be no worse than the baseline case. In the baseline case, all subtasks are executed serially which corresponds to the worst-case scenario of RT-OPEX (no migrated subtask completed execution).

Once all subtasks corresponding to a single processing task are completed, the processing thread continues executing the rest of the decoding tasks. It repeats the same procedure for any task that can be parallelized. RT-OPEX always monitors whether the processing thread violated the task's processing deadline ($T_{rxproc} \leq L_{max}$). Depending on the deadline check status, the processing might result in either an "ACK" or "NACK" message to the radio (state 7) after execution has completed. RT-OPEX then switches the processing thread back to the waiting state.

5.4 Implementation and Evaluation

We implement RT-OPEX and rest of the schedulers on a testbed of software radios, Ethernet, and commodity server hardware. In this section, we give the details of the implementation, the evaluation platform, and performance evaluation of the schedulers.

5.4.1 Implementation

We build a C-RAN multiprocessor scheduler from the ground up. Our scheduler utilizes the `pthread` library to implement the processing and transport components. Each processing core has a dedicated thread pinned to it, and similarly, each antenna or radio has a dedicated transport thread. The transport threads run on a dedicated set of cores that are separate from processing cores. The transport and the processing

threads are synchronized using semaphores. As the transport threads are critical to maintain synchronization between the radios and the GPP (triggered every $1ms$), we use a *one-way locking* mechanism where only processing threads wait for the transport threads. The processing threads are signaled when the transport threads finish writing to the sample IQ buffer. We implement a common watchdog timer that maintains a global reference time using which the deadline misses are detected.

Our scheduler integrates with the OpenAirInterface (OAI) PHY library [105]. OAI is an open-source software implementation of LTE that includes both RAN and Evolved Packet Core, and implements all the PHY-layer functions of LTE Rel 10. OAI implements its own out-of-the-box partitioned scheduler for uplink and downlink processing, but it is not amenable to PHY-layer migration. Therefore, we modularize the OAI processing and write an abstraction layer that abstracts the PHY functions at task level, `taskX`, and subtask level, `subtaskX`, where `X` $\in$ {FFT, demod, decode}. Within each task, the functions that are not currently parallelized are treated as left unchanged. Each of the subtasks can be executed independently, and thus, provides the basis for parallelism. Since OAI implements a complete baseband chain, our abstraction code is tested using traces from OAI simulators to make sure that the processing is reproducible. This step was essential to ensure the correct functioning of the scheduler when the OAI data-structures are duplicated for multiple basestations.

Using our system design, partitioned scheduling is realized by fixing the threads on which basestation's subframes are processed, i.e., when a particular subframe is received from the transport, only the corresponding processing thread is notified. In case of global scheduling, there is no binding of a basestation to threads, and any idle processing thread can process an arriving subframe. In RT-OPEX, the migration of subtasks implies that the some of the `subtaskX` routines from the current processing thread are migrated to a different processing thread. As the global OAI variables and the baseband samples are held in a shared memory (L3 or main), migration of data

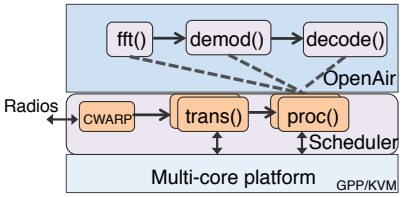


Figure 5.15: Implementation framework for SchedTool.

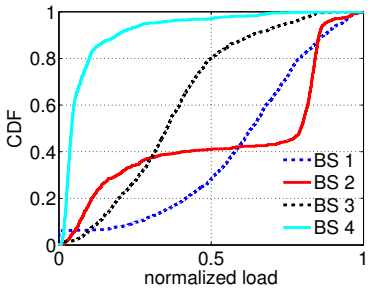


Figure 5.16: Basestation load distribution.

is realized by passing the references to the memory contents.

Enforcing deadlines. For correct operation of a real-time system, we must ensure that a frame processing task must be completed before, or terminated at its deadline. Implementing this in our system is challenging since we abstract away the low-level PHY routines. One possible solution is to pass a timer to each OAI function, and constantly check on the timer. This approach, however, is not practical. Instead, we check on the slack time before we perform each task; using our task model we check if the execution time is less than the slack time, else we drop the task and the subframe. The resulting gaps are however not used for migration.

Our code is packaged into an open-source tool, *SchedTool*, that can be used to validate the performance of different schedulers (Fig. 5.15). *SchedTool* is highly configurable as the user can choose any of the possible scheduling algorithms, number of basestations, number of processing cores, number of antennas, and the transport bandwidth. Moreover, the user can run it for different underlying operating system configurations, including thread scheduling models (e.g., Round-Robin and FIFO), real-time kernel, virtualization, etc. Ultimately, the tool can be used to profile the system performance (deadline-miss rate, load, memory usage) which can, in turn, help operators design and provision compute resources for C-RAN.

5.4.2 Evaluation Platform

We use different state-of-the-art computing and networking platforms in our evaluations. For computing, we use a commodity server (i.e., a general-purpose processor (GPP)) and a virtual machine (VM) that is hosted on a GPP. The GPP is a 32-core (hyper-threading enabled) machine with Intel Xeon E5-2660 2.2 GHz x86 CPUs (SandyBridge architecture), 128GB RAM, 15MB L3 cache, and 1/10 GbE Ethernet ports. For virtualization, we use a KVM hypervisor that emulates the underlying hardware to host a guest operating system. The evaluation with other means of virtualization such as containers (e.g., LXC) is left to future work. To closely match the performance of data-center networks, we consider 1 GbE and 10 GbE Ethernet links that are connected to the GPP/VM through an HP 6600 series Ethernet switch. The GPP uses standard Intel network drivers while the VM uses a *virtio* network driver.

Optimizations. Several optimizations are applied to get the best computing performance. The OAI workload runs on an Ubuntu 14.04 low-latency kernel. Considered as a soft real-time system, the low-latency version is a stable kernel compared to other hard real-time kernels like RTLinux [106] which require custom patches. Various optimization features, such as SSE3/SSE4 instruction set, O3 flags, etc., are enabled. Further, the power-saving features and sleep states available on Intel processors are disabled. This ensures the CPU cores run at a constant maximum frequency. To minimize disruptions from interrupts, the processing threads are pinned to dedicated cores and use FIFO scheduling. We use the OAI timestamps to calculate the processing time. The timestamps are obtained by counting the exact number of CPU clock cycles.

Data collection. Since publicly available basestation traces are difficult to obtain, we devise a measurement setup to get the variations of cellular traffic. We use USRP software radios and log RF samples off the air on Band-13 and Band-17 LTE downlink channels in a city environment. Specifically, we log the signal of 4 cellular towers and

estimate the load by correlating with the average signal energy every $1ms$. Fig. 5.16 shows the distribution of the load variations for the 4 basestations. We then run the scheduler with IQ traces from OAI, where the MCS of each subframe is determined by our basestation load trace. For each experimental setting, we use an AWGN channel model with a fixed SNR of 30dB and collect the processing logs of 30000 LTE subframes from each basestation. Fixing the SNR makes it easier to compare the scheduler performance.

Experimental setup. We consider a 4-bsestation setup, each with two antennas ($N = 2$), running on a GPP platform. We specify the OAI LTE bandwidth to 10MHz, which corresponds to a sampling rate of 15.36MHz, i.e., each subframe contains 15360 samples. We consider a single user uplink transmission and assume 100% PRB utilization. Under this configuration, the nominal PHY throughput can vary from 1.3 to 31.7Mbps, depending on the MCS used. Further, we choose $\lceil L_{max} \rceil$ to 2, i.e., each basestation is assigned 2 CPU cores under partitioned scheduling.

We first run the processing with our C-RAN testbed with radios and the Ethernet transport. As mentioned earlier, the radios in our testbed are WARPv3 SDR boards. From Fig. 5.9(a), observe that the one-way latency from radios to the GPP at 10MHz bandwidth is as high as $0.9ms$. This effectively leaves $1.1ms$ to process each subframe (which is much less than the processing time of $1.5ms$), resulting in a very high deadline-miss rate. Therefore, to accurately emulate real C-RAN deployments, we replace the WARP transport with a fixed transport delay (RTT/2) value ranging from $0.4ms$ to $0.7ms$, that represents various off- and on-site deployment scenarios. In what follows, we describe the results for a single GPP platform, but note that similar results were also observed on our KVM platform.

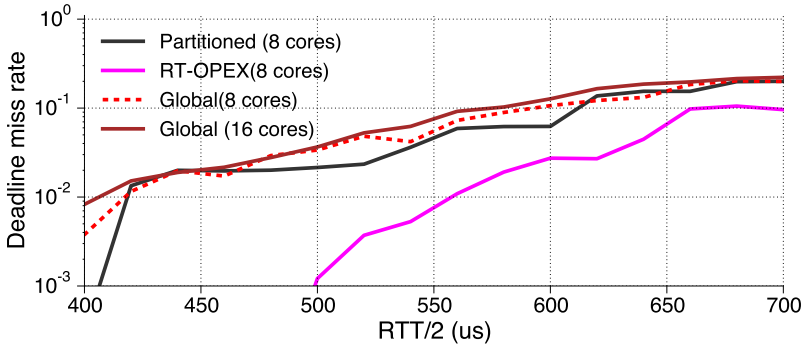


Figure 5.17: Deadline-miss comparison of schedulers.

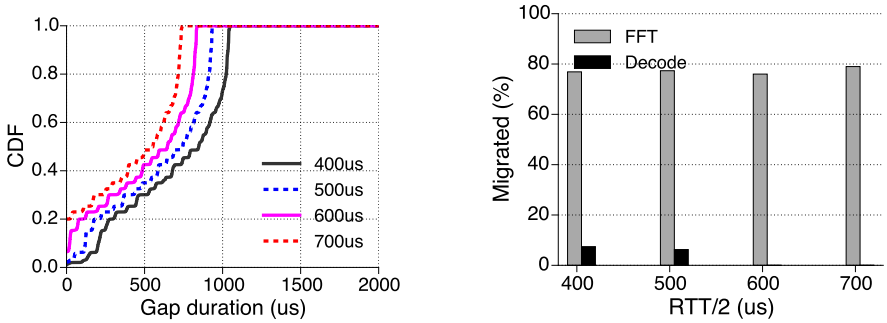


Figure 5.18: Role of RTT/2 in gaps and task-migrations.

5.4.3 Performance comparison

For each transport delay setting, we evaluate the deadline-miss rate for the four base stations and for each scheduler. Fig. 5.17 shows the deadline-miss performance of the different schedulers. The main takeaways from the figure are as follows.

RT-OPEX **Performance:** RT-OPEX exhibits virtually zero deadline-miss rate when latency is less than $500\mu s$. To understand this further, let's look at Fig. 5.17. For RTT/2 less than $500\mu s$, the partitioned scheduler has gaps (only due to processing time variation) larger than $500\mu s$ for 60% of the processed subframes. RT-OPEX utilizes these gaps to migrate FFT and decoding subtasks as evident from the Fig. 5.18(b), where 20% of the decode subtasks are migrated. These migrated decode subtasks belong to subframes with high MCS that are responsible for the deadline misses in the original partitioned scheduler. By migrating these tasks, the processing time drops well below $1500\mu s$, which is less than the processing budget.

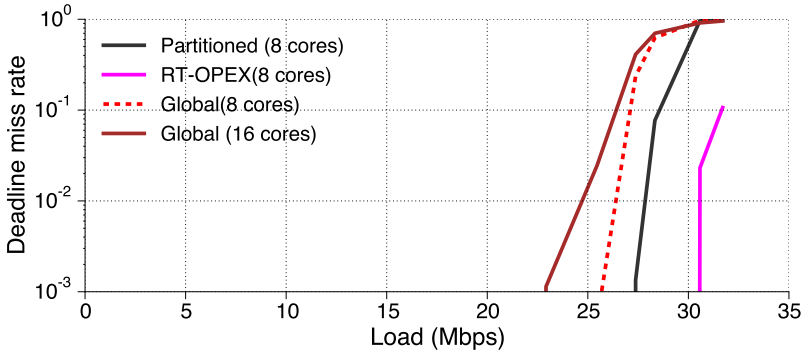


Figure 5.19: Deadline misses vs. load (RTT/2=500μs)

As latency increases beyond 500μs, the gaps get narrower, thus reducing the chances for migrating the decode subtasks. Nevertheless, RT-OPEX keeps on migrating the smaller size FFT subtasks, resulting in the deadline-miss rate significantly lower than that of the partitioned and global schedulers. As evident from Fig. 5.17, the deadline-miss rate of RT-OPEX is one order-of-magnitude better ($10^{-2} \rightarrow 10^{-3}$) than that of both partitioned and globals schedulers.

Partitioned Scheduler: Unlike RT-OPEX, a partitioned scheduler can't exploit gaps available because of the processing time variations. This is evident from the sudden rise of the deadline-miss rate when RTT/2 exceeds 400μs. The available time budget of processing falls below 1600μs. Referring to Fig. 5.4(a), the processing time can exceed 1.5ms for higher MCS values.

As a result, most subframes with MCS larger than 20 will miss their processing deadlines. As RTT further increases above 400μs, the deadline-miss rate increases, albeit at a slower rate. Subframes with lower MCS values can be successfully decoded within 1.3ms (corresponding to RTT/2 = 700μs).

Partitioned Parallel Scheduler: The partitioned parallel scheduler performs the best among the schedulers considered. Recalling that we evaluate this scheduler with a degree of parallelism equal to 2, the processing times are effectively cut by 40% (after counting in scheduling and parallelizing overheads). This indicates that the processing time for subframes with the highest MCS values will virtually never exceed

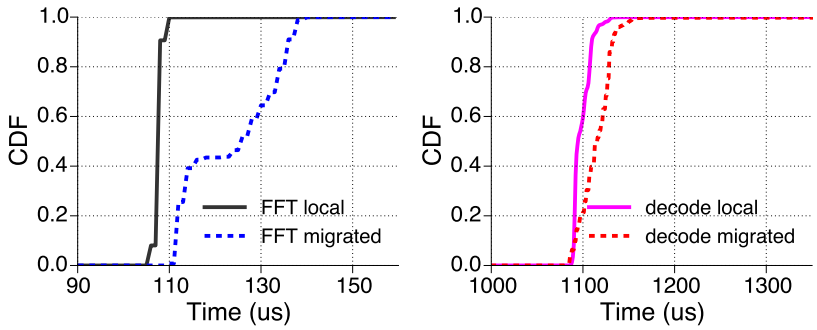


Figure 5.20: Comparison of processing times of local and migrated tasks.

$1.3ms$ — the available time budget at RTT/2 = $700\mu s$. Therefore, the partitioned parallel scheduler never misses any deadline, and hence it is not part of Fig. 5.17.

Global Scheduler: The global scheduler exhibits the most surprising behavior. We evaluate two global schedulers, one running with 8 cores while the other utilizes 16 cores. Theoretically, this scheduler should perform as good as a partitioned scheduler with $\lceil L_{max} \rceil = 2ms$. Both schedulers provide a subframe with all the time needed to finish decoding before its deadline.

Nevertheless, as evident from Fig. 5.17, the global scheduler (1) performs slightly worse than the partitioned scheduler, (2) does not improve when the number of cores is doubled from 8 to 16, and (3) does not exhibit a zero deadline-miss rate even at the lowest RTT value. As explained later, several factors related to the design and execution of the global scheduler contribute to this surprising phenomenon.

In Fig. 5.19, we set RTT/2 to 500μs and show the deadline-miss performance for different subframe loads (corresponding to different MCS values). RT-OPEX's gains are shown to be prominent at higher loads (30Mbps and above) where rest of the schedulers miss deadlines for 100% of the frames. Therefore, assuming a deadline-miss threshold of 10^{-2} that is typical of real-time systems, RT-OPEX can support 15% higher load (31Mbps compared to 27Mbps) than a default partitioned scheduler.

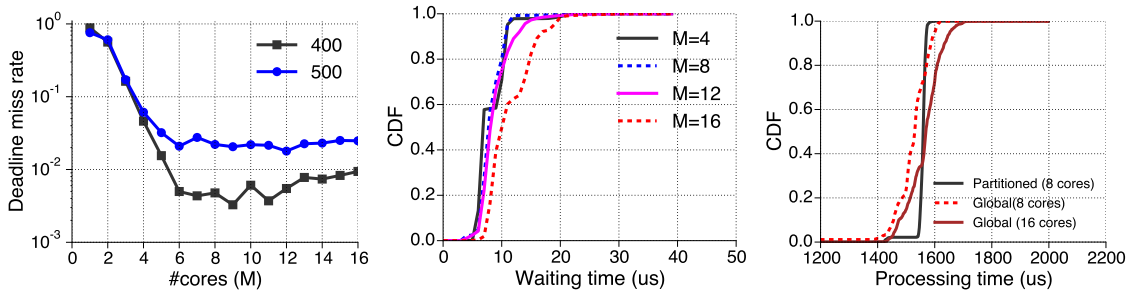


Figure 5.21: Global scheduler as cores are varied

5.4.4 Overheads

The major overhead in migration comes from the transfer of contents from shared memory. Given the complex memory hierarchy of modern processors, providing a detailed cache analysis is out of the scope of this chapter. However, we use our abstraction to calculate the overhead of task migration by simply measuring the time it takes to process a local and a migrated task. Fig. 5.20 compares the processing times of the tasks that are performed locally and that are migrated to and executed on a different core at runtime. Observe that there is always a non-zero overhead to migrate a task. For example, for FFT, the median processing time increases from 108μs to 126μs when it is migrated, i.e., a 18μs increase. For the decode task, the overhead is nearly the same at 20μs. Thus, the cost of migration is a fixed across the subtasks, which corresponds to the fetching of global OAI variables from shared memory to on-chip/local memory. Note that the subframe buffer and transport block buffer are both referred within the OAI eNB data-structure, and therefore, both FFT and decode migration (including subtask) involve the same amount of memory transfer.

Global scheduling, despite offering flexibility, suffers from performance loss and high overheads. More interestingly, increasing the number of cores does not mitigate this (and even decreases performance). To see this, consider Fig. 5.21, where deadline-miss performance saturates beyond 8 cores. There are two reasons for this:

Table 5.2: Qualitative Comparison of Scheduling Approaches in C-RAN.

Scheduling	**Determin-istic**	**Optimal**	**Overhead**	**Flexible to resources**	**Flexible to load**
Partitioned	✔	✔	*low*	–	–
Partitioned–P	✔	✔	*low*	–	–
Global	–	–	*high*	✔	✔
RT-OPEX	✔	✔	*med.*	–	✔

1) increased lock contention, and 2) cache trashing. In the same figure, the waiting time plot shows the distribution of waiting time for a single basestation in the global queue as we vary the number of cores. Theoretically, the waiting time must be 0 since each subframe is guaranteed to be given a core. However, in case of 16 cores, more than 20% subframes wait for more than 20μs in the queue. This is explained from increased lock contention. In addition, as each core in global scheduling switches between different basestations every few subframes, this leads to frequent flushing of its cache that adds to the processing times, as evident from the right-most plot in Fig. 5.21, which shows the processing time distribution for MCS 27. From the plot, we observe that global with 16 cores has a considerably larger processing time (80μs) for more than 10% of the subframes.

5.5 Discussion

Based on our evaluation of the different scheduling approaches, we now discuss how they fare under different system configurations and operator requirements. In Table 5.2, we show the comparison summary of the scheduling approaches.

A. Determinism: Determinism is a desirable feature in the design of hard real-time systems [107]. In comparison with fixed partitioned schedules, global schedules are non-deterministic; a subframe is assigned to the next available core. In contrast, both partitioned scheduler and RT-OPEX follow a deterministic execution

plan which eliminates waiting time, and thus reduce processing time when compared to the global scheduler.

B. Optimality: Given an upper bound on the processing time, an operator can obtain an optimal partitioned schedule (both regular and parallelized) that is feasible and minimizes the number of needed cores [19] . The general problem of obtaining an optimal partitioned schedule is, however, NP-complete [89, 90, 91]. The search space becomes intractably large when designing an optimal parallelized partitioned schedule because of the larger number of subtasks as well as the precedence constraints among the subtasks [108, 90]. However, RT-OPEX is as optimal as a partitioned scheduler since it uses the same number of cores.

C. Overhead: The additional scheduling overhead of a partitioned (and parallelized) schedule is minimal; each subframe is assigned to a predetermined core without the need of locking or task/data migration across cores. On the other hand, a global scheduler incurs higher overhead because of a shared queue as well as from constant cache trashing. RT-OPEX incurs the overhead of subtask migration, which we estimated to be in the order of $20\mu s$ for both FFT and decode subtasks. RT-OPEX takes this overhead into account while migrating so as to guarantee feasible migration. Even with the overhead, we show in §5.4.3 that RT-OPEX achieves a significant improvement in deadline-miss performance.

D. Flexibility to resources: Available resources in C-RAN might change over time as storage, memory, and processor failures are common in a datacenter running on commodity hardware [109]. As a partitioned schedule is provisioned to a set of fixed resources, any change in the available resources results in a significant performance degradation [110]. Alternatively, a global schedule, by virtue of its design, adapts to the underlying resources without the need to design a new schedule (Fig. 5.21). RT-OPEX, suffers from the same limitation of a partitioned schedule, but can automatically exploit any added resources to migrate subtasks.

E. Flexibility to load: From our measurements, we observe that processing times exhibit millisecond-level variations due to varying traffic loads and channel conditions (Fig. 5.1). However, the partitioned schedule fails to adapt to varying processing times. When the processing time of a subframe exceeds the deadline, partitioned schedules drop the subframe resulting in a deadline miss. This occurs even though processing on another resource might introduce a gap. RT-OPEX fills the scheduling gaps by migrating subtasks to the available cores. It, therefore, adapts to the variations in the load. By design, the global scheduler is inherently flexible to the varying processing time.

5.6 Related Work

Real-time wireless frame processing in software has been an active area of research over the past decade [72, 73]. Today, there are commercial software implementations of a fully functional LTE basestation such as Amarisoft [111]. However, their performance is nowhere close to the hardware performance. With the introduction of C-RAN [5], a slightly different variant — cloud-based processing — has received considerable attention.

Much of the research in C-RAN has focused on the system architecture and implementation. The authors of [19] provide a framework to schedule multiple basestations on a multi-core platform in order to meet their processing requirements. However, they assume the processing time for the subframe is fixed and do not consider the effect of deadline-misses. In [88], a more flexible approach to resource management was proposed, where a pool of shared cores are made available based on the dynamics of the load. However, these scheduling decisions are made before wireless frames are received, and thus cannot account for channel variations. The role of virtualization in RAN was described in [73, 112, 113]. A container approach to virtualization was shown to have a slightly better performance than a hypervisor approach. In [114],

the authors adopt a two-tier model to manage a RAN where part of the control that requires less frequent changes goes to a central controller. The design of a remote radio head and its synchronization with the processing units was described in [87]. Finally, [12] and the references therein provide a comprehensive background on the state of the current C-RAN technology.

5.7 Conclusion

C-RAN is a promising solution to the problem of economically managing the scale and size of wireless processing. However, meeting frame processing deadlines without over-provisioning resources remains a major challenge. We proposed to meet this challenge with a new scheduling framework that builds on top of partitioned scheduling and opportunistically exploits the idle processing cycles for parallel processing of frames. Our evaluation results have demonstrated its potential in reducing deadline-misses at no additional cost. In essence, our approach is resource pooling applied at finer timescales.

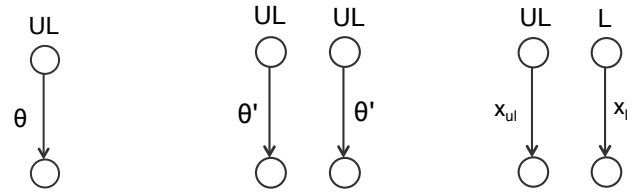
CHAPTER VI

Conclusion and Future Directions

6.1 Concluding Remarks

Future deployments of wireless networks face many challenges from the exponential growth of wireless devices. Their capacity must handle traffic volumes that will be orders of magnitude larger than today's networks. Also, they should provide wide-ranging connectivity, improved quality-of-service, and must achieve this with increased operational efficiency using architectures that are cost-effective, flexible and upgradeable. All of these requirements are part of a bigger quest towards a scalable design, which is a wireless network architecture that can efficiently scale using the resources of spectrum, transmitters, and processors.

In this thesis, we present the building blocks of a scalable wireless network, namely, co-existence between heterogeneous technologies such as LTE and WiFi; co-ordination of multi-antenna transmitters and their feedback design; and transport and processing architectures for efficient centralized processing. Each of these blocks addresses the shortcomings of existing approaches and proposes a solution that outperforms its counterparts in the evaluated scenarios. First, the duty-cycling approach to co-existence was shown to perform poorly in real-world scenarios as it does not account for the transmission gaps or the access fairness. To address this, we propose a CSMA-based access mechanism, which eliminates collisions, and is provably fair. Second, in



(a) Single unlicensed (b) Two unlicensed (c) Unlicensed and licensed

Figure 6.1: Unlicensed and licensed links in the unlicensed spectrum.

a coordinated Network MIMO system, we show that channel state information gets corrupted due to channel mobility, which is mitigated to an extent through our adaptive transmission schemes. Third, we introduce a tree-based design for real-time transport of baseband samples and provide its end-to-end schedulability and capacity analysis. Finally, we present a new framework for processing of baseband samples, that improves deadline-miss performance, and alternatively, reduces the number of networking and computing resources.

6.2 Future Directions

The presented approaches in the thesis open up several avenues for further exploration, least of which are as follows.

6.2.1 A Closer Look at Fairness in Unlicensed Spectrum

In Chapter II, we described our fairness metric and showed through modeling and evaluations that TALOS satisfies our fairness criteria. In related work, [47] proposed a mechanism to achieve proportional fairness between LTE and WiFi. However, these are not the only ways to define fairness in the unlicensed spectrum. Here we look at other possible mechanisms and explore if they can be enforced in practice. Ultimately, we wish to study these mechanisms in actual deployments and report the findings.

Consider the usage of unlicensed spectrum by unlicensed (UL) users, and licensed (L) users who use it opportunistically. We assume the unlicensed users are the pri-

mary occupants of the unlicensed spectrum and do not recognize the transmissions of licensed users. As shown in Fig. 6.1, we consider three scenarios of links transmitting in the unlicensed spectrum: a) A single UL link with throughput θ; b) Two UL links, each with equal throughput θ', assuming fair access between unlicensed devices; and c) A single link L with throughput x_L co-existing with UL link having throughput x_{UL}. We now define the fairness criteria under different models of fairness.

True fairness. We call the co-existence is *truly* fair if link L does not benefit at the cost of UL and the link L has the same throughput as a shared UL links that is, when the following is true:

$$x_{UL} = x_L = \theta'. \tag{6.1}$$

Observe that this is the strictest sense of fairness in the sense that a link L is a drop-in replacement of UL link with the same performance and behavior.

CRN fairness. We borrow the co-existence model of cognitive radio networks (CRNs) [6] and define CRN fairness to hold when the UL link sees at most η, $0 < \eta < 1$, fraction loss in performance, i.e.,

$$x_{UL} \geq \theta(1 - \eta). \tag{6.2}$$

CRN fairness is observed when link L performs continuous sensing and only transmits when the medium is free. Upon a collision with UL transmission, it immediately backs off and waits for a sufficient time for the medium to go idle. The mechanism is similar to the manner secondary users (SU) use the radio bands of incumbent primary users (PUs) in CRNs. As expected, this fairness is biased towards unlicensed links.

TALOS fairness This fairness metric is based on our definition from Fig. 2.18, which states that a link L should not benefit at the cost of two UL links. Formally,

this implies that:

$$x_{UL} = \theta'. \tag{6.3}$$

In contrast to CRN fairness, TALOS fairness is biased towards licensed links. Also, note the difference between True and TALOS fairness, in TALOS, there are no restrictions on the throughput of the licensed link.

6.2.2 Virtualization of IoT Gateways

In this thesis, we consider the transport and processing of only LTE and WiFi frames, but our designs are extendable to any wireless deployment. For instance, our real-time implementation from Chapters IV and V can be applied to baseband processing in internet-of-things (IoT) gateways. Existing IoT gateways employ dedicated wireless interfaces to communicate with IoT devices through Bluetooth/BLE and Zigbee protocols. However, to maximize the compute efficiency, the baseband processing on the gateway can be decoupled from the hardware and carried out on a backend server platform. The only constraint in this implementation comes from the 150μs and 192μs inter-frame spacing (IFS) durations of Bluetooth and ZigBee, respectively. However, as we have shown, these deadlines can be met by taking a principled real-time approach towards transport and processing.

6.2.3 Heterogeneous Standards and Platforms

In Chapter V, we proposed the idea of combining parallelism with scheduling for baseband processing. The gains from this approach are even more significant in a C-RAN deployment of heterogeneous basestations. Newer wireless standards that are targeted towards IoT devices are expected to have characteristically different traffic patterns and deadlines than current cellular traffic. For example, most IoT

devices such as smart-grid meters, activity trackers, home sensors, etc. generate data periodically in short-lived sessions. Backend processing of the aggregated loads of IoT and mobile devices will result in frequent unused CPU cycles. We can apply RT-OPEX to leverage the idle cycles to improve the deadline performance of processing.

As seen from Eq. (5.1), the processing time is dominated by the Turbo decoder. Alternatively, the decoding can be implemented on dedicated baseband accelerators (or DSPs) that are an order-of-magnitude faster. GPUs with their extensive multi-threading are another possibility [96]. The primitives of task migration in these scenarios need further investigation as the migration overhead is architecture specific.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] The Internet of Things. http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf. [Online; accessed 29-Nov-2015].

[2] Cisco Visual Networking Index: Forecast and Methodology, 2014-2019.

[3] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications . IEEE Std. 80211ac Draft 3.0, 2012.

[4] B. Soret, K. I. Pedersen, N. T. K. Jrgensen, and V. Fernndez-Lpez. Interference Coordination for Dense Wireless Networks. *IEEE Communications Magazine*, 53(1):102–109, January 2015.

[5] C-RAN: The road towards green RAN. http://labs.chinamobile.com/cran/wp-content/uploads/2014/06/20140613-C-RAN-WP-3.0.pdf. [Online; accessed 29-Nov-2015].

[6] C. Cordeiro, K. Challapali, D. Birru, and Sai Shankar. IEEE 802.22: The First Worldwide Wireless Standard Based on Cognitive Radios. In *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks DySPAN*, 2005.

[7] LTE in Unlicensed Spectrum: Harmonious Coexistence with Wi-Fi. *Qualcomm Research*, June 2014.

[8] G. Boudreau, J. Panicker, N. Guo, R. Chang, N. Wang, and S. Vrzic. Interference Coordination and Cancellation for 4G Networks. *IEEE Communications Magazine*, 47(4):74–81, 2009.

[9] F. Rusek, D. Persson, Buon Kiong Lau, E.G. Larsson, T.L. Marzetta, O. Edfors, and F. Tufvesson. Scaling Up MIMO: Opportunities and Challenges with Very Large Arrays. *IEEE Signal Processing Magazine*, 30(1):40–60, Jan 2013.

[10] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong. Argos: practical many-antenna base stations. In *Proc. of MOBICOM*, 2012.

[11] C. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne. Real-time communication in packet-switched networks. *Proceedings of the IEEE*, 82(1):122–139, 1994.

[12] A. Checko, H.L. Christiansen, Ying Yan, L. Scolari, G. Kardaras, M.S. Berger, and L. Dittmann. Cloud ran for mobile networks – a technology overview. *Communications Surveys Tutorials, IEEE*, 17(1):405–426, 2015.

[13] Nokia LTE for unlicensed spectrum. *Nokia Whitepaper*, 2014.

[14] U-LTE: Unlicensed Spectrum Utilization of LTE. *Huawei Whitepaper*, 2014.

[15] David Tse and Pramod Viswanath. *Fundamentals of wireless communication.*

[16] H. Balan, R. Rogalin, A. Michaloliakos, K. Psounis, and G. Caire. Achieving High Data Rates in a Distributed MIMO system. In *Proc. of MOBICOM*, 2012.

[17] H. Rahul, S. Kumar, and D. Katabi. JMB: scaling wireless capacity with user demands. In *Proc. of SIGCOMM*, 2012.

[18] Qing Yang, Xiaoxiao Li, Hongyi Yao, Ji Fang, Kun Tan, Wenjun Hu, Jiansong Zhang, and Yongguang Zhang. Bigstation: Enabling scalable real-time signal processingin large mu-mimo systems. In *Proc. of SIGCOMM*, 2013.

[19] Sourjya Bhaumik, Shoban Preeth Chandrabose, Manjunath Kashyap Jataprolu, Gautam Kumar, Anand Muralidhar, Paul Polakos, Vikram Srinivasan, and Thomas Woo. Cloudiq: A framework for processing base stations in a data center. In *Proc. of Mobicom*, 2012.

[20] Chi-Kin Chau, Minghua Chen, and Soung Chang Liew. Capacity of Large-scale CSMA Wireless Networks. In *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, MobiCom '09, pages 97–108, New York, NY, USA, 2009. ACM.

[21] D. Gesbert, S. Hanly, H. Huang, S. Shamai Shitz, O. Simeone, and Wei Yu. Multi-Cell MIMO Cooperative Networks: A New Look at Interference. *in IEEE Journal on Selected Areas in Communications*, 2010.

[22] Christopher Cox. *An Introduction to LTE: LTE, LTE-Advanced, SAE, VoLTE and 4G Mobile Communications.* John Wiley & Sons, 2nd edition, 2014.

[23] LTE-U Technology and Coexistence. *Qualcomm Technologies, Inc*, May 2015.

[24] Broadband Radio Access Networks (BRAN); 5 GHz high performance RLAN; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive. *ETSI EN 301 893 V1.8.1*, March 2015.

[25] IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2007*, pages 1–1076, June 2007.

[26] LTE-U/LAA, MuLTEfire and WiFi; Making Best Use of Unlicensed Spectrum. *Qualcomm*, 2015.

[27] WARP 802.11 Reference Design. `http://warpproject.org/trac/wiki/802.11`. [Online; accessed 23-Sept-2015].

[28] George Nychis, Thibaud Hottelier, Zhuocheng Yang, Srinivasan Seshan, and Peter Steenkiste. Enabling MAC Protocol Implementations on Software-defined Radios. In *Proc. of NSDI*, 2009.

[29] S. Sen, R.R. Choudhury, and S. Nelakuditi. CSMA/CN: Carrier Sense Multiple Access With Collision Notification. *IEEE/ACM Transactions on Networking*, 20(2):544–556, April 2012.

[30] *Handbook on spectrum monitoring.* International Telecommunication Union, 2002.

[31] Eugenio Magistretti, Krishna Kant Chintalapudi, Bozidar Radunovic, and Ramachandran Ramjee. WiFi-Nano: Reclaiming WiFi Efficiency Through 800 ns Slots. In *Proc. of MobiCom*, 2011.

[32] Seongho Byeon, Kangjin Yoon, Okhwan Lee, Sunghyun Choi, Woonsun Cho, and Seungseok Oh. MoFA: Mobility-aware Frame Aggregation in Wi-Fi. In *Proc. of CoNEXT*, 2014.

[33] J. Jermyn, R.P. Jover, I. Murynets, M. Istomin, and S. Stolfo. Scalability of Machine to Machine systems and the Internet of Things on LTE mobile networks. In *International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015.

[34] 3GPP TDocs, Meeting: R1-83 - 2015-11-15 to 2015-11-22, Anaheim. `http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/`. [Online; accessed 25-Jan-2016].

[35] Junxian Huang, Feng Qian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In *Proc. of SIGCOMM*, 2013.

[36] Shuo Deng and Hari Balakrishnan. Traffic-aware Techniques to Reduce 3G/LTE Wireless Energy Consumption. In *Proc. of CoNEXT*, 2012.

[37] Mi Kyung Han, Brian Overstreet, and Lili Qiu. Greedy Receivers in IEEE 802.11 Hotspots. In *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '07, pages 471–480, Washington, DC, USA, 2007. IEEE Computer Society.

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

[39] NS–3. https://www.nsnam.org/. [Online; accessed 14-Apr-2016].

[40] A.M. Cavalcante, E. Almeida, R.D. Vieira, F. Chaves, R.C.D. Paiva, F. Abinader, S. Choudhury, E. Tuomaala, and K. Doppler. Performance Evaluation of LTE and Wi-Fi Coexistence in Unlicensed Bands. In *VTC Spring*, 2013.

[41] Tomi Nihtila, Vitaliy Tykhomyrov, Olli Alanen, Mikko A Uusitalo, Antti Sorri, Martti Moisio, Sassan Iraji, Rapeepat Ratasuk, and Nitin Mangalvedhe. System performance of LTE and IEEE 802.11 coexisting on a shared frequency band. In *Proc. of WCNC*, 2013.

[42] Jeongho Jeon, Qian Clara Li, Huaning Niu, Apostolos Papathanassiou, and Geng Wu. LTE in the Unlicensed Spectrum: A Novel Coexistence Analysis with WLAN Systems. In *Proc. of GLOBECOM*, 2014.

[43] Lili Qiu Sangki Yun. Supporting WiFi and LTE Co-existence. In *Proc. of INFOCOM*, 2015.

[44] E. Almeida, A.M. Cavalcante, R.C.D. Paiva, F.S. Chaves, F.M. Abinader, R.D. Vieira, S. Choudhury, E. Tuomaala, and K. Doppler. Enabling LTE/WiFi coexistence by LTE blank subframe allocation. In *Proc. of ICC*, 2013.

[45] Supratim Deb, Pantelis Monogioudis, Jerzy Miernik, and James P Seymour. Algorithms for Enhanced Inter-Cell Interference Coordination (eICIC) in LTE HetNets. *Trans. on Networking*, 22(1):137–150, February 2014.

[46] Shweta Sagari, Samuel Baysting, Dola Saha, Ivan Seskar, Wade Trappe, and Dipankar Raychaudhuri. Coordinated dynamic spectrum management of LTE-U and Wi-Fi networks. In *Proc. of DySPAN*, pages 209–220. IEEE, 2015.

[47] C. Cano and D.J. Leith. Coexistence of WiFi and LTE in Unlicensed Bands: A Proportional Fair Allocation Scheme. In *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pages 2288–2293, June 2015.

[48] Božidar Radunović, Ranveer Chandra, and Dinan Gunawardena. Weeble: Enabling Low-power Nodes to Coexist with High-power Nodes in White Space Networks. In *Proc. of CoNEXT*, 2012.

[49] Daniel Halperin, Thomas Anderson, and David Wetherall. Taking the Sting out of Carrier Sense: Interference Cancellation for Wireless LANs. In *Proc. of MobiCom*, 2008.

[50] Shyamnath Gollakota, Fadel Adib, Dina Katabi, and Srinivasan Seshan. Clearing the RF Smog: Making 802.11N Robust to Cross-technology Interference. In *Proc. of SIGCOMM*, 2011.

[51] Aditya Gudipati and Sachin Katti. Strider: Automatic Rate Adaptation and Collision Handling. In *SIGCOMM*, 2011.

[52] Aruna Balasubramanian, Ratul Mahajan, and Arun Venkataramani. Augmenting Mobile 3G Using WiFi. In *Proc. of MobiSys*, 2010.

[53] Rajesh Mahindra, Hari Viswanathan, Karthik Sundaresan, Mustafa Y Arslan, and Sampath Rangarajan. A Practical Traffic Management System for Integrated LTE-WiFi Networks. In *Proc. of MobiCom*, 2014.

[54] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[55] Xin Wang and K. Kar. Throughput Modelling and Fairness Issues in CSMA/CA Based Ad-Hoc Networks. In *Proc. of INFOCOM*, 2005.

[56] G. Caire, Sean A. Ramprashad, and H.C. Papadopoulos. Rethinking Network MIMO: Cost of CSIT, Performance Analysis, and Architecture Comparisons. In *Information Theory and Applications Workshop (ITA)*, 2010.

[57] Rice University Wireless Open Access Research Platform.

[58] Krishna C. Garikipati and Kang G. Shin. Improving Transport Design for WARP SDR Deployments. In *Proc. of the ACM SIGCOMM Software Radio Implementation Forum (SRIF)*, 2014.

[59] Q.H. Spencer, A.L. Swindlehurst, and M. Haardt. Zero-forcing Methods for Downlink Spatial Multiplexing in Multiuser MIMO channels. *IEEE Transactions on Signal Processing*, 52(2):461–471, 2004.

[60] M.X. Gong, E. Perahia, R. Want, and S. Mao. Training protocols for multi-user MIMO wireless LANs. In *Proc. of PIMRC*, 2010.

[61] N. Jindal. MIMO Broadcast Channels With Finite-Rate Feedback. *IEEE Transactions on Information Theory*, 52(11):5045–5060, 2006.

[62] R. Porat, E. Ojard, N. Jindal, M. Fischer, and V. Erceg. Improved MU-MIMO Performance for Future 802.11 Systems Using Differential Feedback. In *Information Theory and Applications Workshop (ITA)*, 2013.

[63] J.C. Roh and B.D. Rao. Efficient Feedback Methods for MIMO Channels Based on Parameterization. *IEEE Transactions on Wireless Communications*, 6(1):282–292, 2007.

[64] E. Perahia, A. Sheth, T. Kenney, R. Stacey, and D. Halperin. Investigation into the Doppler Component of the IEEE 802.11n Channel Model. In *Proc. of GLOBECOM*, 2010.

[65] F. Boccardi, H. Huang, and M. Trivellato. Multiuser Eigenmode Transmission for MIMO Broadcast Channels with Limited Feedback. In *IEEE Workshop on Signal Processing Advances in Wireless Communications*, 2007.

[66] M. Kendall. *Rank Correlation Methods*. 1990.

[67] Cisco Systems Inc. Wireless Mesh Access Points, Design and Deployment Guide, Release 7.4, 2013.

[68] X. Zhang, K. Sundaresan, M. Khojastepour, S. Rangarajan, and K. Shin. NEMOx: Clustered Network MIMO for Wireless Networks. In *Proc. of MOBICOM*, 2013.

[69] G. Caire, N. Jindal, M. Kobayashi, and N. Ravindran. Multiuser MIMO Achievable Rates With Downlink Training and Channel State Feedback. *IEEE Transactions on Information Theory*, 56(6):2845–2866, 2010.

[70] S. Singh, K. Sundaresan, A. Khojastepour, S. Rangarajan, and S. Krishnamurthy. One Strategy Does Not Serve All: Tailoring Wireless Transmission Strategies to User Profiles. In *Proc. of ACM Workshop on Hot Topics in Networks*, 2012.

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

[72] Kun Tan, Jiansong Zhang, Ji Fang, He Liu, Yusheng Ye, Shen Wang, Yongguang Zhang, Haitao Wu, Wei Wang, and Geoffrey M. Voelker. Sora: High performance software radio using general purpose multi-core processors. In *Proc. of NSDI*, pages 75–90, 2009.

[73] Navid Nikaein. Processing radio access network functions in the cloud: Critical issues and modeling. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, 2015.

[74] Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10):1374–1396, 1995.

[75] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A Scalable, Commodity Data Center Network Architecture. In *Proc. of SIGCOMM*, 2008.

[76] Qin Zheng and K. G. Shin. On the ability of establishing real-time channels in point-to-point packet-switched networks. *IEEE Transactions on Communications*, 42(234):1096–1105, Feb 1994.

[77] J. P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of Real-Time Systems Symposium*, 1990.

[78] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real-time Communication in Multi-hop Networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(10):1044–1056, 1994.

[79] K. W. Tindell, A. Burns, and A. J. Wellings. An Extendible Approach for Analyzing Fixed Priority Hard Real-time Tasks. *Real-Time Systems*, 6(2):133–151, 1994.

[80] M. Karol, M. Hluchyj, and S. Morgan. Input Versus Output Queueing on a Space-Division Packet Switch. *IEEE Transactions on Communications*, 35(12):1347–1356, 1987.

[81] Laurent Schumacher, Klaus I. Pedersen, Preben E. Mogensen, Niels Jernes Vej, and Dk-Aalborg st. From Antenna Spacings To Theoretical Capacities - Guidelines For Simulating Mimo Systems. In *IEEE PIMRC*, 2002.

[82] R. Bhatia. *Perturbation Bounds for Matrix Eigenvalues*. Society for Industrial and Applied Mathematics, 2007.

[83] A Performance Study of CPRI over Ethernet. `http://www.ieee1904.org/3/meetingarchive/2015/02/tf31502ashwood1a.pdf`. [Online; accessed 14-Apr-2016].

[84] E. Chai, KG. Shin, SJ. Lee, J. Lee, and R. Etkin. SPIRO: Turning Elephants into Mice with Efficient RF Transport. In *Proc. of INFOCOMM*, 2015.

[85] Karl F. Nieman and Brian L. Evans. Time-Domain Compression of Complex-Baseband LTE Signals for Cloud Radio Access Network. In *Proc. of IEEE SIP*, 2013.

[86] Yun Wang and M. Saksena. Scheduling fixed-priority tasks with preemption threshold. In *International Conference on Real-Time Computing Systems and Applications*, 1999.

[87] ZhenBo Zhu, Parul Gupta, Qing Wang, Shivkumar Kalyanaraman, Yonghua Lin, Hubertus Franke, and Smruti Sarangi. Virtual base station pool: Towards a wireless network cloud for radio access networks. In *Proceedings of the 8th ACM International Conference on Computing Frontiers*, 2011.

[88] Wenfei Wu, Li Erran Li, Aurojit Panda, and Scott Shenker. PRAN: Programmable radio access networks. In *HOTNETS*, 2014.

[89] Yingfeng Oh and Sang H. Son. Allocating Fixed-priority Periodic Tasks on Multiprocessor Systems. *Real-Time Systems*, 9(3):207–239, 1995.

[90] Julie Baro, Frédéric Boniol, Mikel Cordovilla, Eric Noulard, and Claire Pagetti. Off-line (optimal) multiprocessor scheduling of dependent periodic tasks. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 1815–1820, 2012.

[91] Joseph Y.-T. Leung and Jennifer Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2(4):237 – 250, 1982.

[92] Bjorn Andersson, Sanjoy Baruah, and Jan Jonsson. Static-priority Scheduling on Multiprocessor. In *Proceedings of IEEE Real-Time Systems Symposium*, pages 193–202, 2001.

[93] A. Srinivansan and Sanjoy K. Baruah. Deadline-based Scheduling of Periodic Task Systems on Multiprocessors. *Information Processing Letters*, 84:93–98, 2002.

[94] Theodore Baker and Sanjoy Baruah. Schedulability Analysis of Multiprocessor Sporadic Task Systems. In *Handbook of Real-Time and Embedded Systems*. Chapman Hall/ CRC Press, 2007.

[95] John Carpenter, Shelby Funk, Phil Holman, Anand Srinivasan, Jim Anderson, and Sanjoy Baruah. A Categorization of Real-time Multiprocessor Scheduling Problems and Algorithms. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC.

[96] Qi Zheng, Yajing Chen, Hyunseok Lee, Ronald Dreslinski, Chaitali Chakrabarti, Achilleas Anastasopoulos, Scott Mahlke, and Trevor Mudge. Using Graphics Processing Units in an LTE Base Station. *Journal of Signal Processing Systems*, 78(1):35–47, 2015.

[97] Z. Shi, E. Jeannot, and J. J. Dongarra. Robust Task Scheduling in Non-Deterministic Heterogeneous Computing Systems. In *Cluster Computing, 2006 IEEE International Conference on*, pages 1–10, Sept 2006.

[98] A. Bastoni, B.B. Brandenburg, and J.H. Anderson. Is Semi-Partitioned Scheduling Practical? In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 125–135, 2011.

[99] LTE Physical Layer Procedures. `http://www.etsi.org/deliver/etsi_ts/136200_136299/136213/08.08.00_60/ts_136213v080800p.pdf`. [Online; accessed 29-Nov-2015].

[100] KVM. `http://www.linux-kvm.org/`. [Online; accessed 29-Jan-2016].

[101] Cyclictest. `https://rt.wiki.kernel.org/index.php/Cyclictest`. [Online; accessed 29-Nov-2015].

[102] Hackbench. `http://manpages.ubuntu.com/manpages/trusty/man8/hackbench.8.html`. [Online; accessed 29-Nov-2015].

[103] CPRI Specification V6.1. `http://www.cpri.info/downloads/CPRI_v_6_1_2014-07-01.pdf`. [Online; accessed 29-Nov-2015].

[104] Krishna C. Garikipati and Kang G. Shin. Improving transport design for warp sdr deployments. In *Proceedings of the 2014 ACM Workshop on Software Radio Implementation Forum*, SRIF '14, 2014.

[105] Open Air Interface. `http://www.openairinterface.org/`. [Online; accessed 29-Nov-2015].

[106] RTLinux. `http://rt.wiki.kernel.org/`. [Online; accessed 29-Jan-2016].

[107] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft, and R. Zainlinger. Distributed Fault-Tolerant Real-Time Systems: The Mars Approach. *IEEE Micro*, 9(1):25–40, 1989.

[108] Thomas Wagner, Alan Garvey, and Victor Lesser. Design-to-criteria scheduling: Managing complexity through goal-directed satisficing. In *Proceedings of the AAAI-97 Workshop on Building Resource-Bounded Reasoning Systems*. Citeseer, 1997.

[109] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. Characterizing Cloud Computing Hardware Reliability. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, 2010.

[110] S. R. Biyabani, J. A. Stankovic, and K. Ramamritham. The Integration of Deadline and Criticalness in Hard Real-Time Scheduling. In *Real-Time Systems Symposium, 1988., Proceedings.*, pages 152–160, Dec 1988.

[111] Amarisoft. `http://amarisoft.com/`. [Online; accessed 29-Jan-2016].

[112] L. Zhao, M. Li, Y. Zaki, A. Timm-Giel, and C. Gorg. LTE virtualization: From Theoretical Gain to Practical Solution. In *Proc 23rd ITC*, 2011.

[113] Chengchao Liang and F.R. Yu. Wireless Network Virtualization: A Survey, Some Research Issues and Challenges. *Communications Surveys Tutorials, IEEE*, 17(1):358–380, Firstquarter 2015.

[114] Aditya Gudipati, Daniel Perry, Li Erran Li, and Sachin Katti. SoftRAN: Software Defined Radio Access Network. In *HotSDN*, 2013.