Title
Improving the Range and Latency in 802.11ac Networks

Author
Yin, Patrick

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Improving the Range and Latency in 802.11ac Networks

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Electrical and Computer Engineering

by

Patrick Yin

2020

ABSTRACT OF THE THESIS

Improving the Range and Latency in 802.11ac Networks

by

Patrick Yin
Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2020
Professor Songwu Lu, Co-Chair
Professor Greg Pottie, Co-Chair

As IoT and drone applications become increasing popular, there exists a demand to be able to increase communications range. In addition, with possible applications in video streaming/AR/VR, the necessity for a low-latency scheme to reduce overall transmission delay becomes more critical. Therefore, we approach this problem on two fronts – one on the PHY layer to solve the long-range issue, and the other on the MAC layer to solve the latency issue. There are preexisting works on long range communications, but these use directional antennas, high powered amplifiers, and protocol customizations on the MAC layer. Unfortunately, these approaches are not suited for the spread out and mobile nature of IoT and drone applications. Therefore, we focus on the PHY layer, specifically on the concept of predistortion, to offer our solution: increased linear power that would offer a boost in omnidirectional range without the need of directional antennas. Although much work has been done to reduce overall latency, the methods do not completely address our problem. To meet millisecond level requirements for delay sensitive applications, we recognize that the long tail of the packet delay distribution poses a significant issue. Although different

mitigation techniques can be developed on various layers, we choose a path that would offer maximum marketability: a software-based solution at the MAC layer that offers the maximum amount of delay reduction without changing components at the physical level. We can therefore couple our long-range solution in PHY with our low latency resolution in MAC in the form of an API that would allow for easy installation and customizations of commodity Wi-Fi systems.

The thesis of Patrick Yin is approved.

Izhak Rubin

Greg Pottie, Committee Co-Chair

Songwu Lu, Committee Co-Chair

University of California, Los Angeles

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Firstly, I would like to acknowledge and express sincere appreciation to my research advisor and committee co-chair, Professor Songwu Lu, for giving me the opportunity to contribute to the frontier of this technology frontier in the form of a novel research project. Without a doubt, this experience has been the defining point of my academic career. It was his direction and leadership that made this all possible.

I would also like to give special recognition and thanks to my academic advisor and committee co-chair Professor Greg Pottie, who has been a steady well of insight and guidance throughout my time at UCLA. It was his guidance that pushed me towards a thesis track, and it was his brief hints and insights that allowed me to make the progress that I have made. Under his tutelage, I was able to push myself to new and greater heights.

An additional special thanks to Professor Izhak Rubin for, when he allowed me to TA for his ECE 132B class, provided me an environment where I could refresh my knowledge of the MAC layer and discover addition inspiration to further my research.

Finally, I would like to express the deepest gratitude to Zhaowei Tan for mentoring me through the process. It was his patience and advice that allowed me to transition from an industry-based mindset to an academia-based approach when it came to research. Our weekly meetings for the better part of a year were a wonderful exchange of thoughts and ideas, and I would like to recognize him for integral role he played in this process.

CHAPTER 1

Introduction

With Wi-Fi connectivity slowly permeating to the more rural parts of the world, we expect demand to rise for a Wi-Fi network with capabilities beyond that of a standard commercial system. Not only would the need for a wide-area outdoor network arise, but applications beyond that of simple internet access become critical. Low-latency sensitive applications like virtual reality or augmented reality require lower latency than what can currently be met. We therefore aim to create a solution in the form of a customizable API as a supplement to current 802.11 systems that focuses on improving the range and latency independently.

When talking about range within a home network, the concept of multipath is critical. Multipath is the bouncing of a signal off objects such that the same signal may be received multiple times from different paths. In such a case, the delay spread of such interference must be accounted for in the duration of the individual transmitted symbols, otherwise fading channels and uneven in-band frequency responses would occur. Within the rural environment that we are focusing on, multipath does not serve as a large hindrance towards range [1]. Since we want a customizable solution, we look at improvements that can be made in the form of a software installation. We do not look at any kind of channel coding, as most range related work has gotten close to theoretical limits. Instead, we look at the circuitry within the actual transceiver. Although more promising changes can be made at the device level, we choose to look at reducing the sideband interference generated at high power by the output amplifier. In a nonlinear device such as an amplifier, as it gets driven more into the non-linear region, where the power gets diverted away from the fundamental signal and

more into the harmonics and intermodulation products. These are detrimental as they may cause interference in neighboring bands in the frequency domain. We look to reduce the magnitude of these products with linearization. In this way, we can pre-process the signal using a set of pre-determined coefficients in a lookup table applied at the FPGA level.

The latency we are looking to reduce occurs mostly at the MAC layer. More specifically, we look to minimize the time between when a packet enters the queue at the link layer from the upper layers of the transmitter and when it receives ACK from the receiver. The specific mechanisms that affect this time would naturally be the retransmission rate, and how long a given packet spends in the queue. Quite obviously, increasing the data rate of a given flow would reduce the latency, due to the time saved during transmission [2]. However, it has been shown [3] that these increased rates do not necessary improve tail latency, which is defined in various papers as the a^{th} percentile, generally 90^{th} or higher, of a packet latency distribution. The tail latency becomes important in AR/VR and other latency sensitive applications that would require a very high a^{th} percentile of packets received within a specific time. In this paper we will analyze how these delays are incurred, and our proposed improvement algorithm.

CHAPTER 2

Background

With advancements in drone and IoT technology, we expect increasingly complex demands, including a need for increased range. Our specific target application would be an outdoor drone network, for geological or oil field survey purposes. Even more simply, local farmers can use a form of that application to search for lost members of their herd, broken or failing infrastructure that needs repair, or just an aerial view for an additional vantage point. This would necessitate a cheap, commonly available, and highly customizable product that can be installed without large amounts of pre-requisite equipment. Therefore, we look towards a standard home Wi-Fi access point to service these highly mobile drones.

The requirements for this solution would be a customizable API that can not only support longer range than a standard home access point, but for live survey purposes, would also need to support the higher data rate and lower latency requirements for HD video streaming. Drone video capture is nothing new, but these surveys necessitating real-time decisions would either require highly sophisticated general image recognition, or a human monitoring a live video stream. As such a sophisticated image recognition program implies a large and expensive neural network, it would not only be simpler but a lot cheaper to implement the network requirements for HD live streaming, and by extension, an AR/VR interface.

While our proposed application may be a lofty goal by the most optimistic standards, the individual aspects of increased range and decreased latency do hold value. We can split the problem and approach it from two independent fronts, with a combined solution aiming to be a supplement to pre-existing 802.11 standards. To do this, we will need to target areas

that are either insufficient or are unspecified within the current standard.

Neither problem is unique: long range communications exists in form of LTE and Satcoms to name a few, and AR/VR/HD video streaming already exists in one capacity or another. We therefore look at the technology implemented by these systems, not with the goal of breaking the record for any metrics, but to survey the current state of the art and see whether novel improvements can be made on our end.

2.1 Related Work: Range

The fundamental roadblock to range is the perceived signal to noise ratio (SNR) at the receiver. Intuitively, the farther away a receiver is from a source, the harder it is to distinguish what is being transmitted, due to the relative noise power received. At the edge of the network, the received signal strength is low, corresponding to a low SNR and indicating a high probability that the receiver makes an error, for example perceiving a bit stream of 1111 to be 1110. This error would be less likely to happen closer to the source with a higher SNR. The goal is to then somehow increase the SNR such that these errors do not happen.

2.1.1 Forward Error Correction

Error correction, and more specifically forward error correction, allows us to do precisely this. The method involves sending redundant bits such that the system is more tolerant of errors. For example, a very simple algorithm can transmit a single bit 3 times, such that 111, 110, 101 etc. would all be interpreted as a 1 at the receiver, allowing for errors to be made, and act as the equivalent of signal gain. There are obvious bandwidth tradeoffs, but mathematically, due to the Shannon limit [4], we can achieve a theoretical gain of 9dB. Therefore, it would make sense to target this method and analyze whether any improvements can be made. In fact, most protocols come standard with some kind of FEC, implemented on the PHY layer as part of the MCS (modulation and coding scheme) table [5] [6]. However,

the primary issue is that these techniques are already so advanced that the standard code specified by 802.11, LDPC (low density parity check), is already very close to the Shannon Limit. Most current research focuses on decreasing the latency or complexity of the process, so there is not much in terms of range improvements on this front.

2.1.2 Ultra-Long Range and Satcom Networks

Recognizing that there are other networks and systems that have achieved long range communications, we will take this section to analyze the methodology of two in particular: ultra long range Wi-Fi networks and satellite communications systems, and subsequently comment on the suitability, aspects of potential interest, and ways they can be built upon to yield a better solution.

For ultra long range networks like WiLDNet and DakNet [1] [7], which can achieve links of up to 100km, they are able to satisfy the link budget deficiency by utilizing high power wireless cards or amplifiers with highly directional antennas set on high locations, thus avoiding multipath propagation issues. With up to 24dBi antenna gain per node, these networks are thus able to operate their amplifiers in the linear region and avoid nonlinear distortion. Coincidentally, another aspect of their approach was to implement what was at the time a novel FEC algorithm to supplement the link budget. However, given that the antennas have a very narrow coverage area over a long distance, very precise tuning is needed to align a pair of nodes, making mobility infeasible. Additionally, the error correction they implemented is now being outperformed by conventional codes [1] [6]

It is well known that satellite communications operate their amplifiers in saturation, or close to, for efficiency reasons [8]. For those applications, efficiency becomes a more critical specification, and therefore different modulations are used to accommodate the fact. While standard 802.11 protocols have an MCS lookup table that utilizes different orders of QPSK and QAM modulation under varying channel conditions, satcoms tend to employ an APSK scheme, as the circular constellation provides a better PAPR than QAM at higher orders,

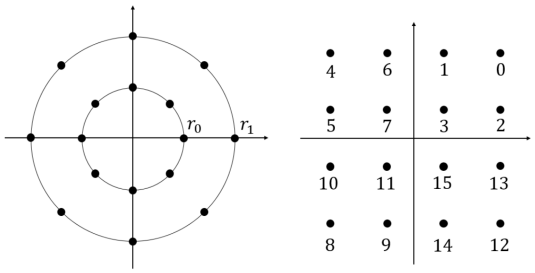

Figure 2.1: Signal Constellations of 16-APSK (left) and 16-QAM (right)

as shown in Figure 2.1 [9]. In addition, a popular standard for satellite communications, IRIG, defines several alternative modulation schemes for space-based communications like a pulse code modulation (PCM)/frequency modulation known as filtered continuous phase frequency shift keying (CPFSK) [10], shaped offset quadrature phase shift keying (SOQPSK-TG), or Feher's quadrature phase shift keying (FQPSK-B); the latter two of which are used for higher bandwidth efficiency requirements. All such modulations are employed due to the constant envelope characteristics which provide minimal spectral regrowth and signal degradation under nonlinear amplifier operation (saturation). However, QAM modulations are more resistant to Rayleigh fading channels; a type of scattering of the propagation of radio waves more common in ground-based wireless communications [11]. Since our goal is a customizable API targeting earth-bound applications, the current MCS index of 802.11ac works well enough that a new modulation scheme does not provide enough benefits to justify the complexities and difficulties of such a change.

2.2 Related Work: Latency

A natural area to look at when reducing latency is the number of retransmissions of the packets being sent. Re-transmission, fundamentally, is a product of error control methods such as ARQ, implemented on the MAC layer. A re-transmission is made when a packet is received in error. To limit the number of re-transmissions during a link, one would rea-

sonably try to limit the number of errors made. These errors present at the bit level, and on a larger scale, the packet level, and correspond to the bit error rate (BER) and packet error rate (PER), respectively. As it is impossible to transmit wireless without errors, methods such as Forward Error Correction (FEC) and Cyclic Redundancy Checks (CRC) allow for some transmissions to be made in error and still maintain a given throughput. The presence of channel coding effectively as signal gain, improving SNR and allowing normally incoherently noisy signals to be received correctly. The utilization of an MCS lookup table at PHY provides further resiliency against poor integrity channels, as a network can automatically decrease the data rate to support a very lossy link, without having to terminate the connection entirely. However, there are mathematical limits to these mechanisms [4], and re-transmission is unavoidable. Ultra long range networks like WiLDNet have utilized a bulk acknowledgement mechanism at MAC to reduce the latency from re-transmissions and increase utilization of the channel [1]. This was done to compensate for the long round trip times (RTT) of each link and does not specifically target the long tail latency that we are concerned with.

2.2.1 Rate Adaptation in LTE

We therefore must acknowledge that different protocols approach this problem differently and optimize their algorithms for different parameters. LTE, for instance, implements a channel estimation algorithm at the PHY layer, dynamically calculating channel coefficients in the form of the $\mathbf{H}$ matrix, in a system modelled as $\mathbf{y} = \mathbf{Hx} + \mathbf{n}$, with $\mathbf{x}$ and $\mathbf{y}$ being the transmit and receive signal, and $\mathbf{n}$ being the associated noise vector. In this way, the channel coefficients can be used to determine SNR and the subsequent fastest transmission rate via a modulation and coding scheme (MCS) lookup table. On the other hand, 802.11 protocols determine data rate a bit differently. While LTE uses SNR to determine the largest modulation supported by the channel, Wi-Fi standards generally implement channel estimation at the MAC layer, in the form of implicit or explicit probing of packets to determine packet

error rate (PER). Explicit probing involves using specific test packets to determine the best data rate that a transmitter should transmit at, given the channel conditions at the time. Implicit probing on the other hand uses live data packets. Given a current transmitting rate, during the probing period, an implicit probing method involves sending live packets at one rate higher than the current rate (determined by the MCS). It will repeat this process until it is transmitting at an unacceptable PER, and it will then settle for the lower rate. This leaves a lot of room for improvements, and we analyze possible methods of improving our latency. While current standards have specific rate adaptation algorithms to maximize the throughput (lost packets will affect data rate too), no central scheduler is specified within 802.11 [12], so latency times are not guaranteed or regulated.

2.2.2 VR and Large Scale Networks

Building on this, we recognize that while latency reduction has been studied extensively, not a lot of work has been done on latency specific rate adaptation. Solutions have been put forth targeting specific applications, like adaptive probing for latency reduction in HD video streaming. While not as stringent, the large amount of data being transferred also requires some latency consideration. While encoding techniques like VBR (Variable Bit Rate) video encoding have drastically reduced the average streaming, it has been found that insufficiencies still exist on the link [13]. In other words, although the specific 802.11 link was rated up to a certain speed, given the different MIMO modes, correctable packet losses were still happening in the form of implicit probing. Implicit probing, done on almost all legacy rate adaptation protocols to reduce overhead [14], is the practice of sending a packet at a higher PHY rate that the current rate. If the packet was successful, then the channel was deemed adequate to support this new rate, and the next highest rate is probed. The process continues until a predetermined number of packets fail, then the last successful rate is chosen. Quite obviously, this is quite inefficient and incurs large amounts of packet losses as inadequate rates are being tested. It will be shown in this thesis that the packet losses

are critical metrics for latency at the MAC layer. This is the rate adaptation that we will build our own algorithm from.

Outside of a rate adaptation algorithm, there are other methods to determining and reducing latency. These efforts usually occur on higher layers, utilizing protocols such as the Simple Network Management Protocol on the application layer. One such work targets latency in large scale networks [15]. However, they measure end-to-end latency, all the way up to the application layer. This incurs large delays at the Transport and Network layer. Although they recognized that low latency can be exacerbated with a bad connection, and within a large enterprise wireless network, (i.e., campus, metropolitan area, IoT network), a user device does not choose the best access point in terms of latency and speed. It identifies that channel utilization, channel congestion, and SNR are most important for latency sensitive AP selection rather than the widespread RSSI (received signal strength indicator). They then construct a decision tree and use a machine learning algorithm to choose the best AP. Based on these factors, although they validate our initial theories on latency reduction, the effort is deemed out of scope.

CHAPTER 3

PHY Layer: Digital Predistortion

3.1 Challenges to Range Increase

First, we look at range at the PHY layer. From the Friis transmission equation, we know that the ratio of power received to power transmitted is inversely proportional to distance squared. Various models of transmission under different conditions (i.e., free space, two-ray, Okumura-Hata) also confirm the relationship of transmitted power to distance of varying orders. Therefore, we can conclude that, fundamentally, if we want to increase the range of our system, we need to increase the power of the output transmitter. However, an increase in power comes with a cost: as the amplifier is driven harder, the device is pushed into a saturation region where non-linearities start showing up and affecting the signal integrity. These non-linearities show up as spurs in the neighboring frequency bands, which will interfere with our intended signals and prevent coherent transmission.

A nonlinear device, by definition, is a device where the input power and output power does not plot as a straight line. Most amplifiers are linear backed off, but once it is driven close to the rails, the signal starts clipping, and the output power stagnates. This is called the saturation region of the amplifier and is nonlinear in nature: while the input power increases, output power does not increase by the same amount. Generally, the response for an amplifier can be written as follows:

$$Y = A_0 + A_1 x + A_2 x^2 + \cdots + A_n x^n \tag{3.1}$$

where Y is the output and x is the input. For amplifiers in the linear region, coefficients $A_i = 0 \quad \forall \quad i > 1$. As the amplifier gets more into saturation, the excess power is diverted into the nonlinear products: x^2, x^3, etc. The magnitudes of the coefficients of these products change the closer the amplifier gets to saturation and is responsible for the deviation away from the linear equation. Assuming x is a single sinusoidal input with frequency ω, $x = Acos(\omega t + \phi)$, the non-linearities present as harmonics: x^2, x^3 correspond to 2ω and 3ω on the frequency domain. This can easily be proven by expressing the input as the first term of its Euler form, where $x = Ke^{j(\omega t+\phi)}$ [16].

However, the input is seldom a single tone. And particularly for 802.11ac, the transmission is at least a 64 sub-carrier OFDM signal. We can approximate what happens with a two-tone analysis, $x = x_a + x_b$. The expansion for the first 3 terms is as follows:

$$A_1 \times x = A_1(x_a + x_b) \tag{3.2}$$

$$A_2 \times x = A_2(x_a + x_b)^2 = A_2(x_a^2 + 2x_a x_b + x_b^2) \tag{3.3}$$

$$A_3 \times x = A_3(x_a + x_b)^3 = A_3(x_a^3 + 2x_a^2 x_b + 2x_a x_b^2 + x_b^3) \tag{3.4}$$

The first term is a linear representation. If the device is fully linear, then the output will only yield the original input terms.

The second term yields a square product and produces $cos^2(x) = \frac{1}{2}(cos(2x) + 1)$. While this yields a harmonic at twice the frequency, 2ω, it also produces a DC term. This term is the DC offset and will directly raise the noise floor as part of the $\frac{1}{f}$ flicker noise. This would not be an issue for standard superheterodyne receivers, as the low frequency components get filtered out by the receive filter. However, our setup, as well as all cellphones and most Wi-Fi enabled devices, use a direct conversion receiver architecture, which necessitates consideration of the DC offset product and flicker noise. Direct conversion receivers couple the input into the LO to self-mix the received signal to baseband. Because of these, it is

extremely susceptible to sensitivity degradation from the aforementioned products [17] [18]. Sensitivity is measured as the weakest signal that a receiver can identify, measured against the noise floor. If a transmitter sends a DC component along with its intended signal, the noise floor at the receiver will rise and require a higher signal power to accurately resolve.

With the third term, the primary issue is the $3a^2b$ and $3ab^2$ which will yield $2\omega_a - \omega_b$ and $2\omega_b - \omega_a$ terms, from the Euler identity. If ω_a and ω_b are closely spaced, which they will be in a OFDM scheme, these intermodulation products will show up in the side band, close to the original signals, causing interference with neighboring subcarriers. In addition, 802.11ac is capable of modulation schemes up to 256-QAM, which will necessitate consideration of nonlinear amplifier effects on accuracy of demodulation. In QAM, edge points on the constellation diagram require higher power than the center points due to the modulation scheme, which modulates both amplitude and phase. This creates peak to average power ratio issues, as it requires a greater range of linear power from the amplifier to properly transmit, otherwise either edge points will saturate and cause sideband interference, or center points will not have the power to coherently transmit to the receiver. Again, a receiver's ability to resolve an input signal depends on its signal-to-noise ratio (SNR), which is a direct function of the power at the receiver. A low input power means a low SNR, which will cause demodulation errors, and prevent accurate information transfer.

One can therefore conclude that for a fixed output amplifier, a device operating under the 802.11ac protocol is inherently range limited. Beyond a certain distance, the dynamic range required to maintain a set data speed is unachievable, as one would either generate sideband interference at high power, or high SNR leading to bit errors at low power.

Now that we have analyzed all problems associated with increasing the drive of an amplifier as well as the various incompatibilities of pre-existing solutions, we identify the concept of predistortion as the most suitable path forward for our application. Predistortion is part of a family of linearization techniques, primarily used to improve the efficiency and output power of amplifiers. One of the more well-known applications is in the implementation of a

Doherty amplifier, which has seen a resurgence in 4G and 5G MIMO systems for its high efficiency at back off. With up to 256-QAM modulation in 802.11ac, the probability of an amplifier operating at peak power is very low, so high efficiency is required when the amplifier is backed off from saturation. Modern implementations of the Doherty utilize dynamic load modulation [19], which modulates the load to improve back off efficiency. This is supplemented by the predistortion linearizer, which suppresses the additional spurs generated from load modulation.

For us, we will use the subsequent sections to analyze how predistortion can mitigate the adjacent channel leakage ratio (ACLR) and the sideband spurs generated when our system is run close to saturation.

3.2 Predistortion Algorithm: Introduction

Predistortion is a method of pre-processing the input signal to invert the coefficients of the expected nonlinear products generated by the amplifier such that the magnitude of the intermodulation products at the output are significantly lower than what it would be without the processing. However, certain variables like number and density of coefficients, and most noticeably the dynamic range of the ADC, will create a limit on the correctable power [20], as shown in Figure 3.1. Even so, a predistorted signal would still experience a higher output power and lower intermodulation products given the same input power, allowing for an increase in range.

This fits into our intent of delivering a customizable API without the need of modifying the hardware at the transmitter.

There are essentially two methods of predistortion: feed-forward and feedback. The simplest approach is feedback, where we take the output of the amplifier in question, apply some adaptation or processing, then feed it back to the input. Once again, this type of feedback can be split into RF or analog predistortion and digital predistortion. RF predistortion is

an implementation of the most basic concept: it directly routes the processed output back to the input of the amplifier. This does not take into account the noise generation, insertion loss, and nonlinearities of the previous stages, namely the mixer, the filter required to maintain the bandwidth expansion, and the digital to analog converter converters [21]. The noise requirements of these components would require additional filtering and consideration at the output of the amplifier. Therefore, we turn to digital predistortion, which downconverts the output signal before applying some adaptation algorithm directly to the digital baseband, bypassing the aforementioned requirements.

Digital predistortion implementation methods can be categorized into either memoryless models or models with memory. Memorylessness, via a statistical definition, is the characterization that the future evolution of the process is independent of the past given the present state. A linear amplifier is memoryless as its output only depends on the input with some gain and a constant time delay, as the signal needs to propagate through the amplifier. It is only when we get into the nonlinear region that we need to consider memory effects. As shown above, nonlinearity generates intermodulation products in the frequency domain. Including the consideration of the phase ϕ in the equation $x = ke^{j(\omega t+\phi)}$, it becomes clear that the phase shift is frequency dependent. In the event that two or more products land on the same frequency, which is inevitable given the OFDM nature of our application, the amplitude then becomes phase dependent. This proves that a nonlinear amplifier has memory effects, as a frequency dependence in the frequency domain is time dependence in the time domain.

Given the nonlinearity was previously described using a Taylor series expansion, nonlinearity with memory can be modelled using the expanded Volterra series and its derivatives. The complexity of its implementation and large number of coefficients required make this impractical. Additionally, memory effects become more apparent under wider bandwidth and higher power [22], so for our applications we can safely focus on the memoryless model. This model assumes instantaneous nonlinearity, characterized by the AM/AM and AM/PM

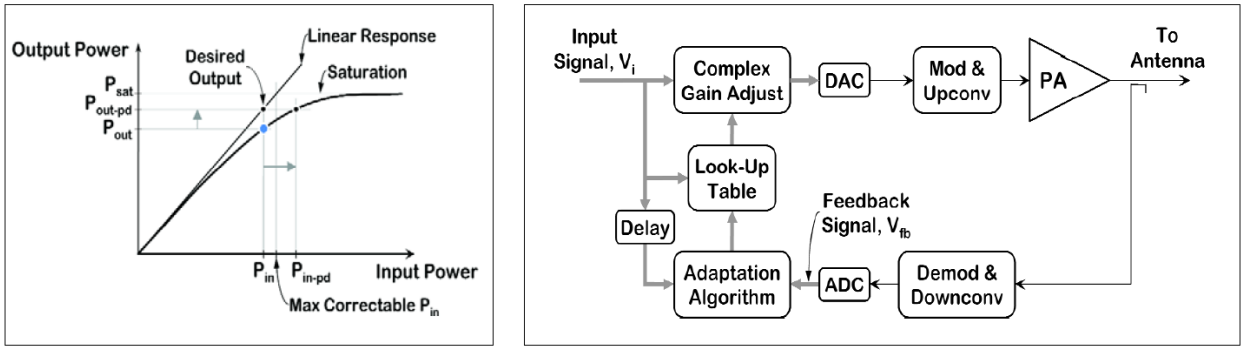

Figure 3.1: Maximum Correctable Power (right) and Predistortion Feedback Circuit (left)

responses. For this model, we use a lookup table-based algorithm, shown in Figure 3.1. In this method, the output of the amplifier is coupled off using a directional coupler with about 20 dB of coupling loss, causing negligible degradation of the signal output. However, since we are at the output of the amplifier, the signal has already been modulated and converted to analog. To bypass the mixer products and filter requirements, the signal needed to be demodulated and converted to digital. There, we can use our algorithm to analyze the coefficients of the nonlinear products at a given power level. We can then apply a look up table and effectively pre-distort the signal such that the new signal into the amplifier contains the inverse of the nonlinear coefficients produced by the amplifier, creating a more linear signal.

This method does require a feedback path, which for our implementation, using an Analog Devices AD9371 Transceiver, does not exist, as shown in Figure 3.2 [23]. However, there is a work around: we can manually retrieve the output signal at a range of power levels, and externally apply the adaption algorithm to populate our look up table values as a function of input power. The measurement device used to extract nonlinear data, the spectrum analyzer, will have its own nonlinearities that must be taken into consideration. We will go into this in a later section.

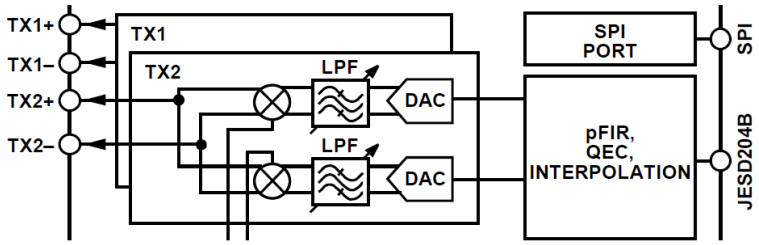

Figure 3.2: Output Transmit Architecture of AD9371

3.3 Predistortion Algorithm: Analysis

Since we lack a feedback path, we can also pursue an alternative feed-forward approach; where we can measure the nonlinear behavior and apply the coefficients via FPGA without the need for feedback. In this approach, we make use of a vector modulator, which can split a signal into an in-phase and quadrature component, 90 degrees apart in phase [24], with the additional feature of amplitude control, using a mixture of pin diodes and hybrid couplers. This technique is a novel implementation of predistortion in mixers, whereas previously predistortion has only been explored in amplifiers [25]. However, it is analogous to the predistortion methods implemented in amplifiers, which suit our setup perfectly as the AD9371 does not make use of an output amplifier; rather it connects an active mixer directly to the output [23]. In this feedforward approach, the input signal is split into a distortion path and a delay path. The distortion path is further split into another delay path and the IMD path. IMD error signals are then generated and fed through the vector modulator before being recombined with a delay path signal to rectify the intermodulation products (Figure 3.3).

The IMD generator in the figure is a simple nonlinear amplifier that can generate intermodulation products. This is for theoretical purposes, as we do not intend on adding another amplifier into our signal chain. Given that this is a simple nonlinear amplifier, similar to equation 3.1, we can model the output of the IMD generator as a Taylor series. Using the same two-tone example from before, $x = Acos(\omega_1 t) + Acos(\omega_2 t)$, the Taylor series then

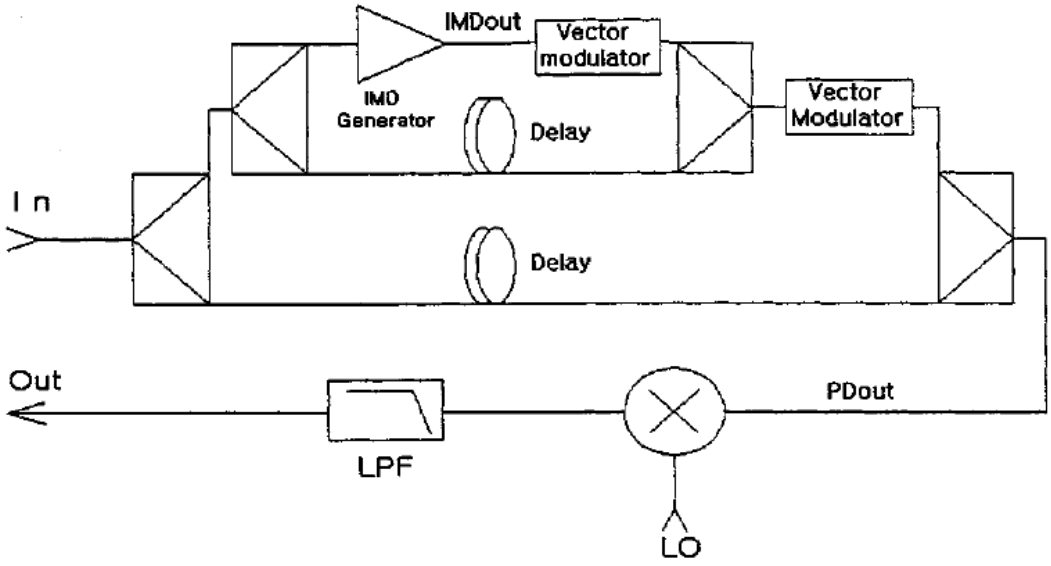

Figure 3.3: Example of a Feedforward Predistortion Circuit with a Mixer

becomes:

$$IMD_{out} = a_0 + a_1 A[cos(\omega_1 t) + cos(\omega_2 t)] + \frac{3}{4} a_3 A^3[cos(2\omega_1 t - \omega_2 t) + cos(2\omega_2 t - \omega_1 t)] + \cdots \quad (3.5)$$

We can then apply the vector modulator, which has amplitude control, to remove the fundamental signal. Applying a phase delay and another vector modulator, we can expect some phase delay and amplitude change, and express the output right before the mixer, PD_{out} as follows:

$$PD_{out} = b_0 + b_1[cos(\omega_1 t) + cos(\omega_2 t)] + b_2[cos(2\omega_1 t - \omega_2 t + \alpha) + cos(2\omega_2 t - \omega_1 t + \alpha)] + \cdots \quad (3.6)$$

Here we are expressing b_n as some amplitude variable and α as some phase delay, both of which can be controlled via the vector modulator. With a general mixer having an output described by:

$$M_{out} = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n \quad (3.7)$$

where x is the product of the LO signal $Lcos(\omega_L t)$ and the original signal x, and the general

nonlinear coefficients of the mixer are c_n, then the output of the overall circuit, after selective low pass filtering, and with some phase delay for the n^{th} harmonic being ϕ_n, becomes:

$$\begin{aligned} IF_{out} &= b_1 c_2 L cos(\omega_1 t - \omega_L t + \phi_1) \\ &+ b_1 c_2 L cos(\omega_2 t - \omega_L t + \phi_1) \\ &+ b_2 c_2 L cos(2\omega_1 t - \omega_2 t - \omega_L t + \phi_2) \\ &+ b_2 c_2 L cos(2\omega_2 t - \omega_1 t - \omega_L t + \phi_2) \\ &+ \frac{3}{4} c_4 {b_1}^3 L cos(2\omega_1 t - \omega_2 t - \omega_L t + \alpha + \phi_3) \\ &+ \frac{3}{4} c_4 {b_1}^3 L cos(2\omega_2 t - \omega_1 t - \omega_L t + \alpha + \phi_3) + \cdots \end{aligned} \tag{3.8}$$

Since we are specifically targeting the 3rd intermodulation product, the most important of which was shown in the sections above to be $2\omega_a - \omega_b$ and $2\omega_b - \omega_a$, we can therefore isolate those products, and conclude that we can effectively cancel it when:

$$\begin{aligned} b_1 c_2 &= \frac{3}{4} c_4 {b_1}^3 \\ \phi_2 &= \alpha + \phi_3 + 180^o \end{aligned} \tag{3.9}$$

The former equation is the amplitude condition, and the latter is the phase condition. Note that the mixer amplitude L has been cancelled, so that it is independent of our calculations. In the same way, it can be shown that the higher order intermodulation products can be cancelled or mitigated in the same way. As a note: for this to work, the output mixer needs to be thoroughly characterized such that all c_n and ϕ_n coefficients are known to a precise degree. In addition to that, the b_n and α coefficients are controlled by the vector modulator.

Since we do not have the real estate to add the necessary circuitry, these coefficients can be manually calculated for our specific set up, and a look up table can be generated according to Figure 3.1, with the algorithm being applied at the FPGA level according to the Tick setup. This is all being done with the understanding that these procedures can be avoided with a proper feedback path.

CHAPTER 4

MAC Layer: Reducing Tail Latency

4.1 Critical Metrics of Tail Latency

The second half of our design involves reducing the latency of our transmission. If we were to target mobile AR/VR applications, a high data rate at an ultra-low latency is critical. Mobile augmented reality (AR) latency requirements have been shown to be on the order of 1-3ms [26]. Previous generation 802.11n networks specified speeds up to 600Mbps for a single AP while current generation 802.11ac networks offer similar speeds at close distances and up to 7Gbps for an 8x8 MIMO setting [5]. Tests of the stock protocol 802.11n under average speed conditions have shown latency delays as low as 5ms [2]. This would suggest that the 802.11ac stock protocols we are looking at is somewhat close in meeting these requirements. However, the overall 802.11 Wi-Fi standard does not specify a centralized scheduler, which means there is no guarantee of throughput or latency for any client [12]. In addition, it has been shown that a higher data rate does not always imply lower overall latency [3]. In the past, the assumption was that a higher data rate meant lower latency delays. The faster the transmission rate, the quicker the information gets transmitted and received, and the less time it takes for an overall transmission. However, further analysis and experimentation shows that this is not the case. Rather, while a higher data rate may lower *average* latency, it does not have the same effect on the tail end of the delay distribution, as packets landing in that range will experience higher overall delays due to increased queuing and service times. Instead, the latency, and specifically the tail latency, is dominated by the retransmission rate. While some time can be saved by transmitting faster, the rate of error

inevitably goes up, and a small percentage of packets need to be sent again. Due to the lack of a central scheduler, or rather the lack of priority assigned to retransmitted packets, these retransmitted packets may or may not get stuck in the queue behind higher priority packets. While they may as whole take longer to be successfully transmitted, the fact that the majority of packets are transmitted faster lowers the average latency and hides this issue. This becomes an issue in high latency sensitive applications like ours, contributing to unacceptable lag times and quality of service degradation. Based on this, it is clear that the standard 802.11 rate adaptation algorithm prioritizes high goodput over latency, creating a prime opportunity to target that area in lowering our latency.

In terms of tail latency reduction, previous work has been done in the form of the Low Latency Rate Adaptation algorithm (LLRA) [3] and the MIMO Rate Adaptation (MiRA) [27], implemented on 802.11n networks. These will be our primary reference moving forward. In brief, LLRA targets the tail latency in 802.11n networks, while MiRA introduces the novel rate adaptation and lightweight probing mechanism to reduce packet loss, while still maintaining an implicit probing method. We build on this algorithm by implementing in 802.11ac networks. The primary difference between the 802.11ac and 802.11n is the sophistication of the MIMO modes, increasing from 4x4 to 8x8, and allowing for MU-MIMO. In addition, the introduction of beamforming in 802.11ac allow for more efficient transmissions.

4.2 Low Latency Rate Adaptation Algorithm: Introduction

With a clear path forward, we can now analyze the rate adaptation algorithm for our specific use. Given that the 802.11 standard does not fully specify any rate adaptation, we can therefore craft our algorithm to fit within the standard and satisfy our goal of a customizable API without large scale changes and incompatibilities.

Firstly, we want to know what the best rate given a specific channel condition is. As stated before, the fastest data rate may not always yield the lowest latency. Therefore, we

implement the concept of rate control to answer this question. While 802.11 standards do not regulate latency, some rate control is applied in the form of a lookup table at the PHY layer to maximize throughput. With the advent of MIMO and beamforming technology in 802.11ac, we now have a diverse variety of rates vs spatial streams to choose from. It has been shown that there is a non-negligible, non-monotonic relation between the rate option and error rates when considering all rates in both single stream and double stream modes [27]. As we know that error rates are directly related to increases in latency [3], we can therefore apply a rate control algorithm to find the rate that yields the lowest latency.

Secondly, we can use the concept of frame aggregation. Building on the bulk acknowledgement mechanisms from the past [1], we can pack multiple packets into an aggregated frame and send all at once to avoid the delays associated with the individual acknowledgements of each packet. The maximum size of the aggregated frame is bound by the round-trip time (RTT) of the frame, and therefore, the transmission time. With an increase in data rate, we can increase our frame size.

Finally, we address the frame failure possibility with retransmission dispatching. Once a transmission has so many errors that recovery is impossible, a retransmission request is made in the form of withholding an acknowledgement. The transmit side expects an acknowledgement for its transmission, and when one is not received, a retransmission is made with two types; *hardware retry* and *software reschedule*, as seen in Figure 4.1 [3]. Initially, packets are placed into the software queue as it arrives to the MAC layer on the transmit side. It is not placed into the hardware queue until all the priority messages have been scheduled and transmitted. Once that is done, the packets are combined to form an aggregate frame, then transmitted all at once. The hold times experienced by these packets in the software queue contribute to the service time portion of our delay. A block acknowledgement (Block-ACK) is sent from the receiver. If the ACK is received for only a portion of the frame, then the packets received in error get placed into the software queue, where they wait for higher priority packets to be sent first. However, if no ACK is received,

which means the entire frame was received in error, then hardware retry is implemented, where the entire frame stays in the hardware queue for re-transmission. This means it gets the maximum priority and immediate re-transmission.

As observed in LLRA [3], and restated previously, the retransmission is critical to the tail latency. A secondary observation followed from that, in that under low loss conditions, initial queuing delay is negligible, so that the fastest rate setting among those requiring the minimum retransmissions is preferred. Additionally, frame aggregation aggravates the impact of retransmission because by default in 802.11 standards, a software reschedule is used. As shown in Figure 4.1, the software queue will incur a larger amount of delay than the hardware queue, since the default priority setting is not customized to ensure these packets leave as quickly as possible and reduce the latency. This is further exacerbated in MIMO conditions, where channel congestion causes longer hardware queues as well. This is where we can implement retransmission dispatching to mitigate these issues. For all packets determined to be part of the long tail distribution of latency that we are looking to reduce, any retransmitted packets within this group receive the highest priority, so that they may leave the software queue as soon as possible. This method minimizes the latency while making sure that the other packets do not incur additional wait times as a result.

Therefore, we can combine these 3 components for our rate adaptation algorithm, where we use the rate control to search for the lowest latency rate among the options we have available. Rate control then balances between these two quantities such that under low queue situations, it selects the highest rate with low PER, rather than the highest goodput rate which may incur higher PER, and under high traffic situations it selects the faster rate with higher PER, as it drains the queue faster. The frame aggregation scheduling is then used to reduce the queuing delays, while the retransmission dispatching handles the service times by prioritizing retransmitted frames to avoid long hold times.

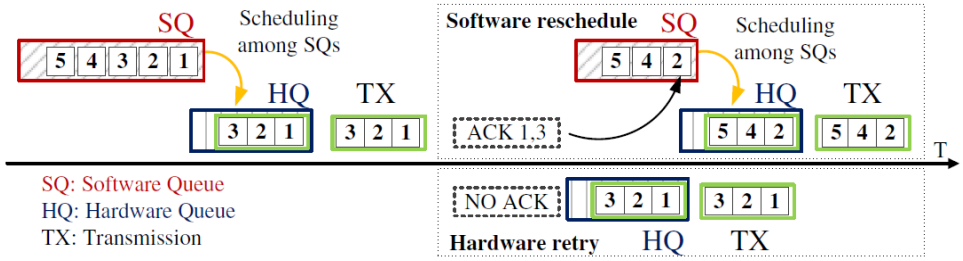

Figure 4.1: Hardware Retry and Software Reschedule flow

4.3 Low Latency Rate Adaptation Algorithm: Analysis

To implement the algorithm, we need to follow a few rules which describe the behavior of the latency, as observed in LLRA [3].

- **Rule 1:** When queuing delay does not vary with rate settings, the service time thus makes a difference. The fastest rate setting among those requiring the least number of retransmissions is preferred. Otherwise, we should consider the initial queuing delay and service time together for each rate setting.
- **Rule 2:** Aggressive aggregation is applied to the first group of packets, but not together with the second group.
- **Rule 3:** To offset the tail delay of the first group, prioritized reschedule is applied to the group's packets that require software reschedule.

We can thus customize the rate control, frame aggregation, and retransmission dispatching to adhere to these rules. In addition, these rules make obvious the need for a quick evaluation algorithm. Given that we will compare different rates' latency against each other, we need to define, then measure, the parameters used to calculate the latency. This manifests as a lightweight probing mechanism that measures a rate's PER and calculates the latency accordingly [3] [27]. These components will be detailed below.

4.3.1 Low Latency Rate Control

To implement the rate control, we need to search for the most suitable rate. The search will be triggered by both time-driven and event-driven approaches. For the event-driven approach, the search is triggered once the current rate gets worse. There will be an evaluation of PER and retransmission rate, and if that degrades, the algorithm will begin searching for a new rate. However, if the performance does not worsen, the time-driven approach will trigger after a period of time, as it searches for a new rate and refreshes stale information. Iterating through all available rates is impractical, as given the amount of MIMO modes and data rates possible, the channel will have changed by the time a suitable rate is found. Therefore, we apply a set of pruning rules to automatically rule out certain rates, derived from [3]. These are used to mathematically eliminate rate settings as compared to the current rate, R_{best}, and compare the actual latency between two candidate rate settings. The latency is given as $D_{est} = T_q[0] + T_{srv}$, where $T_q[0]$ is the initial queuing delay, dependent on the preceding frame, N_{pre}, and T_{srv} is the service time, dependent on retransmission time, N_{rt}, and the transmission time, t_{tx}.

Theorem 1: Given R_{best} with $N_{rt} = 0$, any low rate R such that $R < R_{best}$ is pruned since it cannot perform better than R_{best} (i.e. $D_{est}(R) \geq D_{est}(R_{best})$)

Theorem 2: Given a rate R which $R > R_{best}$ and $D_{est}(R) > D_{est}(R_{best})$, the higher rate $R^{'}$ (i.e. $R^{'} > R$) in the same MIMO mode is pruned since it cannot perform better than $D_{est}(R_{best})$

Theorem 3: Given two adjacent rates R_{low} and R_{high}; if they have identical N_{pre}, the one with the smaller N_{rt} is better. If N_{rt} is identical, then they need to be compared based on a measured D_{est}.

This is put into practice as an example shown in Figure 4.2 [3]. SS, DS, and TS correspond to single stream, double stream, and triple stream modes, respectively, and the number before that refers to the data rate indexed by the MCS table. Starting at 162DS, a search is triggered when the latency increases at that rate. Assuming N_{rt} changes from 0 to 1, and

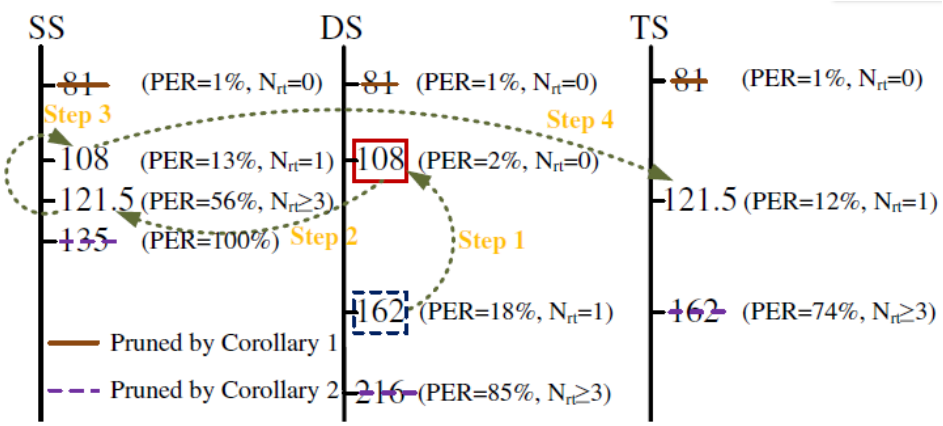

Figure 4.2: Example Flow Diagram of the Search Algorithm

$N_{pre} = 0$, we can then search within the same MIMO mode. It searches downward for a rate with less delay (108DS), and since it has an N_{rt} of 0, switches to it. The higher rate settings can be pruned, as can the lower ones, due to Theorem 2 and 1 respectively. The algorithm then concludes that this is the best rate and begins probing the next higher rate in the single stream mode (121.55SS). Since it is worse than 108DS due to $N_{rt} \geq 3$, the higher rates are eliminated. It stops at 108SS, which is worse than the current 108DS (All lower rates in the single stream are eliminated). Finally, it looks at the next higher rate in the triple stream, 121.5TS. As both the higher and lower rates are worse, the search concludes at 108DS.

4.3.2 Frame Aggregation Scheduling and Retransmission Dispatching

Frame aggregation, meant to target the tail latency, can be approached as how much of the tail we are trying to reduce. By dividing the packets based on retransmission count, we can apply frame aggregation to those packets with delays higher than the expected retransmission count at the α^{th} percentile, and therefore reduce the latency of the packets above that percentile. As previously discussed, retransmission count is a direct function of the PER, so the PER threshold can be calculated according to the target percentile of packets being split.

For a given PER at a specific rate setting, we can estimate the retransmission count

as a function of the a^{th} percentile. Assuming that packet errors for these given conditions are independent, the probability of a successful transmission after being preceded by i failed packets is given by the geometric distribution as $p^i(1-p)$. This also represents the percentage of successful frames after $i+1$ transmissions. By setting our split to the α^{th} percentile, we can find N_{rt} as the smallest integer that satisfies $\sum_{i=0}^{N_{rt}} (1-p)p^i = \alpha$. We can then show PER threshold vs. retransmission count, as shown in Table 4.1 [3].

N_{rt}	1	2	3
$\alpha = 90^{th}$	10.0%	31.6%	46.4%
$\alpha = 95^{th}$	5.0%	22.3%	36.8%

Table 4.1: PER Threshold vs retransmission count

We can then calculate N_{rt} at the targeted α^{th} percentile and split the packets into above and below that number. In other words, for a given PER and rate, we expect a percent of the packets to be retransmitted less than N_{rt} times. The packets with a retransmission count at or below that N_{rt} number are subject to aggressive frame aggregation scheduling; packing as many packets into the frame as possible, with the highest priority to avoid long wait times at the software queue. This is done so that the packets that need to be retransmitted more than N_{rt} times are not stuck behind a long queue. These packets do not get assigned the highest packets. The number of packets per frame is a function of the rate, which is determined by our rate control algorithm. Once all the packets in first group have been serviced, aggressive frame aggregation is applied to the second group. This way, we can reduce the tail delay of the first group of long latency packets without increasing the initial delay, $T_q[0]$, of the second group. More specifically, $T_q[0]$ can now be defined as $T_q[0] = (N_{pre}+1) \times T_{sw\_int} + T_{HQ}$. T_{sw_int} represents how often a frame is moved from the software queue to the hardware queue, T_{HQ} is the amount of time spent in the hardware queue, and N_{pre} is the number of preceding frames to a given packet, which can be estimated based on the average number of simultaneous packets in the queue and the rate's maximum aggregation size.

4.3.3 Light-weight probing

Similar to the search algorithm, we desire a method of quickly determining the PER of a given rate. While there are two methods of determining packet error rate, probing based and SNR based, most platforms do not support the latter, so we stick with the former. Following pre-existing work done on lightweight probing [3] [27], we see that the probing mechanism consists of 2 components for eliminating probing packets for invalid rates:

- **Slow-Start Pruning:** It detects an inapplicable rate with few packets. To probe a rate, it starts from the frame with one packet size, and then exponentially increases the frame size, stopping when either the collected results are sufficient or the packet loss count has verified the inapplicability of the rate. A rate is inapplicable when its PER exceeds certain threshold associated with the retransmission count, such that it can perform no better than the current best rate, according to Table 4.1
- **Association Rule-Based Pruning:** uses correlations among rates to infer a rate's inapplicability from other rates under similar channel conditions. The underlying premise is that each rate setting behaves similarly among time-varying samples under the same channel condition, so rate settings can be correlated with each other in their performance. An example of this is shown in Table 4.2 [3].

In addition, this mechanism uses success inheritance, which inherits successful probing data from a higher rate.

History Probing	121.5 SS	162 DS	121.5 TS
1	1	1	1
2	1	1	0
3	0	0	0
4	1	0	1
5	1	1	1

Table 4.2: Probing Transaction Example

CHAPTER 5

Implementation and Evaluation

5.1 Introduction to Our Test Bed

We evaluate our algorithm with a preexisting SDR system, Tick. Tick is an SDR (software defined radio) system that provides programmability and ensures low latency at both PHY and MAC [28]. It has been programmed to support 802.11a/g/n/ac protocols. For our purposes, it is a self-contained transceiver system that allows for point-to-point communication with no additional equipment required besides a host PC. For our setup, we decide to use two Tick modules to model our system as a two-point network, for simplicity. In practical larger scale operations, we anticipate multiple modules communicating with a single access point. With just the host PC, Tick allows us to test our low latency software without introducing any additional latency impact in our targeted section. The FPGA provides a medium to support all MAC layer operations required to implement our low latency algorithm as well as the digital operations needed for the linearization portion. It then interfaces with the RF front-end, which operates the transmit and receive hardware.

5.1.1 AD9371 Transceiver

The RF front-end is a radio card supporting the AD9371 transceiver chip. The AD9371 itself is a dual channel transceiver IC with integrated synthesizers and digital signal processing [23]. Although not explicitly stated, it can support standard 802.11ac communications by virtue of its operating frequency range and narrowband flatness in the 2.4GHz and 5GHz bands.

Internally, the output stage of the RF chain is an active mixer, with the assumption that it would be connected to an external power amp lifer for range purposes. The overall lineup consists of quadrature error correction and programmable digital filters, which eliminates the need for them in the digital baseband, bypassing an element of concern when considering digital predistortion. Externally, it has multiple transmit and receive channel, capable of both FDD and TDD applications, and able to support SISO and MIMO modes as well.

It has four high speed serial interface links each for the transmit and receive chains, JESD204B. These are multi-gigabit serial data links to connect with an FPGA for customization purposes. Our linearization coefficients will be generated at the FPGA, then added to the digital baseband signal before passing through the JESD204B lanes to the AD9371, which has additional programmable FIR filters for interpolation and sampling artifact removal capabilities at the digital and analog domains.

The receiver side consists of 2 independent receiver channels, each of which is a direct conversion system with a digitally adjustable attenuator. The incoming signal gets self-mixed to the baseband, before passing to the ADC which has adjustable sample rates and FIR filters with additional decimation capabilities.

5.1.2 KC705 FPGA

The FPGA we use is an KC705 evaluation board for the Xilinx Kintex 7 series FPGA. It can natively interface with the host PC via JTAG and PCIe, but we have connected an external USB3.0 module via one of the two FMC (FPGA Mezzanine Card) interfaces, the other connecting to the AD9371 Transceiver evaluation board. The full configuration can be seen in Figure 5.1. Both interactions are supported internally via the JTAG logic module, which also allows for software configurations using the Xilinx software tool.

Tick uses this exact setup to experiment with several software and hardware techniques to reduce latency, all implemented within the embedded processor of the FPGA. In summary,

it utilizes multi clock domain pipelining to speed up PHY layer data-flow processing like message passing, and implements an accelerator-rich environment at the MAC to direct control flow and reduce latency.

The embedded microprocessor used is MicroBlaze, a soft microprocessor core implemented entirely within the general purpose memory of the KC705. Given that most of Tick is out-of-scope, we will omit descriptions of the rest of the system level features and modules critical for execution, like the clock generation and the memory modules.

Tick itself is programmed in Ubuntu, with most of the code implemented in Verilog, and a small portion in C for the embedded processor logic. We aim to code the linearization portion of our solution in the form of modifications to the digital baseband implemented in Verilog, and the rate adaptation portion in C.

5.2 Implementation in Tick

Our intent was to test our algorithm over the preexisting 802.11ac protocol currently running. The output port of the AD9371 radio card was intended for an output power amplifier. We instead connected an omnidirectional antenna to each transmit and receive port, knowing that the max range would be limited, but instead looking to see how much improvement we can achieve over the stock protocol. Performance metrics here would be maximum distance while still maintaining minimum coherent transmission, as well as max distance with respect to MCS CQI (channel quality indicator).

In terms of latency, we look to plot the cumulative distribution functions comparatively between a standard high goodput rate setting in 802.11ac and the corresponding low latency adaptation rate that we are implementing.

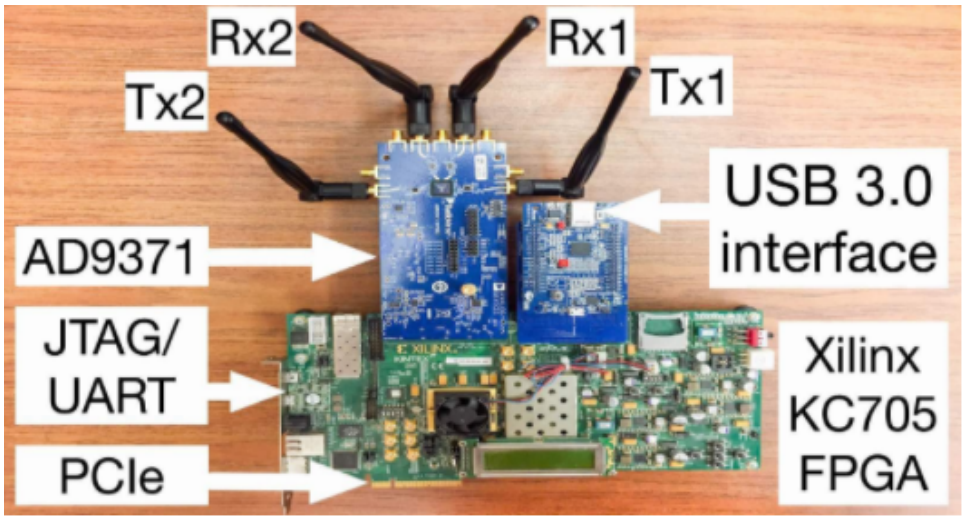

Figure 5.1: Tick Setup

5.2.1 Transceiver Device Characterization

In order to properly implement the predistortion algorithm, we needed to thoroughly define the mixer nonlinear coefficients and phase delays. This would require accurate characterization of both the ACLR (adjacent channel leakage ratio) and the intermodulation products. While ACLR can be measured through a modulated signal on a spectrum analyzer, the intermodulation products would require a separate element programmed in Verilog. Summarily, instead of transmitting normally via the 802.11 protocol, we need to send unmodulated test tones, preferably two, at a controllable power level. The element itself would have needed to be able to cycle through many different power levels, frequencies, and tone spacing. We are looking specifically for the difference in power of the intermodulation products, *relative* to the desired tones, measured in dBc, or dB below carrier. This dBc measurement directly correlates to the ratios of the magnitudes, or A_i/A_1, $\forall\ i > 1$, as described by equations 3.1 and 3.7. In addition, AM/PM tests can be performed to determine the phase delay. We can then determine all our coefficients and create our lookup table. However, due to COVID-19 restrictions, lab access was forbidden, and implementation of this section was not possible.

5.2.2 Algorithm implementation

Our code in implemented in C. We split the algorithm into two parts, a probing algorithm to improve upon the implicit probing that comes standard with the 802.11 protocol, and the rate control that determines the best latency rate. The rate control algorithm will search across all available rates for the candidate rate with the next lowest latency according to rules defined in section 4.3.1, using channel information determined by a probing mechanism, as described in section 4.3.3

5.2.2.1 Rate Control

While in lab testing and access to equipment was again unavailable due to COVID-19, we were able to construct the logic that forms the basis of the search algorithm. The search algorithm differentiates MIMO modes in that it will search through a single mode (SS, DS, TS, etc.) before looking across modes. By Theorem 3, it compares adjacent rates, moving upward in rate setting until the highest rate is reached or everything higher is eliminated by Theorem 2. Then it moves downward until the lowest rate setting is reached, or alternatively, the highest one with $N_{rt} = 0$. After that, Theorem 1 dictates that it can start searching other modes with $N_{rt} = 0$, continuing the process until all rates have been evaluated or mathematically eliminated.

Based on this, we would need to maintain several matrices indexing the PER of different rates and MIMO modes, along with the critical parameters like N_{rt} and N_{pre} needed to evaluate latency, which has been defined previously as $D_{est} = T_q[0] + T_{srv}$. These parameters would be constantly refreshed with the probing function or an inquiry within the FPGA measurements. With $T_q[0] = (N_{pre} + 1) \times T_{sw\_int} + T_{HQ}$ and $T_{srv} = T_{tx} + N_{rt} \times (T_{HQ} + T_{tx})$, we can solve for the latency. T_{tx}, which is the sum of transmission through air and the MAC overhead, and T_{HQ}, which is the hardware queuing delay, and T_{sw_int} can all be measured and estimated at the FPGA level. N_{pre} can be calculated in accordance with [3]. As detailed

in previous sections, N_{rt} can be calculated as a function of the a^{th} percentile using the PER for the channel. The PER of a given rate is further analyzed using the probing mechanism, described below. During implementation, we will assume the mechanism has already given us the proper N_{rt} values, and all missing variables have been measured and defined. A snippet is detailed in Figure 5.2.

5.2.2.2 Probing

Our proposed probing algorithm is built on the same principle as the implicit probing implemented in standard rate adaptation algorithms. To determine the PER and thus the channel behavior, we actively measure the incoming packets and keep track of how many packets need retransmission. In MiRA, this was done with a single A-MPDU (aggregate MAC Protocol Data Unit), as each aggregate frame contains enough packets to make an accurate estimation [27]. For us, since the number of packets needed to determine PER and thus N_{rt} is dependent on a, we remove the frame limit and probe until we have a determination or until we collect enough errors to invalidate the rate. For example, if we were to set our $a = 95$, then based on Table 4.1, our PER threshold for an N_{rt} of 1 is 5%. So, in Figure 4.2, starting at 108DS with an N_{rt} of 0, the latency rate control decides to probe 162DS. 162DS is then probed, until its PER of 85% proves it is not a better rate setting.

Here, the probing mechanism will actively measure the error percentage, or PER, denoted as p in the expression $\sum_{i=0}^{N_{rt}} (1-p)p^i = \alpha$. it will then return N_{rt} to the rate control algorithm for determination.

Additionally, it will need to dynamically calculate N_{pre} and D_{est} according to section 4.3.2. To save both operating overhead and memory, we only probe as required per the rules we set. With 10 different data rates each for different channel widths and configurations across 4 MIMO modes, continuously updating for N_{rt} and N_{pre} for every element is inefficient.

```
int theorem3()
{
    int num_rates = 10; int num_modes = 4; //10 MCS rates, 4 MIMO modes
    int chan_nrt[num_rates][num_modes]; //indexes Nrt, 10x4
    int check_channel = refresh_timer();
    int channel_change =1;
    // channel_change is a function indicating whether Nrt has channged
    int cur_rate = 5; cur_mode = 2
    if ((check_channel == 1)||(channel_change==1))
    {
    //checks PER of the rate above/below it, updates Nrt in channel_nrt matrix
        probefunc(cur_rate+1, cur_mode);
        probefunc(cur_rate-1, cur_mode);
        //probefunc is the robing function
        int npre_up = calc_npre(cur_rate+1, cur_mode);
        //determine npre of both rates
        int npre_down = calc_npre(cur_rate-1, cur_mode);
        if (npre_up==npre_down) //if both have same npre, execute following
        {
            if(chan_nrt[cur_rate+1][cur_mode]<=chan_nrt[cur_rate-1][cur_mode])
            {
                cur_rate = cur_rate +1; //change to the higher rate
            }
            else
            {
                cur_rate = cur_rate -1;
            }
            update_rate(cur_rate); //switches the current rate to the new rate
        else //need to calculate Dest
        {
            //assume all parameters known and stored, like Tq[0] and Tsrv
            int dest_up=calc_dest(npre_up, chan_nrt[cur_rate+1][cur_mode]);
            int dest_down=calc_dest(npre_down, chan_nrt[cur_rate-1][cur_mode]);
            //calc_dest calculates Dest based on the given params
            if (dest_down<dest_up)
            {
                curr_rate = cur_rate-1;
            }
            else
            {
                curr_rate = cur_rate+1;
            }
            update_rate(cur_rate);
        }
    }
    return;}
```

Figure 5.2: Search Algorithm Snippet for Theorem 3

5.3 Evaluating the Algorithm

While lacking in live parameters, we were able to insert dummy variables in their place. By creating a placeholder matrix with estimated values for N_{rt}, N_{pre}, $T_q[0]$, and such, we were able to confirm our logic, as shown in Figure 5.2 with theorem 3. As an example, we started in the middle, with a rate index of 5 and mode index of 2 corresponding to 351DS in 802.11ac. The algorithm then probed the 2 neighboring rates for their PER, and calculated the N_{rt}, N_{pre}, and other associated parameters accordingly, before making a determination.

CHAPTER 6

Conclusion

Our analysis has shown us that the nonlinear behavior of the amplifier does have a significant influence on the spectral performance. While our work was limited to narrowband considerations, we recognize that in wideband or broadband systems a more complex model needs to be considered, especially when it comes to accounting for memory effects. Nonetheless, we have seen that the linearizer has a large role in the signal chain, whether it be intermodulation mitigation or efficiency-based applications like dynamic load modulation.

In terms of latency, we have shown that most systems prioritize goodput over latency when it comes to rate adaptation, and that these algorithms are insufficient when it comes to long tail latency. Furthermore, we note that the ambiguity in the standard 802.11 opens the door for possibly cross-layer improvements. For example, we can see an approach of improving the latency on both PHY and MAC simultaneously, as current methods are more or less confined to independent single layer solutions.

In conclusion, our work has shown that there are some definitive areas of improvement for existing 802.11 systems. While most home networks in urban environments are incentivized to maintain a low range to avoid interference with neighboring networks, we can see that a customizable software-based API that can control range is very useful. This especially rings true in the latency domain, as latency sensitive operations do not consist of the bulk of a user's internet traffic. Therefore, a low latency mode that can be switched on would be very useful.

REFERENCES

[1] Rabin K Patra, Sergiu Nedevschi, Sonesh Surana, Anmol Sheth, Lakshminarayanan Subramanian, and Eric A Brewer, "Wildnet: Design and implementation of high performance wifi based long distance networks," *USENIX Symposium on Networked Systems Design & Implementation*, vol. 4, pp. 87–100, 2007.

[2] David Newman, "802.11n gear 10 times faster than current Wi-Fi offerings," *Network World*, Oct 2008.

[3] C. Li, C. Peng, S. Lu, X. Wang, and R. Chandra, "Latency-aware rate adaptation in 802.11n home networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 1293–1301.

[4] Stephen B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall, Upper Saddle River, NJ, USA, 1995.

[5] Matthew S Gast, *802.11 ac: a survival guide: Wi-Fi at gigabit and beyond*, " O'Reilly Media, Inc.", 2013.

[6] Reina Riemann and Keith Winstein, "Improving 802.11 range with forward error correction," *Technical Report, MIT-C, SAIL-TR-2005-011*, 12 2005.

[7] S. Aust, R. V. Prasad, and I. G. M. M. Niemegeers, "Outdoor long-range wlans: A lesson for ieee 802.11ah," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1761–1775, 2015.

[8] The Consultative Committee for Space Data Systems, "Bandwidth-efficient modulations," *CCSDS Green Book*, vol. 2, pp. Chapter 2, Oct 2009, [Online; accessed 21. Nov. 2020].

[9] R. Wei and X. Wang, "Differential 16-qam and 16-apsk for uplink massive mimo systems," *IEEE Wireless Communications Letters*, vol. 7, no. 2, pp. 170–173, 2018.

[10] Inter range Instrumentation Group, "Transmitter and receiver systems," *IRIG Telemetry Standard*, p. Chapter 2, Jul 2019, [Online; accessed 21. Nov. 2020].

[11] Marco Baldi, Franco Chiaraluce, Antonio de Angelis, Rossano Marchesani, and Sebastiano Schillaci, "A comparison between APSK and QAM in wireless tactical scenarios for land mobile systems," *J. Wireless Com. Network.*, vol. 2012, no. 1, pp. 1–14, Dec 2012.

[12] Ilya Grigorik, "Performance of Wireless Networks: WiFi - High Performance Browser Networking (O'Reilly)," *High Performance Browser Networking*, Apr 2016.

[13] An Chan, Henrik Lundgren, and Theodoros Salonidis, "Video-aware rate adaptation for mimo wlans," in *2011 19th IEEE International Conference on Network Protocols*. IEEE, 2011, pp. 321–330.

[14] Starsky HY Wong, Hao Yang, Songwu Lu, and Vaduvur Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*, 2006, pp. 146–157.

[15] Kaixin Sui, Mengyu Zhou, Dapeng Liu, Minghua Ma, Dan Pei, Youjian Zhao, Zimu Li, and Thomas Moscibroda, "Characterizing and improving wifi latency in large-scale operational networks," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 2016, pp. 347–360.

[16] Kuo-Chang Chan, "IP3 and Intermodulation Guide | Maxim Integrated," March 2013, [Online; accessed 3. Nov. 2020].

[17] A. Loke and F. Ali, "Direct conversion radio for digital mobile phones-design issues, status, and trends," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 11, pp. 2422–2435, 2002.

[18] A. Lozowski and B. Galwas, "Homodyne frequency demodulator for phase noise measurement systems," in *1995 25th European Microwave Conference*, 1995, vol. 2, pp. 1251–1253.

[19] Frederick H Raab, "High-efficiency linear amplification by dynamic load modulation," in *IEEE MTT-S International Microwave Symposium Digest, 2003*. IEEE, 2003, vol. 3, pp. 1717–1720.

[20] Wan-jong Kim, S. Stapleton, and H. Kim, "Linearizing Power Amplifiers Using Digital Predistortion, EDA Tools and Test Hardware," *High Frequency Electronics*, 2004.

[21] Maxim Integrated, "RF Predistortion vs. Digital Predistortion - Maxim Integrated," March 2013, [Online; accessed 3. Nov. 2020].

[22] Keysight, "Digital Pre-Distortion (DPD) Concept," Sep 2020, [Online; accessed 3. Nov. 2020].

[23] "AD9371 Datasheet and Product Info | Analog Devices," Mar 2017, [Online; accessed 20. Nov. 2020].

[24] Youngwook Kim, Youngsik Kim, and Soonghak Lee, "Linearized mixer using predistortion technique," *IEEE Microwave and Wireless Components Letters*, vol. 12, no. 6, pp. 204–205, 2002.

[25] JS Kenney and A Leke, "Design considerations for multicarrier cdma base station power amplifiers," *Microwave Journal*, vol. 42, no. 2, pp. 76–83, 1999.

[26] Wouter Pasman and Frederik W Jansen, "Latency layered rendering for mobile augmented reality," in *SYMPOSIUM ON INFORMATION THEORY IN THE BENELUX*. Werkgemeenschap voor Informatie-en Communicatietheorie; 1998, 2000, pp. 45–54.

[27] Ioannis Pefkianakis, Yun Hu, Starsky H.Y. Wong, Hao Yang, and Songwu Lu, "Mimo rate adaptation in 802.11n wireless networks," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, New York, NY, USA, 2010, MobiCom '10, p. 257–268, Association for Computing Machinery.

[28] Haoyang Wu, Tao Wang, Zengwen Yuan, Chunyi Peng, Zhiwei Li, Zhaowei Tan, Boyan Ding, Xiaoguang Li, Yuanjie Li, Jun Liu, et al., "The tick programmable low-latency sdr system," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017, pp. 101–113.