Comb Decoding towards Collision-Free WiFi

Shangqing Zhao, Zhe Qu, Zhengping Luo, Zhuo Lu, and Yao Liu,
University of South Florida

https://www.usenix.org/conference/nsdi20/presentation/zhao

This paper is included in the Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)

February 25–27, 2020 • Santa Clara, CA, USA

978-1-939133-13-7

Open access to the Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20) is sponsored by

NetApp®

Comb Decoding towards Collision-Free WiFi

Shangqing Zhao, Zhe Qu, Zhengping Luo, Zhuo Lu, Yao Liu
University of South Florida, Tampa FL 33620.

Abstract

Packet collisions happen every day in WiFi networks. RTS/CTS is a widely-used approach to reduce the cost of collisions of long data packets as well as combat the hidden terminal problem. In this paper, we present a new design called comb decoding (CombDec) to efficiently resolve RTS collisions without changing the 802.11 standard. We observe that an RTS payload, when treated as a vector in a vector space, exhibits a comb-like distribution; i.e., a limited number of vectors are much more likely to be used than the others due to RTS payload construction and firmware design. This enables us to reformulate RTS collision resolution as a sparse recovery problem. We create algorithms that carefully construct the search range for sparse recovery, making the complexity feasible for system design and implementation. Experimental results show that CombDec boosts the WiFi throughput by 33.6% – 46.2% in various evaluation scenarios.

1 Introduction

CSMA/CA is fundamental for WiFi to coordinate multiple nodes to access the wireless channel. RTS/CTS is a widely-used mechanism in WiFi, which uses short RTS packets for fast collision inference, transmission path check as well as combating the hidden terminal problem [5]. In many early WiFi products, RTS/CTS was disabled due to the concern of overhead [32]. With the substantial increase of data demand in recent years, WiFi data packets become longer and longer, and today's WiFi devices send an RTS packet when the size of a data packet exceeds a given threshold. The threshold is usually set to around 2,300 bytes [1, 4, 6, 7, 9] to balance the performance and the overhead. RTS/CTS is also used in advanced WiFi functionalities, such as beamforming and MU-MIMO [5]. In addition, global RTS/CTS [41, 45, 64] has also been proposed for cross-technology communications.

The essence in the RTS/CTS mechanism is to trade a small cost of the RTS collision for a potentially large cost of data collision. In this paper, we revisit the RTS collision problem and present the comb decoding (CombDec) system to resolve RTS collisions without changing the 802.11 standard and thus improve the wireless channel utilization as well as the network throughput.

The observation behind designing CombDec is that the data payload of an RTS packet, when treated as a vector in a vector space, exhibits a comb-like distribution. Specifically, the RTS content consists of 160 bits, which leads to 2^{160} possibilities. We find that the standard-structured data fields in RTS actually result in at most around 2^{21} possible contents in today's WiFi networks. Therefore, the probability distribution of such contents will exhibit a comb-like shape: only up to 2^{21} out of 2^{160} contents having non-zero probabilities.

As a result, we reformulate the RTS collision problem as a weighted sum problem: we consider the received signal due to an RTS collision is the sum of all 2^{21} RTS contents transmitted at the same time, but with different channel weights. An RTS content has a non-zero channel weight if it is actually transmitted, and zero weight otherwise. Then, resolving the collision is equivalent to solving for the channel weight for each RTS content. The key observation to solve the problem is that a vast majority of the 2^{21} channel weights should be zeros because a collision involves only a few RTS packets in a real-world network. In other words, the vector that includes all channel weights is sparse, which opens a path to use sparse recovery [18, 19, 47] to resolve RTS collisions.

One significantly challenging issue around collision resolution based on sparse recovery is the computational complexity because we start from around 2^{21} possibilities of RTS signals to resolve a collision. To cope with this issue, we analyze how a key RTS data field, `Duration` (that specifies the time duration an RTS packet reserves), is constructed in state-of-the-art 802.11 firmware. A comprehensive set of 802.11ac packet traces are also collected to understand the distribution of `Duration` in various scenarios. It is found that today's firmware imposes extra restrictions on `Duration` and is biased towards a limited number of value selections, and the distribution of `Duration` in real-world packets is highly uneven and patterned. Based on this observation, CombDec is designed with two key components: (α, β)-construction

and γ-decimation, which adaptively narrow down the search range in sparse recovery to a set of only hundreds of potential RTS signals, making system design of collision resolution practical for WiFi networks.

We implement CombDec in a 20-node network testbed, and evaluate it in different scenarios. Experimental results demonstrate that our design has both direct and indirect impacts on today's WiFi systems and setups.

Direct Impact: Today's WiFi devices usually adopt a conservative RTS threshold (i.e., around 2300 bytes), which results in 30% - 45% data transmissions initiated by RTS (according to our packet trace collection and analysis). By directly using CombDec with current RTS settings, we find via experiments that CombDec is able to decode 98% of two-RTS collisions and improve the network throughput by up to 23.3%.

Indirect Impact: CombDec offers a new capability of decoding RTS collisions and in fact encourages changing today's WiFi setups for more RTS transmissions. Therefore, by reducing the RTS threshold to zero and letting every device always send RTS before data (indicating that most collisions in the network become RTS collisions), CombDec significantly improves the network throughput by up to 46.6% in experimental evaluations.

The design of CombDec is the first systematic work towards resolving RTS collisions in WiFi networks. It is non-invasive and redefines the role of the RTS functionality and pushes WiFi towards a collision-free environment.

2 Motivation and Design Intuition

In this section, we introduce the motivation and key idea of reformulating the problem of packet collisions.

2.1 Packet Collision and Resolution

We use a noise-free, flat-fading uplink scenario as a simple motivating example: Alice and Bob send their packets to the AP at the same time. Alice's and Bob's packets consist of L time-domain baseband symbols, represented by vectors[1] $\mathbf{x}_A \in \mathcal{X}$ and $\mathbf{x}_B \in \mathcal{X}$, respectively, where $\mathcal{X} \subset \mathbb{C}^{L\times 1}$ denotes the set of all possible baseband symbol vector for $\mathbf{x}_A$ and $\mathbf{x}_B$, and $\mathbb{C}^{L\times 1}$ is the L-dimensional complex vector space.

Then, the received signal at the AP can be written as

$$\mathbf{y} = h_A\mathbf{x}_A + h_B\mathbf{x}_B, \tag{1}$$

where $h_A, h_B \in \mathbb{C}$ ($\mathbb{C}$ denotes the complex plane) are the channel gains from Alice and Bob to the AP, respectively.

If we look at the collision (1) and assume that Alice's and Bob's signals go through similar channel conditions to the AP, Alice's or Bob's signal will have an SNR around 0dB due to mutual interference. Simply given (1), the AP is less likely to recover Alice's or Bob's signal due to two major reasons.

- If the AP adopts a traditional decoding design, it cannot decode a signal with SNR around 0dB, because an acceptable SNR is usually 10dB or above [29] for WiFi.
- Although multi-user detection [59] has been developed as a vital solution to decode multiple user's signals, this technique in general requires that users employ distinct spread spectrum codes [60] to differentiate themselves at the signal level. Nonetheless, there is no such code design in WiFi.

Apparently, additional information is needed to resolve the collision (1). Our key observation is that the RTS packet format itself provides valuable information for collision resolution in a WiFi network.

2.2 Anatomy of RTS in WiFi

The RTS/CTS mechanism lets a sender reserve the channel by sending an RTS packet first. Once the receiver replies with a CTS packet, the sender transmits the data packet. The RTS packet specifies a network allocation vector (NAV), which is the total time duration it wants to reserve, including the time durations of the CTS, the data and the ACK.

The data payload in an RTS packet consists of 20 bytes or 160 bits, and we denote it as an RTS data vector $\mathbf{b} \in [0,1]^{160}$, where $[0,1]^{160}$ is the space for all vectors with length 160, whose element is either 0 or 1. At the PHY layer, the data vector $\mathbf{b}$ is interleaved, coded and modulated into a signal vector $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ is the set of all values of $\mathbf{x}$. These processes together can be denoted as a one-to-one function mapping $f : [0,1]^{160} \to \mathcal{X}$, which converts the RTS data vector $\mathbf{b}$ to the RTS signal vector $\mathbf{x} = f(\mathbf{b})$. As f is one-to-one correspondence, $|\mathcal{X}| = 2^{160}$ ($|\cdot|$ denotes the cardinality, or the number of elements, of a set).

We observe that all RTS data vectors in $[0,1]^{160}$ are not equally probable in the real world because all data fields in RTS are well structured and specified.

- `FrameControl` contains 2 bytes specified in 802.11.
- `Duration` is the 2-byte NAV in microseconds. The last bit is set to 0 and thus it holds up to 2^{15} values.
- `RA` and `TA` (6 bytes each) are the destination and source addresses, respectively. As today's WiFi is widely used for Internet access, stations communicate mostly with the AP. The AP knows that `RA` in an RTS packet sent to it is its own address and `TA` should be the address of one of its stations. Suppose that a dense network can support 2^6 stations (e.g., Linksys EA8500 firmware supports up to 51 stations [11]) and the number of possible values of `TA` in RTS is 2^6.
- `FCS` (4 bytes) is for error detection and relies on other data fields. It provides no additional information.

Thus, from the AP's perspective, the number of RTS data vectors of interest is 2^{15} (from `Duration`) $\times$ 2^6 (from `RA`/`TA`) = 2^{21} in the full RTS vector space $[0,1]^{160}$. As a result, the

[1] Throughout this paper, a vector is by default a column vector instead of a row vector, unless otherwise specified.

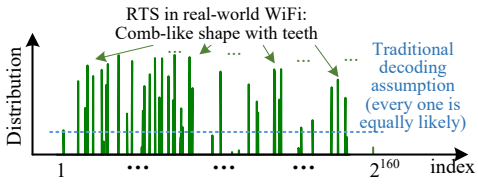

Figure 1: Example: distribution of RTS signal vectors.

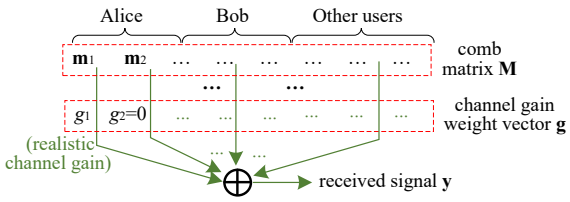

Figure 2: Reformulation of network collision.

number of RTS signal vectors of interest is also 2^{21} in the signal vector space $\mathcal{X}$ with $|\mathcal{X}| = 2^{160}$. If we index all vectors in $\mathcal{X}$ from 1 to 2^{160} and measure via the probability distribution how each vector is likely to be seen in the real world, we will obtain a comb-shaped distribution similar to the example shown in Figure 1. We call an RTS signal vector a *tooth vector* if the probability that it can be seen at the AP is positive. Although the index goes from 1 to 2^{160} in Figure 1, the number of tooth vectors should be no less than 2^{21} according to our analysis. The comb-shaped distribution is in evident contrast to the traditional decoding assumption (also illustrated in Figure 1) that all potential values of a signal vector are equally probable [31]. This opens a door for us to go beyond traditional decoding to resolve RTS collisions.

2.3 Idea of Collision Resolution

Based on the observation from Figure 1, we present the basic idea regarding how to resolve an RTS collision.

2.3.1 Problem Reformulation

Denote by $\mathcal{M} = \{\mathbf{m}_i\}_{i\in[1,M]}$ ($M = |\mathcal{M}|$) the set of tooth vectors. Define the comb matrix $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_M]$ (i.e., each column in $\mathbf{M}$ is a tooth vector). The AP can preconstruct the comb matrix $\mathbf{M}$ by inserting all possible RTS to $\mathbf{M}$ to form the columns. The AP's goal is to find exactly which ones in $\mathbf{M}$ are actually involved in the collision.

Mathematically, we reformulate the collision problem to an equivalent one: assume that all tooth vectors in $\mathbf{M}$ are transmitted to the AP, but they go through different wireless channels. In particular, the tooth vectors involved in the actual collision go through the wireless channels with realistic channel gains, but those not involved in the collision go through the channels with zero channel gains. For example, in Figure 2, Alice, Bob and other users have different tooth vectors. The received signal $\mathbf{y}$ is considered as the sum of all these tooth vectors weighted by different channel gains. The channel gain weight of a tooth vector is the realistic channel gain if it is indeed transmitted, and zero otherwise. If Alice's transmitted signal is $\mathbf{m}_1$, the channel gain weight g_1 for $\mathbf{m}_1$ is the real channel gain between Alice and the AP, and the weights for the rest of Alice's tooth vectors are all zeros (e.g., $g_2 = 0$ as Alice transmits $\mathbf{m}_1$ not $\mathbf{m}_2$). As such, the received signal $\mathbf{y}$ can be reformulated as

$$\mathbf{y} = \sum_{i=1}^{M} \mathbf{m}_i g_i = \underbrace{[\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_M]}_{\text{comb matrix } \mathbf{M}} \underbrace{[g_1, g_2, \cdots, g_M]}_{\text{channel gain weight vector } \mathbf{g}}^{T}, \quad (2)$$

where $\mathbf{g}$ is called the channel gain weight vector and $\cdot^T$ denotes the matrix transpose. Based on (2), collision resolution is equivalent to solving for unknown $\mathbf{g}$ given $\mathbf{y}$ and $\mathbf{M}$. Then, the tooth vectors actually involved in the collision correspond to non-zero elements in the solved $\mathbf{g}$.

2.3.2 Solution based on Reformulation

There are two key observations on the reformulation in (2).

- The unknown channel gain weight vector $\mathbf{g}$ is sparse in a real-world network because of two reasons: (i) There is no self-collision. As shown in Figure 2, if Alice sends a tooth vector $\mathbf{m}_1$, we have $g_1 \neq 0$; and any other tooth vector belonging to Alice will have a zero channel gain weight (e.g., $g_2 = 0$) since there is no way Alice transmits both $\mathbf{m}_1$ and $\mathbf{m}_2$. (ii) Because of the random backoff in WiFi, a collision is likely caused by only several users transmitting at the same time. Thus, $\mathbf{g}$ should include only several non-zero elements.
- The comb matrix $\mathbf{M}$ should exhibit a nearly random matrix property. Each tooth vector $\mathbf{m}_i$ in $\mathbf{M}$ is mapped from an RTS data vector through interleaving and error-correction coding. Their main purpose is to scramble and re-map all bits into a larger bit space in a (nearly) random way such that the error-correction performance approaches the random coding performance in Shannon's capacity [24].

Given the sparse property and nearly random matrix property, the channel gain weight vector $\mathbf{g}$ should be recovered with high probability by $\mathcal{L}_1$-norm minimization according to the theory of compressive sensing [26, 28, 38]. Thus, resolving an RTS collision leads to the following optimization.

$$\begin{aligned} &\text{Given:} && \text{comb matrix } \mathbf{M} \text{ and received signal } \mathbf{y}, \\ &\text{Objective:} && \mathbf{g}_{\text{solution}} = \arg\min \|\mathbf{g}\|_1, \text{ subject to } \mathbf{y} = \mathbf{M}\mathbf{g}, \end{aligned} \quad (3)$$

where $\|\mathbf{g}\|_1$ is the $\mathcal{L}_1$-norm of $\mathbf{g}$ (i.e., $\|\mathbf{g}\|_1 = \sum_{g_i\in\mathbf{g}} |g_i|$).

The theoretical framework lays out a promising path towards resolving RTS collisions. Although the $\mathcal{L}_1$-norm minimization in (3) can be solved by many efficient algorithms

[17–19, 27, 62], directly applying (3) to collision resolution incurs an unbearable cost because the comb matrix **M** consists of up to $2^{21} = 2,097,152$ tooth vectors according to our initial analysis in Section 2.2. Thus, significant challenges exist to make collision resolution meaningful and practical.

3 Construction of Comb Matrix

The initial step towards our CombDec design is to find a way to reduce the size of the comb matrix **M** (that can include 2^{21} tooth vectors) for a low-cost solution. In this section, we analyze the 802.11 standard, firmware and packet traces to show that there is a feasible way to significantly reduce the size of 2^{21}. Then, we design two algorithms, (α, β)-construction and γ-decimation, to reduce 2^{21} to only a few hundreds, while maintaining the high performance for collision resolution. The use of (α, β)-construction and γ-decimation clears the major hurdle towards system implementation.

3.1 Standard and Firmware based Analysis

As aforementioned in Section 2.2, **M** consists of 2^{21} tooth vectors because of `Duration` and `RA`/`TA` fields in RTS. The `Duration` field specifies the 15-bit NAV in microseconds with $2^{15} = 32,768$ potential values, which largely contribute to the size of **M**. 802.11 specifies that the NAV is computed as one data packet duration, plus one CTS, one ACK, and three SIFS durations. Given a network setup, SIFS, CTS and ACK durations are usually fixed (e.g., SIFS is fixed to be 16 μs in 802.11ac at 5GHz). Thus, the value space of the NAV depends dominantly on the value space of the time duration of a data packet. In 802.11, the time duration of a data packet is bounded by aPPDUMaxTime, the maximum time duration of a data packet (in μs), and indirectly bounded by aPSDUMaxLength, the maximum payload length of a data packet (in bytes). As aPPDUMaxTime for 802.11ac is $5,484\mu s$, the space size of the NAV is reduced from 32,768 to at most 5,484 in the 802.11ac network.

Today's WiFi chipsets may still avoid transmitting a long packet (close to $5,484\mu s$) due to the cost consideration or hardware limitations. Hence, drivers implement their own packet length constraints on a data packet, which is usually less than the standard-defined aPSDUMaxLength or aPPDUMaxTime. We perform comprehensive code analysis on WiFi drivers and find that different vendors indeed pose different constraints on their own chipset, further limiting the value selection of the NAV. The detailed firmware analysis can be found in Appendix A.

As a result, we can leverage these constraints to further reduce the value space of the NAV in RTS. However, many WiFi drivers (in particular 802.11ac ones) are still proprietary and distributed in the binary form. It is not practical to study every WiFi chipset/firmware and optimally minimize the value space of the NAV in RTS packets. In what follows, we analyze real-world packet traces to develop a generic way to narrow down the value space.

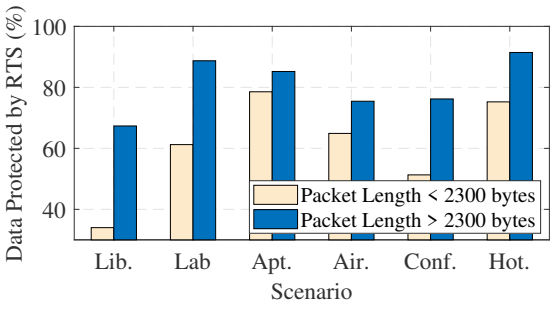

Figure 3: Percentages of data packets protected by RTS.

3.2 Packet Trace based Analysis

The key to reducing the size of the comb matrix is through shrinking the value space of NAV in RTS. We have shown that a WiFi driver can restrain the value space of NAV. Moreover, implementation-dependent rate control and data aggregation in proprietary WiFi drivers are not likely to produce uniformly distributed NAV values, but may be more biased towards certain selections and yield a NAV distribution similar to Figure 1. Our objective is to collect massive packet traces to understand the NAV distribution. Then, we create generic algorithms to select those NAVs that are the most likely to be seen for constructing the comb matrix **M**.

The first step towards understanding the NAV distribution in real-world RTS packets is to collect a substantially large number of packet traces for analysis. As no set of 802.11ac packet data is publicly available, we conducted our own measurements and collected in total 1.3 TB packet trace data with 2.33 billion packets in realistic environments[2], including (i) a public library (65.21 GB), (ii) three academic conferences (31.14 GB), (iii) five residential communities (65.69 GB), (iv) three major-brand hotels (109.48 GB), (v) four major US airports (88.5 GB), (vi) a university research lab (938.81 GB). The library, conference, and airport data traces were measured only within the business hours (i.e., 9am–5pm).

Figure 3 shows the the percentages of data packets that are protected by RTS among all data packets collected in different scenarios. Although a typical RTS threshold is set to around 2300 bytes [1, 4, 6, 7, 9], we see from Figure 3 that data packets of less than 2300 bytes are still likely to be protected by RTS. For example, in the hotel scenario, 78.55% data packets are initiated by RTS even when their lengths are less than 2300 bytes. Moreover, around 70% - 90% data packets of over 2300 bytes are protected by RTS in different scenarios. Overall, we observe from Figure 3 that RTS is still

[2]Note that the payloads of all data packet were removed after the collection to avoid the privacy concern.

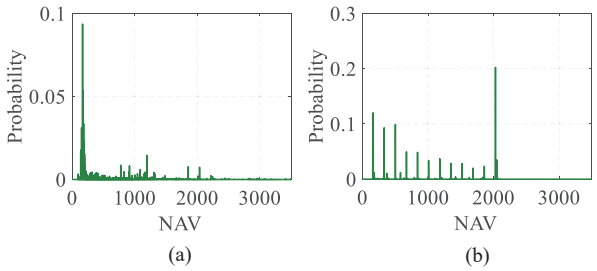

Figure 4: Distribution of NAVs from (a) the airport dataset, and (b) all Belkin devices in all datasets.

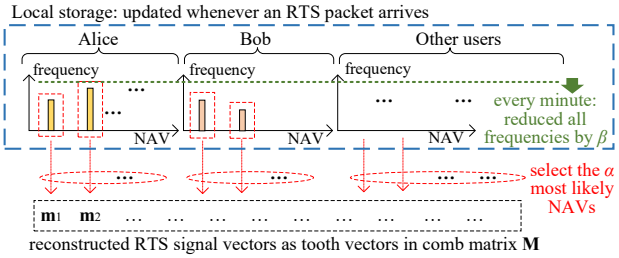

Figure 5: (α, β) construction running at the AP.

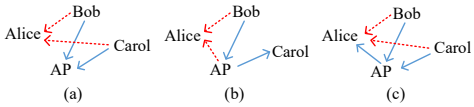

Figure 6: Different collision scenarios in Alice's view.

intensively used in today's Wi-Fi networks.

Looking into the NAV values in RTS packets, we observe that these values are unevenly distributed. For example, Figure 4(a) shows the NAV distribution of RTS packets in the airport dataset, which reveals that many NAV values (particularly around 230 μs) are much more likely to be observed than the others. The NAV distribution also depends on a WiFi driver. For example, Figure 4(b) plots the distribution of the Belkin WiFi driver measured from RTS packets sent by Belkin devices (recognized by MAC addresses) in all datasets. The figure shows that the distribution is quite patterned, indicating that the driver has several NAV levels to construct payloads. We observe uneven or patterned distributions in all datasets and offer a more detailed analysis in Appendix B.

Thus, if we only choose the most likely NAV values (instead of all possible values) to construct the comb matrix M, the size of M should be substantially reduced. In addition, our design should not be device/firmware specific. For example, it may be possible to select NAV values based on the pattern of Belkin devices in Figure 4(b). But this method is too cumbersome because we have to examine the behaviors of all different WiFi devices. Our strategy is to use an online algorithm that actively computes the NAV distribution of a device, and then selects the most likely NAV values from the computed distribution to construct the comb matrix **M**.

3.3 The (α, β)-Construction Algorithm

To select the most likely NAV values to construct **M**, a node (either the AP or a station) should store the distribution of the NAV values in RTS packets from every other node in a network. In addition, the node should keep updating the storage to account for nodes joining or leaving the network. To this end, we propose the (α, β)-construction algorithm running at individual nodes to construct **M**.

3.3.1 Algorithm Design

For a node that runs the algorithm, it records the frequency (i.e. the number of appearances) of a NAV value in RTS packets transmitted by every other node in the network. When a new RTS packet with a NAV value is decoded, the node will increase the frequency of that NAV value by one in its local storage. There are two key factors α and β in the algorithm.

- Coverage factor α: For every other node in the network, the algorithm selects the α NAV values with the highest frequencies to form the tooth vectors and the comb matrix.
- Forgetting factor β: The algorithm decreases the frequencies of all NAV values by β every minute. The minimum frequency is always set to be zero. And if the frequencies of all NAV values associated with a node become zero, the node's information will be all removed from the storage as it is considered no longer active or out of the network.

The (α, β)-construction algorithm works differently for the AP and stations. Figure 5 shows how it works at the AP: the AP stores the frequency of each possible NAV value from each station. When a collision happens at the AP, it knows the collision must be due to at least two stations (which could be Alice, Bob, or others) transmitting to it. Thus, the AP selects the α most likely NAV values from Alice, constructs an RTS data vector using each of these values, together with Alice's MAC address as `TA` and its own MAC address as `RA`, then maps each RTS data vector by function f (including interleaving, error-correction coding, modulation, IFFT) in Section 2.2 to a signal vector (which is a tooth vector in the comb matrix **M**). Then, the AP repeats the same process for Bob and all other stations to finally obtain the full **M**.

A station's construction of the comb matrix differs from the AP. For example, as shown in Figure 6, Alice observes a collision (a) when two other stations Bob and Carol are transmitting to the AP, (b) when one other station Bob is transmitting to the AP and at the same time the AP is transmitting to a third station Carol, or (c) when the AP is transmitting to Alice while Bob and Carol are transmitting to the AP. In all three cases, collision resolution is only meaningful for Alice in case (c) because the collision in case (c) includes the AP's

Table 1: Miss rate and average size of **M**.

α=600 β=10	Miss rate	Ave. # of nodes	# of tooth vectors in **M**
Library	6.9 %	12.8	7680
Conferences	3.6 %	11.6	6960
Apartments	1.5 %	6.5	3900
Hotels	0.7 %	10.3	6180
Airports	8.8 %	18.8	11280
Lab	5.7 %	6.3	3780

signal intended for Alice. Even when Alice successfully resolves cases (a) and (b), Alice only knows that there is no signal of interest and then stops. Therefore, the construction is sufficient for Alice as long as case (c) can be resolved by Alice. According to case (c), a station should construct the comb matrix by using RTS signal vectors from the AP to itself and other RTS signal vectors from other stations to the AP. The construction process is similar to Figure 5.

3.3.2 Selections of (α,β) and Cost Evaluations

The size of the resultant comb matrix **M** depends on both α and β. In particular, α is the number of tooth vectors from one node, and β in fact determines how many nodes will be used in constructing **M** because (i) all frequencies are decreased by β every minute and (ii) a node will be removed from the construction when all its frequencies become zero. The algorithm with a larger β forgets nodes faster, thereby reducing the number of nodes used for constructing **M**.

The performance of (α,β)-construction can be evaluated by the miss rate, defined as the probability that when an RTS packet arrives at a node, **M** constructed by the algorithm at the node does not include the NAV value in the RTS packet. A large α and a small β are able to reduce the miss rate, but at the same time increase the size of **M**, incurring more cost.

Our objective is to find the pair of (α,β) to balance the miss rate and the complexity for general WiFi scenarios. To this end, we simulate a WiFi network in each of the packet datasets, replay all collected packets to simulate RTS arrivals at each node, and measure the miss rate of the algorithm with different values of α and β. Table 1 shows one selection of (α,β)=$(600,10)$ for all scenarios that achieves a good balance between the miss rate and the size of **M** (measured by α multiplying the average number of nodes used for constructing **M**). We can see that all miss rates are below 9% with around 4,000–12,000 tooth vectors in **M**.

3.4 The γ-Decimation Algorithm

Through 802.11 standard analysis, firmware analysis, packet trace analysis and (α,β)-construction, we have dramatically shrink the size of **M** from the initial 2,097,152 tooth vectors to 12,000 or fewer vectors. All these push the optimization in (3) to the practice. However, finding the $\mathcal{L}_1$-norm minimization with $12,000$ vectors in (3) still incurs a substantial cost. We propose γ-decimation to further reduce such a cost while maintaining the high performance.

Denote by M the number of tooth vectors in **M** constructed by (α,β)-construction. The basic idea of the γ-decimation algorithm ($\gamma > 1$ is called decimation rate) is to select, based on the received signal vector **y**, M/γ vectors out of all M tooth vectors in **M** to form a decimated comb matrix $\mathbf{M}'$.

The design intuition is that the received signal **y** contains only several tooth vectors in **M** that we aim to find out. If we compute the correlation between **y** and each tooth vector $\mathbf{m}_i$ in **M**, defined as $C(\mathbf{y},\mathbf{m}_i) = \|\mathbf{m}_i^H\mathbf{y}\|_2$ ($\cdot^H$ denotes conjugate transpose and $\|\cdot\|_2$ denotes the $\mathcal{L}_2$-norm), we then obtain M correlation values. Due to the property of correlation, we should observe a high correlation value if a tooth vector is indeed included in **y**, and a low correlation value otherwise. However, due to channel noise and limited length of tooth vectors, some tooth vectors not in **y** may also exhibit high correlation values. But it is not necessary to exactly identify which tooth vectors with high correlation values are indeed in **y** at this stage, γ-decimation just chooses M/γ tooth vectors that have the highest correlation values to form the decimated comb matrix $\mathbf{M}'$. It is very likely that tooth vectors involved in the collision are included in $\mathbf{M}'$ as long as M/γ is sufficiently large. We provide theoretical analysis for the performance of γ-decimation and show that all RTS signals involving a collision will survive the decimation and be included in $\mathbf{M}'$ with high probability in Appendix C.

As a result, the final comb matrix for (3) is constructed as follows: (i) (α,β)-construction constructs the comb matrix **M** with M tooth vectors (this steps ensures a low miss rate), (ii) γ-decimation decimates **M** into the decimated comb matrix $\mathbf{M}'$ with only M/γ tooth vectors (this steps ensures that tooth vectors involving the actual collision are preserved with high probability), and (iii) the decimated comb matrix $\mathbf{M}'$ is used in (3) for collision resolution to finally identify which tooth vectors are included in the collided signal **y**.

To make (3) feasible for today's systems, $\mathbf{M}'$ should have only hundreds of tooth vectors. As (α,β)-construction maintains up to around $12,000$ vectors (shown in Table 1), the decimation rate γ should be around 12 or more.

3.5 Complexity Analysis

In the following, we estimate the computational complexity and storage complexity of CombDec.

3.5.1 Computational Complexity

We evaluate CombDec's complexity by comparing it with a benchmark, which is the complexity to decode a typical 802.11 data packet with 40MHz bandwidth, 3/4 convolu-

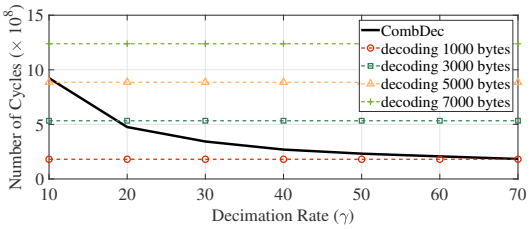

Figure 7: Number of cycles: CombDec vs benchmark.

tional coding, and 64QAM. We describe the major computational operations involved in the benchmark and CombDec.

- The benchmark complexity is proportional to the data payload length, denoted by N bytes. Decoding a data packet requires the FFT and Viterbi algorithms. The packet contains $0.0247N$ OFDM symbols plus 4 PHY headers symbols. The FFT on each symbol requires $64\log(128)$ complex multiplications and $128\log(128)$ complex additions [10]. In addition, the Viterbi algorithm incurs $4^K(10.67N+252)$ real additions and $2^K(10.67N+252)$ real comparisons, where K is the constraint length of convolutional code [12].
- In CombDec, suppose (α,β)-construction maintains M tooth vectors of length L, the correlation of the received signal with each of the tooth vector in γ-decimation needs a total of ML complex additions and multiplications. Among all correlation values, selecting the M/γ largest values incurs $M + M\log(M)/\gamma$ comparisons by using the max heap tree approach [3]. The complexity of $\mathcal{L}_1$ minimization depends on an iterative algorithm and is bounded by the cubic polynomial complexity [62]. In order to estimate an exact computational cost, we use simulations to run the primal-dual interior-point algorithm [47] to solve the $\mathcal{L}_1$-norm minimization in M/γ tooth vectors and compute the average numbers of additions, multiplications and comparisons.

As the complexity involves different types of operations, including additions, multiplications and comparisons. We need to have a unified cost utility metric to measure the overall costs of CombDec and the benchmark. We use the number of general CPU cycles as the utility metric [2]. In particular, one real addition or comparison is counted as one cycle and one multiplication is 4 cycles [2]; and a complex operation is 4 times the number of cycles incurred by its real counterpart [10]. Note that an algorithm or a computational operation can be specifically optimized on a particular signal processing software or hardware platform. Our estimation using CPU cycles is not intended to be exactly accurate for an implementation platform, but serves as an approximate way to demonstrate what computational complexity level CombDec is at when compared with the benchmark.

Figure 7 shows the total numbers of cycles incurred by CombDec and the benchmark. In Figure 7, we let (α,β)-construction form $M = 12,000$ tooth vectors to accommodate the airport scenario in Table 1 and set a typical constraint length $K = 7$ in the Viterbi Algorithm. It is observed from Figure 7 that choosing γ to be in [20, 50] leads to the complexity of CombDec roughly equivalent to decoding a packet with 1000–3000 bytes, which makes CombDec ready for system implementation.

3.5.2 Storage Complexity

CombDec also incurs a storage cost. First, CombDec for a device must store the frequency of each NAV value for any other active device in the network. The cost of storing all NAV frequencies is equal to the NAV space size multiplying the average number of active devices. There are 5,484 possible NAV values in 802.11ac and around 18.8 active nodes under (α,β) construction for the airport scenario (shown in Table 1). Hence, the storage cost is 5484 $\times$ 18.8 $\times$ 1 = 101 KB when the frequency value is a one-byte integer. Second, after γ-decimation, all tooth vectors should be stored for $\mathcal{L}_1$-minimization. The storage cost is the number of tooth vectors multiplying the length of a tooth vector. When $\gamma \in [20,50]$, the number of decimated tooth vectors in the airport scenario is 240 to 600. If a typical RTS packet is transmitted at 12 Mbps (4 64-subcarrier OFDM symbols) and the element in tooth vector is a 4-byte complex number, the length of a tooth vector is $64\times4\times4 = 1$ KB. Thus, the cost of storing all these tooth vectors is in the range of $[240,600]$ KB.

Overall, the major storage cost is around [341, 701] KB. This cost is also reasonable for today's WiFi systems. For example, Qualcomm's 802.11ac chipset IPQ4018 has an on-chip memory of 256 MB [50] and related APs cost as low as tens of dollars [8, 13].

4 CombDec System Design

The (α,β)-construction and γ-decimation algorithms pave the way for a feasible system solution to (3). In this section, we present CombDec system design. We first introduce the system architecture, then describe each key component.

We design CombDec as an independent decoder in addition to the traditional 802.11 decoder. Figure 8 shows four major components in CombDec: the prologue module, (α,β)-construction, γ-decimation, collision resolution, and the epilogue module. As shown in Figure 8, CombDec is triggered only when decoding of either the PHY header or the PHY payload fails. In the following, we present the designs of individual CombDec Components.

The Prologue Module: The prologue module does all preprocessing before collision resolution.

Combining Multi-path Signal Components: The received signal may include a number of multi-path components with different time offsets. To improve the performance of CombDec under multi-path fading, we use the maximum ratio combining (MRC) [31] to combine the multi-path signal components. Specifically, denote by $s(n)$ the n-th time-domain

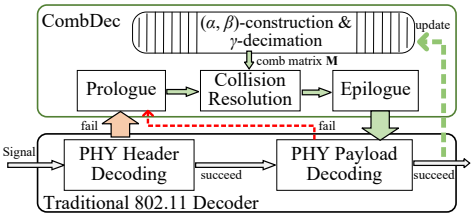

Figure 8: Architecture of CombDec decoder.

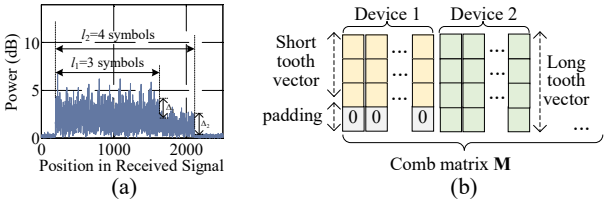

Figure 9: (a) Power level changes in the received signal. (b) Padding in comb matrix construction.

symbol in the received signal. Based on the 802.11 packet preamble, we use a matched filer [32, 58] to detect the time offset and estimate the channel gain of each signal component. Given K components found, we use the MRC [31] to coherently sum all K time-shifted copies of $s(n)$ and obtain $s'(n) = \sum_{k=1}^{K} h_k^* s(n+\delta_k)$, where k-th component having time offset δ_k and channel gain h_k, and h_k^* is the complex conjugate of h_k. Note that signal components may be from different senders in a collision, CombDec does not differentiate them in the prologue module.

Collision Recognition and Early Stop: Next, CombDec decides if $s'(n)$ can be potentially resolved. There are three types of collisions: RTS-only, RTS-data (involving at least one RTS packet and one data packet), and data-only collisions. CombDec will stop if the collision belongs to data-only collision. Note that the recognition does not need to be 100% accurate, it simply provides a way to exclude obvious data-only collisions that cannot be resolved by CombDec.

In WiFi networks, the beginnings of collided packet transmissions are roughly aligned because of CSMA/CA (if we do not consider the hidden terminal problem). Thus, we measure the time durations of different power levels of the received signal to identify the type of a collision. According to 802.11, the data rate for RTS is selected by a station from a limited set of basic rates defined by the AP (e.g., 12 Mbps and 24 Mbps are widely observed in our packet traces). Thus, a RTS time duration can be measured by a very limited number of OFDM symbol durations, e.g., 3 (or 4) OFDM symbol durations for 24 (or 12) Mbps. We record the time duration l_i, from the beginning of the signal $s'(n)$, to the position in $s'(n)$ where the i-th signal power change happens and is larger than a threshold Δ. When the power level reaches the noise floor, we stop and obtain a set of time durations $\mathcal{L} = \{l_i\}_{i\in[1,|\mathcal{L}|]}$. We consider a collision as (i) RTS-only if each value in $\mathcal{L}$ is close to one of the RTS durations corresponding to the basic rate set defined by the AP, (ii) RTS-data if one value is close to an RTS duration and any other value is not close to any of the RTS durations, and (iii) data-only otherwise. Figure 9(a) shows an example of RTS-only collision, where two measured time durations occupy 3 and 4 OFDM symbol durations, respectively, which is recognized as the collision of two RTS packets with different rates.

The Collision Resolution Module: The prologue module outputs either RTS-only or RTS-data signals for collision resolution, (α, β) construction will construct every tooth vectors for the comb matrix. As devices in a network may use different basic rates to transmit RTS packets (leading to different lengths of tooth vectors), CombDec zero-pads the shorter tooth vectors with higher data rates to form the complete comb matrix $\mathbf{M}$ used for collision resolution. Figure 9(b) shows an example of constructing the comb matrix $\mathbf{M}$ by zero-padding shorter tooth vectors. Then, $\mathbf{M}$ is γ-decimated to $\mathbf{M}'$, which is used in the primal-dual algorithm [47] that solves the $\mathcal{L}_1$-minimization in (3). Note that if a collision is RTS-data, the module resolves the collided RTS signal vectors by treating any data signal as the noise, and then leaves the potential decoding of data for the epilogue module.

The Epilogue Module: The collision resolution module yields a small set of potential RTS signal vectors involved in the collision, denoted by $\mathcal{R}$. There are still important questions left: (i) Which RTS in $\mathcal{R}$ a receiver should choose to reply with CTS? (ii) Can we decode the data in the presence of an RTS-data collision? (iii) Moreover, we also need an error detection mechanism to ensure the collision is correctly resolved because errors may happen in CombDec. We first describe how CombDec chooses data or RTS. The AP and stations have different decision making processes.

Decision Making at AP: The AP observes a collision when multiple stations transmit to the AP at the same time.

- Choosing the RTS with the largest NAV: if the collision is RTS-only, the AP chooses the RTS of the largest NAV to reply with CTS. Note that we intend to maximize the channel utilization in this way. A more advanced policy (e.g., considering the utilization and fairness) can be designed and adopted going beyond the main scope of this paper.
- Choosing data over RTS: If the collision is RTS-data, the AP first decodes all RTS packets in the collision resolution module, then proceeds to decode the data. If the data is successfully decoded, the AP chooses data over RTS and sends back the ACK to the sender of the data. This is because data packets are usually longer than RTS packets. Giving priority to data should improve the channel utilization.

Decision Making at Stations: A station observes a collision when multiple other nodes (either other stations or the AP) transmit at the same time. As shown in Figure 6(c), a

station only cares about the AP's signal transmitted to it.

- If $\mathcal{R}$ includes an RTS vector from the AP to the station, the collision is due to the AP transmitting to the station and at the same time at least one other station transmitting to the AP, the station always sends CTS to the AP.
- Otherwise, there is no RTS in $\mathcal{R}$ intended for the station. If the collision is RTS-data, the station proceeds to decode the data because the data may be intended for it; otherwise, the station stops.

Error Checking and Data Decoding: Once a decision is made (choosing RTS or data), we must ensure there is no error with the chosen RTS or we must proceed to decode the data. How to proceed with error checking and data decoding? We find that a traditional 802.11 receiver, as shown in Figure 8, already has a PHY payload decoder with the cyclic redundancy check (CRC) mechanism. Thus, we should leverage the existing architecture to perform the decoding and error checking to minimize the complexity of CombDec.

As a result, if CombDec decides to act on a particular RTS signal or to decode a data signal, it uses interference cancelation to remove all other signals from the collided signal. Specifically, if a decision is to decode the data, CombDec removes all RTS signal vectors in $\mathcal{R}$ from the received signal $s'(n)$ and write the resultant signal as $s'_c(n) = s'_c(n) - \sum_{i=1}^{|\mathcal{R}|} r_i(n)g_i$, where $r_i(n)$ and g_i are the n-th element and the channel gain weight of the i-th RTS tooth vector in $\mathcal{R}$, respectively. Similarly, if the decision is to act on the j-th RTS vector (i.e., choose the j-th RTS in $\mathcal{R}$ to reply with CTS), CombDec removes all other RTS tooth vectors from the $s'(n)$ and the resultant signal becomes $s'_c(n) = s'_c(n) - \sum_{i=1, i\neq j}^{|\mathcal{R}|} r_i(n)g_i$.

Finally, the signal $s'_c(n)$ goes from CombDec into the traditional PHY payload decoder, which performs decoding and error checking, then passes the correct RTS or data to the MAC layer for protocol processing (e.g., replying with CTS). In addition, the MAC address and data rate information of a correct RTS packet is stored and its NAV value is also updated in (α, β)-construction as shown in Figure 8.

5 Evaluation

In this section, we evaluate the performance of CombDec. We first introduce the experimental setups, then measure the performance benefits CombDec brings to WiFi networks.

5.1 Setups

Testbed Implementation: We have implemented the prototype of CombDec on 20 USRP X310/300 devices. Each device is equipped with two UBX-160 daughterboards and two VERT 2450 antennas. We implement a basic 802.11ac PHY-MAC architecture with 20-MHz settings: 64 OFDM subcarriers (including 48 data subcarriers), BPSK, QPSK, 16QAM, and 64QAM modulations, Alamouti code based MIMO, and convolutional coding at the PHY layer, and CSMA/CA scheme with an initial contention window size of 8 [5] at the MAC layer. Control packets including RTS, CTS, and ACK are also implemented.

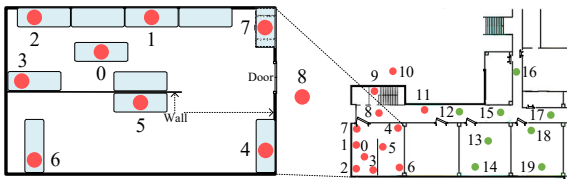

Figure 10: Environment for experiments.

Experimental Settings: We aim to measure the performance of CombDec in a realistic indoor environment inside a campus building shown in Figure 10. Network nodes are placed at various locations and transmit packets whose contents are generated according to our collected 802.11ac packet traces.

Note that we do not implement the 256QAM modulation as we found no single packet using a data rate associated with 256QAM in all collected packet traces. This does not severely affect the performance evaluation since 256QAM is intended only for very high SNR conditions.

We use the following default settings for experiments (unless otherwise specified): (i) all nodes are saturated; i.e., they always have packets to transmit; (ii) α=600, β=10, and γ=20 for CombDec; (iii) the airport dataset is used to generate packets as it represents the most crowded condition in all datasets; (iv) all nodes have the same transmit power.

Evaluation Metrics: We use the following metrics to evaluate the real-time performance of CombDec.

- Success probability of collision resolution is defined as the probability that CombDec recovers exactly all collided RTS signals. The recovery will be considered as a failure if CombDec recovers only a subset of collided RTS signals or mis-identifies an RTS signal not involved in the collision.
- Normalized throughput (or utilization efficiency), is defined as the percentage of the time duration on the wireless channel that is used to deliver data packets. An ideal network should have a normalized throughput of 1. However, control signals (e.g., RTS/CTS) and collisions deteriorate the throughput. We aim to show how much channel utilization efficiency CombDec can improve via resolving RTS collisions.
- Throughput gain ratio, defined as the ratio between the increased normalized throughput from traditional 802.11 decoding to CombDec and the normalized throughput under traditional decoding. Throughput gain ratio can directly reflect how CombDec improves the network performance.

5.2 Success Probability

We first evaluate the success probability of CombDec for collision resolution. In this evaluation, multiple nodes only

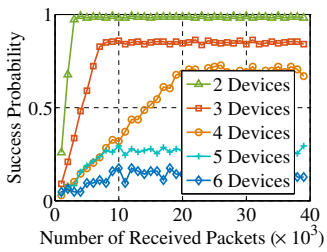

Figure 11: Success probabilities under 2-6 transmitters.

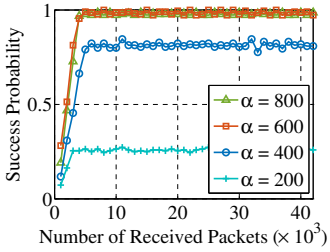

Figure 12: Impact of value of α in (α, β)-construction.

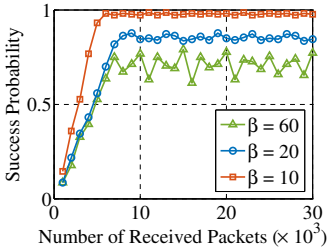

Figure 13: Impact of value of β in (α, β)-construction.

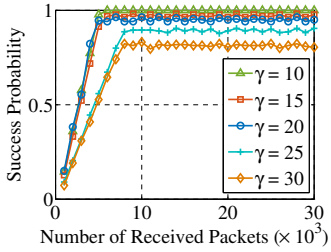

Figure 14: Impact of value of γ in γ-decimation.

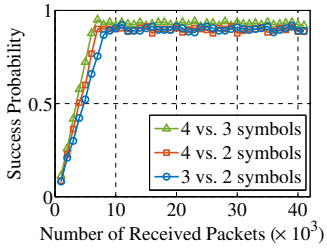

Figure 15: Resolving collisions with different rates.

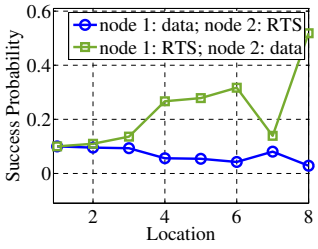

Figure 16: Resolving RTS-data collisions.

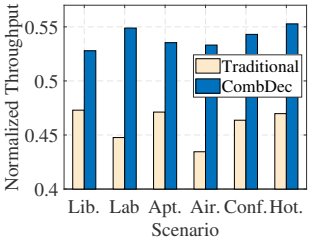

Figure 17: Throughputs under different scenarios.

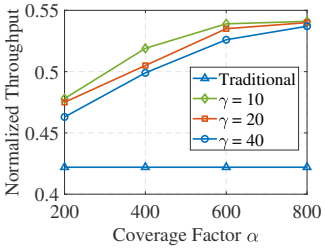

Figure 18: Throughputs with different α and γ.

transmit RTS packets to the AP placed at location 0 in Figure 10, where the success probability is measured.

Impact of Number of Transmissions: We evaluate CombDec's ability to resolve collisions due to two and more transmissions. To this end, we place multiple transmitters at location 1, which send RTS packets at the same time to the AP at location 0. Figure 11 shows the success probabilities that the AP resolves the collision under 2–6 transmitters. The success probability is computed for every 1,000 packets received. It is noted from Figure 11 that as the total number of received packets at the AP increases, CombDec gradually gains the NAV information in RTS packets, and thus the success probability also increases and finally remains stable. It is also observed that CombDec is able to resolve 98% of two-node collisions and 86% of three-node collisions. The performance degrades when the number of transmitters increases. However, a collision caused by 5 or more WiFi nodes is much less frequent because of the random backoff in CSMA/CA.

Values of α and β: We evaluate the impacts of α and β on the success probability. We place two nodes at location 1 that transmit RTS packets at the same time to the AP at location 0. Figure 12 shows that as α goes from 200 to 600, the success probability increases from 0.28 to 0.98; and further increase of α will not substantially improve the success probability. Figure 13 shows the impact of β on the success probability. We observe that when β increases from 10 to 60 (i.e., CombDec forgets the history faster), the success probability reduces from 0.98 to 0.77. From both figures, we can see that the uniform selection of $\alpha = 600$ and $\beta = 10$ yield very high performance for the airport scenario, and accordingly are also suitable for other less crowded scenarios.

Value of γ: We adopt the same setup in Figure 13 to evaluate the impact of γ. Figure 14 illustrates that gradually increasing γ does not severely decrease the success probabilities. For example, when γ becomes 30, the success probability reduces to 0.831. This indicates that adjusting γ can smoothly balance the performance and the implementation cost.

Impact of Zero-Padding: In each of our packet datasets, we find that a majority of RTS packets are transmitted at the same data rate; however, RTS packets with different rates do exist. These packets have different lengths of 2, 3, or 4 OFDM symbols. Therefore, a minority of collisions involving RTS packets with different rates. CombDec uses zero-padding to solve this issue as discussed in Section 4. We measure the ability of CombDec to resolve such a type of collisions. Hence, we place two nodes at location 1 sending RTS packets with different lengths to the AP. Figure 15 shows that the success probabilities remain approximately the same when RTS packets have different lengths in a collision. From Figure 15, we conclude that CombDec has no difficulty in resolving this type of collisions.

RTS-Data Collisions: CombDec also attempts to resolve an RTS-data collision via canceling the RTS signal from the received signal and then performing decoding. To evaluate such an ability, we place one node (node 1) at location 1 to send RTS to the AP, and place another node (node 2) at one

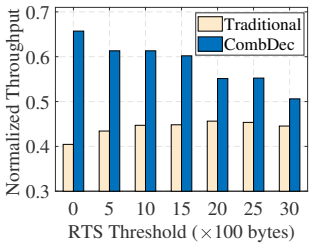

Figure 19: Throughputs with different thresholds

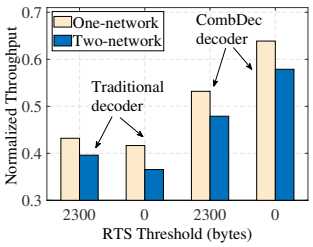

Figure 20: Throughputs in collocated networks.

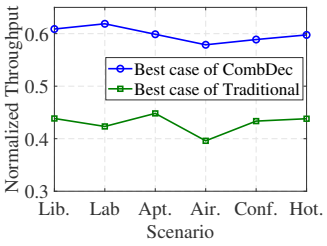

Figure 21: Throughputs in collocated networks.

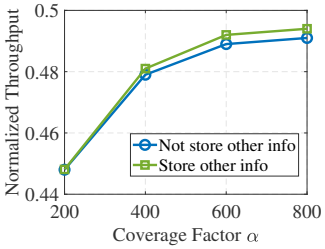

Figure 22: Storing information of other network.

of locations 1–8 to send data to the AP for the first round of experiments, and then let node 1 send data and node 2 send RTS for the second round. The first and second rounds represent the scenarios in which the receiving power of RTS is greater than and less than that of data, respectively. We consider the collision is resolved when both RTS and data packets are decoded successfully. Figure 16 depicts the success probability of collision resolution as node 2's location changes. The figure shows that generally, the success probability is higher when the receiving power of RTS is higher than that of data. This is because CombDec first treats any data packet as the noise to recover any RTS packet from the receiving signal. The results demonstrate that CombDec, primarily designed to handle RTS-only collisions, is capable of resolving RTS-data collisions in some scenarios.

5.3 Network Performance Evaluation

We then evaluate the benefits of CombDec for the network performance. Note that it is impossible to measure the network performance with CombDec under different setups at the same time because the resolution of a collision directly affects follow-on network dynamics. We have to measure the performance under different setups over non-overlapping measurement periods. Therefore, we conduct experiments during off-business hours to minimize the impact of environmental factors on different measurement periods.

5.3.1 Single Network Scenario

We first consider a single-network scenario: 12 nodes are placed at locations 0–11, in which the AP is at location 8, as shown in Figure 10. The network does not run in the MU-MIMO mode (i.e., the AP does not transmit data via MU-MIMO to multiple stations in the downlink). The RTS threshold is set to be 2,300 bytes for all nodes (i.e., RTS is triggered only when a data packet to be transmitted has a length over 2,300 bytes). The value of 2,300 is typical for today's WiFi products (e.g., the default value in Cisco APs is 2347 [9]).

Throughput Improvement: Figure 17 demonstrates the comparisons of normalized throughputs under traditional 802.11 decoding and CombDec. Note that the throughput performance is always measured at the AP. We can see that CombDec is able to uniformly boost the performance of traditional 802.11 decoding. For example, the normalized throughput for the airport scenario increases from 0.43 to 0.53, leading to a throughput gain ratio of (0.53 - 0.43)/0.43 = 23.3%. In all different scenarios, we observe that the throughput gain ratio under CombDec is 11.6%–23.3%.

Improvement by Tuning α and γ: We aim to find if we can improve the performance by tuning α and γ, which are important factors to balance the performance and complexity. Figure 18 shows the normalized throughputs for various α and γ values in the airport scenario. The figure shows that keeping increasing α and decreasing γ do not always lead to evident improvement. For example, when α goes from 600 to 800 and γ decreases from 20 to 10, the throughput under CombDec only increases from 0.491 to 0.494.

Improvement by Reducing RTS Threshold: It is still possible to further improve the network performance. Our observation is that traditionally, an RTS collision is considered not resolvable; therefore, many WiFi devices are conservative to set the RTS threshold. The transmission of a data packet triggers an RTS transmission only when its data size is greater than the threshold. In all previous experiments, the threshold is set as a typical value of 2,300 for today's networks. Now CombDec has the capability of decoding RTS collisions; therefore, it can be beneficial to encourage more RTS transmissions by reducing the threshold. Figure 19 compares the normalized throughputs under traditional 802.11 decoding and CombDec for different RTS thresholds. The figure shows that reducing the threshold generally decreases the throughput performance under traditional decoding; however and interestingly, it substantially boosts the performance under CombDec. The best case for traditional decoding is to set the threshold as 2000–2500, resulting in a normalized throughput of 0.456. By contrast, the best case for CombDec is to remove the threshold and let everyone always send RTS before data, yielding a higher throughput of 0.657. The throughput gain ratio is computed as (0.657-0.456)/0.456 = 44.08%. This encouraging result shows that CombDec has an immediate impact on today's practice of setting the RTS

threshold, and significantly reducing this threshold can push WiFi towards a collision-free environment.

5.3.2 Collocated Networks

Next, we place a new network close to the single network used in previous experiments. In the new network, 8 nodes are placed at locations 12–19 and the AP is at location 15, as shown in Figure 10. The two networks use the same frequency and thus interfere with each other. We call the APs in the original and new networks AP 1 and AP 2, respectively.

Figure 20 shows the throughput performance under different settings in the airport scenario. In Figure 20, the one-network performance is the performance measured at AP 1 in the previous single-network scenario (without the new network); and the two-network performance is measured as the average of the throughputs measured at AP 1 and AP 2. We can observe from Figure 20 that when the new network is placed, the throughput performance degrades due to mutual interference. CombDec still performs better than traditional 802.11 decoding. In the two-network scenario, the best case for CombDec is setting the RTS threshold to 0 (which is also beneficial to solving the hidden terminal problem), yielding a throughput of 0.579; and the best case for traditional decoding has a throughput of 0.396. The throughput gain ratio is thus (0.579 - 0.395)/0.395 = 46.6%, which is also a substantial throughput improvement. Figure 21 compares the best case throughput performance between CombDec (removing the RTS threshold) and traditional coding (setting the threshold to 2,300) in different scenarios. It can be seen that the throughput gain ratio of CombDec is 33.6% – 46.2%.

As discussed in Section 3, CombDec is designed to only store information of its own network. In the two-network scenario, it is possible to enhance the performance of CombDec by letting (α, β)-construction store the information (including MAC addresses, NAVs, and RTS rates) of the other network. Figure 22 shows that storing other network information can further yet slightly improve the throughput performance of CombDec with a fairly large α.

6 Related Work

Interference Cancelation and Mitigation: In the literature, successive interference cancelation (SIC) has been proposed to decode collisions by using either pre-coded signatures or different receiving powers [15, 33, 35, 36, 39, 40, 53, 66]. The time offsets in different packet collisions (e.g., in the presence of hidden terminals) has also been leveraged to resolve collisions [32, 42, 63]. In addition, interference cancelation was widely studied in the full-duplex mode [20,22,23,56,66]. In cross-technology communication, corrupted packets may also be decoded by detecting the interference type [21,37]. A number of studies have also proposed interference alignment and nulling with or without channel state information [43,46]. These approaches cannot be readily adapted to regular WiFi scenarios considered in this paper, where RTS packets collide at the beginning of each transmission.

Multi-user Detection: CombDec is related to multi-user detection that attempts to decode multiple users' signals from the overlapped signal [25, 44, 61]. In cellular networks, CDMA has been widely used to assign distinct spread spectrum codes to different users [60]. However, there is no such code design in RTS packets. Constructive interference [25] is able to receive multiple synchronized transmissions. Nevertheless, it requires all packets have the same content, which is impossible for RTS signals. The work in [51] applied the time division technique to the byte level such that multiple users can share the same packet. This method needs a strict coordination among all users. A multi-user system is built in [44] through sharing multiple channels to users who are allowed to duplicate the signal into these channels. Applying these designs to WiFi requires modification of the standard; in contrast, CombDec is a non-invasive design.

Improving WiFi Performance: Substantial efforts have been devoted to improving the WiFi link performance [14, 16,30,34,48,49,55,57,65]. For example, [14,30,57] focused on optimizing the user selection algorithm in MU-MIMO and [16, 54, 65] aimed to improve the beamforming related techniques. Different algorithms were also investigated to improve the rate adaptation in WiFi [34, 49]. Recently, a rapid picocell switching has also been proposed for wireless transit networks [55]. CombDec is orthogonal to these studies that aim to improve WiFi performance in different aspects. We show that CombDec makes it possible to resolve RTS collisions and pushes WiFi towards a collision-free environment.

7 Conclusion

This paper provides a systematic study to resolve RTS collisions in WiFi networks. Our core contribution is a new decoding system CombDec that uses (α, β)-construction, γ-decimation and sparse recovery to resolve RTS collisions. CombDec does not require changing the 802.11 standard and redefines the role of the RTS functionality in WiFi. We show via system implementation and extensive evaluation that CombDec has a beneficial impact on WiFi networks and substantially improves the throughput performance by 33.6% – 46.2% in various scenarios.

Acknowledgments

We would like to thank Dr. Romit Roy Choudhury for shepherding our paper and the anonymous reviewers for the insightful comments. The work at University of South Florida in this paper was supported in part by NSF under grants CNS-1553304 and CNS-1717969.

References

[1] Arris router setup. http://www.cktv.ru/files/modem/ARRIS_Router_Setup_Web_GUI_UG.pdf, 2012.

[2] How expensive is an operation on a cpu. https://streamhpc.com/blog/2012-07-16/how-expensive-is-an-operation-on-a-cpu/, 2012.

[3] k largest(or smallest) elements in an array. https://www.geeksforgeeks.org/k-largestor-smallest-elements-in-an-array/, 2012.

[4] Linksys user guide. https://content.etilize.com/User-Manual/1026969914.pdf, 2014.

[5] Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11*, 2016.

[6] Dlink user manual. http://files.dlink.com.au/Products/DSL-2740M/Manuals/DSL-2740M_A1_Manual_v1.00(DI).pdf, 2017.

[7] Tplink user guide. https://static.tp-link.com/2017/201712/20171212/1910012191_TL-WR902AC%203.0_UG.pdf, 2017.

[8] GL.iNET B1300 IPQ4018 dual band wifi router. https://www.alibaba.com/product-detail/GL-iNET-B1300-ipq4018-dual-band_60779146003.html, 2018.

[9] Advanced radio settings. https://www.cisco.com/assets/sol/sb/isa500_emulator/help/guide/ae1269129.html, 2019.

[10] Introduction to fft and ofdm. http://shodhganga.inflibnet.ac.in/bitstream/10603/42180/10/10_chapter%202.pdf, 2019.

[11] Linksys official support. https://www.linksys.com/us/support-product?pid=01t80000003ouh0AAA, 2019.

[12] The viterbi algorithm. http://www.cim.mcgill.ca/~latorres/Viterbi/va_alg.htm, 2019.

[13] ZBT apg222. https://www.alibaba.com/product-detail/newest-atheros-chipset-IPQ4018-802-11ac_60541163350.html, 2019.

[14] Narendra Anand, Jeongkeun Lee, Sung-Ju Lee, and Edward W Knightly. Mode and user selection for multi-user mimo wlans without csi. In *Proc. of IEEE INFOCOM*, 2015.

[15] Jeffrey G Andrews. Interference cancellation for cellular systems: a contemporary overview. *IEEE Wireless Communications*, 12:19–29, 2005.

[16] Oscar Bejarano, Roger Pierre Fabris Hoefel, and Edward W Knightly. Resilient multi-user beamforming wlans: Mobility, interference, and imperfect csi. In *Proc. of IEEE INFOCOM*, 2016.

[17] Emmanuel Candes and Justin Romberg. l1-magic: Recovery of sparse signals via convex programming. 4:14, 2005.

[18] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

[19] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE transactions on Information Theory*, 51, 2005.

[20] Bo Chen, Yue Qiao, Ouyang Zhang, and Kannan Srinivasan. Airexpress: Enabling seamless in-band wireless multi-hop transmission. In *Proc. of ACM MobiCom*, 2015.

[21] Gonglong Chen, Wei Dong, Zhiwei Zhao, and Tao Gu. Towards accurate corruption estimation in zigbee under cross-technology interference. In *Proc. of IEEE ICDCS*, 2017.

[22] Lu Chen, Fei Wu, Jiaqi Xu, Kannan Srinivasan, and Ness Shroff. Bipass: Enabling end-to-end full duplex. In *Proc. of ACM MobiCom*, 2017.

[23] Jung Il Choi, Mayank Jain, Kannan Srinivasan, Phil Levis, and Sachin Katti. Achieving single channel, full duplex wireless communication. In *Proc. of ACM MobiCom*, 2010.

[24] Thomas M Cover and Joy A Thomas. *Elements Of Information Theory*. John Wiley & Sons, 2012.

[25] Manjunath Doddavenkatappa, Mun Choon Chan, Ben Leong, et al. Splash: Fast data dissemination with constructive interference in wireless sensor networks. In *Proc. of NSDI*, 2013.

[26] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52:1289–1306, 2006.

[27] David L Donoho, Michael Elad, and Vladimir N Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52:6–18, 2006.

[28] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[29] Jim Geier. Wi-Fi: Define minimum SNR values for signal coverage. *Enterprise Networking Planet: Standards & Protocols*, 2008.

[30] Yasaman Ghasempour and Edward W Knightly. Decoupling beam steering and user selection for scaling multi-user 60 ghz wlans. In *Proc. of ACM MobiHoc*, 2017.

[31] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[32] Shyamnath Gollakota and Dina Katabi. *Zigzag decoding: combating hidden terminals in wireless networks*. 2008.

[33] Shyamnath Gollakota, Samuel David Perli, and Dina Katabi. Interference alignment and cancellation. In *Proc. of ACM SIGCOMM*, 2009.

[34] Aditya Gudipati and Sachin Katti. Strider: Automatic rate adaptation and collision handling. In *Proc. of ACM SIGCOMM*, 2011.

[35] Aditya Gudipati, Stephanie Pereira, and Sachin Katti. Automac: Rateless wireless concurrent medium access. In *Proc. of ACM MobiCom*, 2012.

[36] Daniel Halperin, Thomas Anderson, and David Wetherall. Taking the sting out of carrier sense: interference cancellation for wireless lans. In *Proc. of ACM MobiCom*, 2008.

[37] Anwar Hithnawi, Su Li, Hossein Shafagh, James Gross, and Simon Duquennoy. Crosszig: combating cross-technology interference in low-power wireless networks. In *Proc. of IEEE IPSN*, 2016.

[38] Piotr Indyk. Explicit constructions for compressed sensing of sparse signals. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 30–33. Society for Industrial and Applied Mathematics, 2008.

[39] Sachin Katti, Shyamnath Gollakota, and Dina Katabi. Embracing wireless interference: Analog network coding. *Proc. of ACM SIGCOMM*, 2007.

[40] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. Xors in the air: Practical wireless network coding. In *Proc. of ACM SIGCOMM*, 2006.

[41] Song Min Kim and Tian He. Freebee: Cross-technology communication via free side-channel. In *Proc. of ACM MobiCom*, 2015.

[42] Linghe Kong and Xue Liu. mzig: Enabling multi-packet reception in zigbee. In *Proc. of ACM MobiCom*, 2015.

[43] Swarun Kumar, Diego Cifuentes, Shyamnath Gollakota, and Dina Katabi. Bringing cross-layer mimo to today's wireless lans. In *Proc. of ACM SIGCOMM*, 2013.

[44] Tianji Li, Mi Kyung Han, Apurv Bhartia, Lili Qiu, Eric Rozner, Yin Zhang, and Brad Zarikoff. Crma: Collision-resistant multiple access. In *Proc. of ACM MobiCom*, 2011.

[45] Zhijun Li and Tian He. Webee: Physical-layer cross-technology communication via emulation. In *Proc. of ACM MobiCom*, 2017.

[46] Kyle Miller, Atresh Sanne, Kannan Srinivasan, and Sriram Vishwanath. Enabling real-time interference alignment: Promises and challenges. In *Proc. of ACM MobiHoc*, 2012.

[47] Renato DC Monteiro and Ilan Adler. Interior path following primal-dual algorithms. part i: Linear programming. *Mathematical programming*, 44(1-3):27–41, 1989.

[48] Peshal Nayak, Michele Garetto, and Edward W Knightly. Multi-user downlink with single-user uplink can starve tcp. In *Proc. of IEEE INFOCOM*, 2017.

[49] Jonathan Perry, Peter A Iannucci, Kermin E Fleming, Hari Balakrishnan, and Devavrat Shah. Spinal codes. In *Proc. of ACM SIGCOMM*, 2012.

[50] Qualcomm. Qualcomm ipq4018 soc. `https://www.qualcomm.com/products/ipq4018`, 2019.

[51] Sudipta Saha and Mun Choon Chan. Design and application of a many-to-one communication protocol. In *Proc. of IEEE INFOCOM*, 2017.

[52] I Richard Savage. Probability inequalities of the tchebycheff type. *Journal of Research of the National Bureau of Standards-B. Mathematics and Mathematical Physics B*, 65:211–222, 1961.

[53] Souvik Sen, Naveen Santhapuri, Romit Roy Choudhury, and Srihari Nelakuditi. Successive interference cancellation: A back-of-the-envelope perspective. In *Proc. of ACM SIGCOMM*, 2010.

[54] Clayton Shepard, Hang Yu, Narendra Anand, Erran Li, Thomas Marzetta, Richard Yang, and Lin Zhong. Argos: Practical many-antenna base stations. In *Proc. of ACM MobiCom*, 2012.

[55] Zhenyu Song, Longfei Shangguan, and Kyle Jamieson. Wi-fi goes to town: Rapid picocell switching for wireless transit networks. In *Proc. of ACM SIGCOMM*, 2017.

[56] Karthikeyan Sundaresan, Mohammad Khojastepour, Eugene Chai, and Sampath Rangarajan. Full-duplex without strings: Enabling full-duplex with half-duplex clients. In *Proc. of ACM MobiCom*, 2014.

[57] Sanjib Sur, Ioannis Pefkianakis, Xinyu Zhang, and Kyu-Han Kim. Practical mu-mimo user selection on 802.11 ac commodity networks. In *Proc. of ACM MobiCom*, 2016.

[58] George Turin. An introduction to matched filters. *IRE transactions on Information Theory*, 6:311–329, 1960.

[59] Sergio Verdu et al. *Multiuser detection*. Cambridge university press, 1998.

[60] Andrew J Viterbi and Andrew J Viterbi. *CDMA: principles of spread spectrum communication*, volume 122. Addison-Wesley Reading, MA, 1995.

[61] Yin Wang, Yunhao Liu, Yuan He, Xiang-Yang Li, and Dapeng Cheng. Disco: Improving packet delivery via deliberate synchronized constructive interference. *IEEE Transactions on Parallel & Distributed Systems*, pages 713–723, 2015.

[62] Allen Y Yang, Zihan Zhou, Arvind Ganesh Balasubramanian, S Shankar Sastry, and Yi Ma. Fast l1-minimization algorithms for robust face recognition. *IEEE Transactions on Image Processing*, 22:3234–3246, 2013.

[63] Junmei Yao, Tao Xiong, and Wei Lou. Beyond the limit: A fast tag identification protocol for rfid systems. *Pervasive and Mobile Computing*, 21:1–18, 2015.

[64] Xinyu Zhang and Kang G Shin. Cooperative carrier signaling: Harmonizing coexisting wpan and wlan devices. *IEEE/ACM Transactions on Networking (TON)*, 21:426–439, 2013.

[65] Anfu Zhou, Teng Wei, Xinyu Zhang, Min Liu, and Zhongcheng Li. Signpost: Scalable mu-mimo signaling with zero csi feedback. In *Proc. of ACM MobiHoc*, 2015.

[66] Wenjie Zhou, Tanmoy Das, Lu Chen, Kannan Srinivasan, and Prasun Sinha. Basic: backbone-assisted successive interference cancellation. In *Proc. of ACM MobiCom*, 2016.

APPENDIX A

We analyze popular WiFi drivers and AP firmware to understand these practical constraints: (i) Linux kernel drivers: ath9k (Qualcomm/Atheros 802.11n chipsets), brcmsmac (Broadcom 802.11n chipsets), and iwlwifi (Intel 802.11n/ac chipsets); (ii) 802.11ac AP firmware in Linksys EA8500, EA9500, TP-Link AC1200, C5400, and AD7200.

In particular, ath9k allows the maximum length of a data payload to be either 8,192 or 65,535 bytes (aPSDUMaxLength for 802.11n is 65,535) based on its version number (as shown in Listing 1); brcmsmac uses the maximum duration of 5,000 μs (aPPDUMaxTime for 802.11n is 10,000 μs) (as shown in Listing 2); iwlwifi allows the maximum duration to be 4,000 μs (aPPDUMaxTime for 802.11ac is 5,484), as the excerpted code shown in Listing 3. In addition, Linksys EA8500, EA9500, and TP-Link AC1200, C5400 and AD7200 share the same code: the maximum length of a data payload is 65,535 bytes (aPSDUMaxLength for 802.11ac is 4,692,480)(as shown in Listing 4).

Listing 1: Source code in Qualcomm/Atheros ath9k (802.11n)

```
/* hw.h */
...
#define ATH_AMPDU_LIMIT_MAX     (64 * 1024 - 1)
...
/* hw.c */
    ...
    if (AR_SREV_9160_10_OR_LATER(ah) ||
        AR_SREV_9100(ah))
       pCap->rts_aggr_limit = ATH_AMPDU_LIMIT_MAX;
    else
       pCap->rts_aggr_limit = (8 * 1024);
    ...
```

Listing 2: Source code in Broadcom brcmsmac (802.11n)

```
/* ampdu.c */
...
/* max dur of tx ampdu (in msec) */
#define AMPDU_MAX_DUR       5
...
    ampdu->dur = AMPDU_MAX_DUR;
...
```

Listing 3: Source code of Intel iwlwifi (802.11ac)

```
/* mvm/constants.h */
...
#define IWL_MVM_RS_AGG_TIME_LIMIT       4000
...
/* mvm/rs.c */
...
    lq_cmd->agg_time_limit =
        cpu_to_le16(IWL_MVM_RS_AGG_TIME_LIMIT);
...
```

Listing 4: The same source code in Linksys EA8500/EA9500 and TP-Link AC1200/C5400/AD7200.

```
/* include/linux/ieee80211.h */
...
/*
Maximum length of AMPDU that the STA can receive.
Length = 2 ^ (13 + max_ampdu_length_exp)-1 (octets)
*/
enum ieee80211_max_ampdu_length_exp {
    ...
    IEEE80211_HT_MAX_AMPDU_64K = 3
};
...
```

APPENDIX B

Our comprehensive data collections capture WiFi packet traces under a diversity of traffic load conditions over long time periods. For each transmitter in the packet traces, we compute the NAV distribution in its RTS packets. We find that the NAV distribution is highly patterned or uneven in its value space. Figure 23 shows the NAV distributions of top five devices that send the largest numbers of RTS packets in different measurement environments. We observe from Figure 23 that each device's NAV values almost concentrate in the small value regions. For example, in the lab scenario, 92.1% NAV values in RTS packets from device 3 is 156.

We also notice that, interestingly, WiFi devices from the same manufacturer (identified by their MAC addresses) do exhibit similar NAV distribution in their RTS packets. Figure 24 illustrates the distributions of all NAV values in RTS packets transmitted by devices from 15 common manufacturers. It is seen from Figure 24 that the distributions exhibit different patterns by manufacturers, mainly due to their distinct firmware designs. Similar to Figure 23, Figure 24 shows the NAV distributions are highly uneven and patterned with a small number of NAV values much more likely to show up in RTS packets than the others.

In addition to NAV, we also measure the data rates of RTS packets. Figure 25 depicts the distribution of RTS dta rates in different environments. We can see that a large number of devices adopt 12 Mbps and 24 Mbps data rates to send RTS. Furthermore, in the conference and hotel scenarios, most devices even only use the 24 Mbps data rate.

Based on the packet trace analysis, if we select the NAV values that are most likely in RTS packet from each device to construct the comb matrix $\mathbf{M}$, we should be able to decrease the size of $\mathbf{M}$ at the cost of a small performance penalty.

APPENDIX C

Performance of γ-decimation: For the tooth vector set $\mathcal{M} = \{\mathbf{m}_i\}_{i\in[1,M]}$, where $M = |\mathcal{M}|$ and $\mathbf{m}_i \in \mathbb{C}^{L\times 1}$, γ-decimation selects M/γ tooth vectors with largest correlation values to form a new comb matrix. Denote by $\mathcal{M}'$ the set consisting of these selected M/γ tooth vectors by γ-decimation. Without loss of generality, we assume the collided signal $\mathbf{y}$ contains the first S tooth vectors, i.e., $\mathbf{m}_1, \cdots, \mathbf{m}_S$. In this section, notations are summarized as follows: (i) $o(1)$ denotes a function that converges to 0 as $L \to \infty$; (ii) $\mathsf{E}(\cdot)$ and $\mathsf{Var}(\cdot)$ denote the expectation and variance operators, respectively; (iii) for a complex number m, m^* is the complex conjugate of m, and $|m|$ is the magnitude of m. Now we state the following theorem to show the performance of γ-decimation:

Theorem 1 *Define event A as the event that* $\mathbf{m}_1$, $\mathbf{m}_2$, $\cdots$, $\mathbf{m}_s$ *are all selected by γ-decimation in* $\mathcal{M}'$*. Then, it holds that* $\mathsf{P}(A) = 1 - o(1)$ *(i.e., event A happens with high probability).*

Proof: We first normalize the correlation between comb matrix $\mathbf{M}$ and the received vector $\mathbf{y}$. From (2), we have

$$\begin{aligned}\mathbf{z} &= \frac{1}{L}\mathbf{M}^H\mathbf{y} \\ &= \frac{1}{L}\begin{bmatrix} \mathbf{m}_1^H\mathbf{m}_1 & \mathbf{m}_1^H\mathbf{m}_2 & \cdots & \mathbf{m}_1^H\mathbf{m}_M \\ \mathbf{m}_2^H\mathbf{m}_1 & \mathbf{m}_2^H\mathbf{m}_2 & \cdots & \mathbf{m}_2^H\mathbf{m}_M \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{m}_M^H\mathbf{m}_1 & \mathbf{m}_{M-1}^H\mathbf{m}_2 & \cdots & \mathbf{m}_M^H\mathbf{m}_M \end{bmatrix}\begin{bmatrix} g1 \\ g2 \\ \vdots \\ g_M \end{bmatrix}.\end{aligned} \tag{4}$$

Let $\mathbf{z} = [z_1, z_2, \cdots, z_M]^H$. It holds that $\forall z_i \in \mathbf{z}$,

$$z_i = \frac{1}{L}\sum_{s=1}^{M} g_s \mathbf{m}_i^H \mathbf{m}_s = \frac{1}{L}\sum_{s=1}^{M}\sum_{k=1}^{L} g_s m_{i,k}^* m_{s,k}. \tag{5}$$

Because each tooth vector $\mathbf{m}_i$ has the random property by coding, for any entry $m_{j,i} \in \mathbf{m}_i$, we have $\mathsf{E}(m_{j,i}) = 0$ and let $\mathsf{E}(|m_{j,i}|^2) = \sigma^2$.

Since tooth vectors $\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_S$ are the ones to be resolved, we know the first S members in $\mathbf{g}$ are not zeros. Define Y as

$$Y = \sum_{m=S+1}^{M} \mathbf{1}_{\{|z_m|>h\}}, \tag{6}$$

denoting the number of false alarms (i.e., noise exceeding the threshold), where h is a threshold, and $\mathbf{1}_{\{|z_m|>h\}}$ is the indicator function defined as

$$\mathbf{1}_{\{|z_m|>h\}} = \begin{cases} 1, & \text{if } |z_m| > h \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

To evaluate the performance of γ-decimation, we define another event

$$B = \left(\bigcap_{s=1}^{S}\{|z_s| > h\}\right)\bigcap\{Y \le (\gamma M - S)\},$$

where the first part denotes that the correlation values z_1, z_2, $\cdots, z_S$ are all above the given threshold h and the second part indicates that there are at most $(\gamma M - S)$ other correlation values above h. If event B happens, $\mathbf{m}_1$, $\mathbf{m}_2$, $\cdots$, $\mathbf{m}_S$ will be selected by γ-decimation and thus event A must happen. This means $\mathsf{P}(A|B) = 1$ and $\mathsf{P}(A) \ge \mathsf{P}(B)$. Therefore, according to Fréchet inequalities, we obtain

$$\begin{aligned}\mathsf{P}(A) &\ge \mathsf{P}\left(\left(\bigcap_{s=1}^{S}\{|z_s| > h\}\right)\bigcap\{Y \le (\gamma M - S)\}\right) \\ &\ge \mathsf{P}\left(\bigcap_{s=1}^{S}\{|z_s| > h\}\right) - \mathsf{P}\left(Y > (\gamma M - S)\right).\end{aligned} \tag{8}$$

From (8), we need two steps to finish the proof:

- **Step 1:** prove $\mathsf{P}\left(\bigcap_{s=1}^{S}\{|z_s| > h\}\right) = 1 - o(1)$.
- **Step 2:** prove $\mathsf{P}\left(Y > (\gamma M - S)\right) = o(1)$.

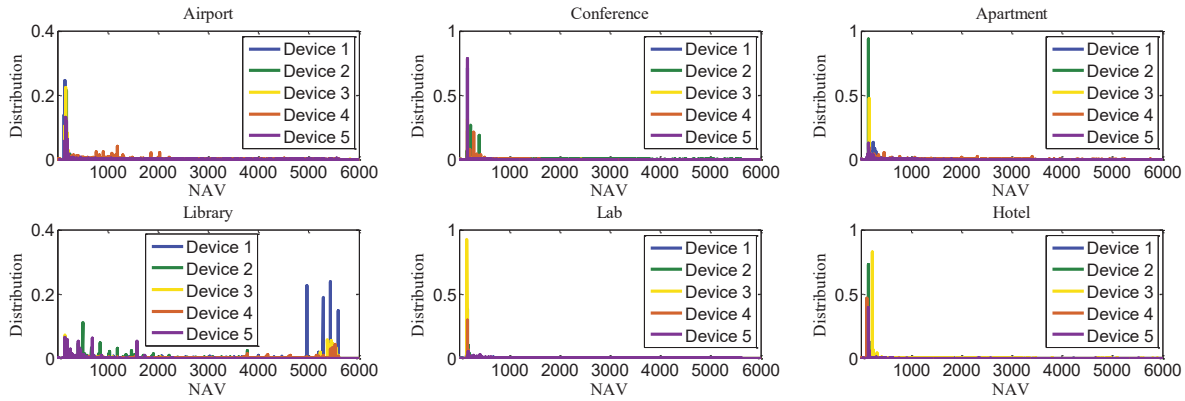

Figure 23: NAV distributions at different locations.

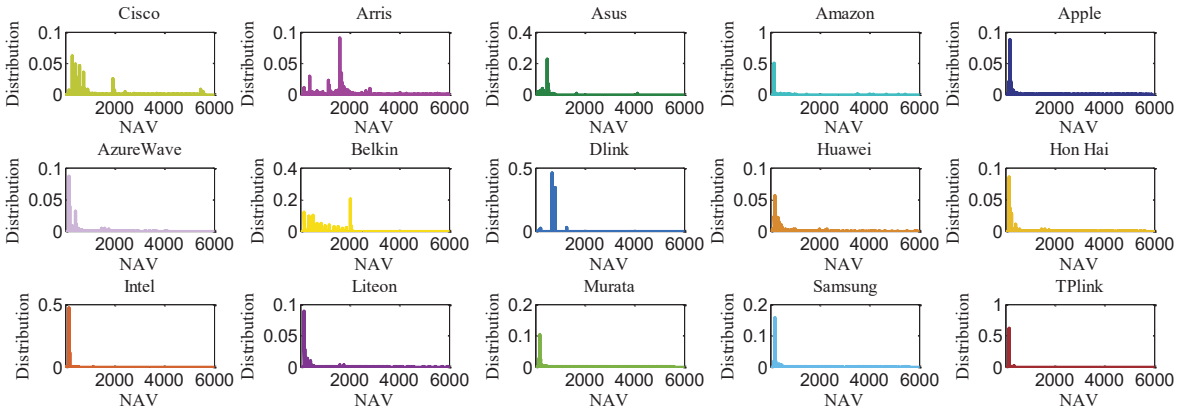

Figure 24: NAV distributions of different vendors.

Step 1:

By Fréchet inequalities, we have that

$$\mathsf{P}\left(\bigcap_{s=1}^{S}\{|z_s|>h\}\right)\geq\sum_{s=1}^{S}\mathsf{P}(|z_s|>h)-(S-1). \quad (9)$$

According to Cantelli's inequality [52], for $1\leq s\leq S$, we have the probability

$$\mathsf{P}(|z_s|>h)\geq 1-\frac{\mathsf{Var}(|z_s|)}{\mathsf{Var}(|z_s|)+(h-\mathsf{E}(|z_s|))^2}. \quad (10)$$

Next we derive $\mathsf{E}(|z_s|)$ and $\mathsf{Var}(|z_s^2|)$ respectively. Because $1\leq s\leq S$, without loss of generality, we consider the first element z_1. From (5), by leveraging Lemma 1, we have

$$\mathsf{E}(|z_1|)=\mathsf{E}(\frac{1}{L}g_1\mathbf{m}_1^H\mathbf{m}_1+\frac{1}{L}\sum_{s=2}^{S}g_s\mathbf{m}_1^H\mathbf{m}_s)=g_1\sigma^2, \quad (11)$$

and

$$\begin{aligned}\mathsf{E}(|z_1|^2)&=\mathsf{E}\left(\frac{1}{L^2}\left(\mathbf{m}_1^H\sum_{s=1}^{S}g_s\mathbf{m}_s\right)^2\right)\\&=\frac{1}{L^2}\mathsf{E}\left(\sum_{s=1}^{S}\sum_{k=1}^{L}g_s m_{1,k}^* m_{s,k}\sum_{s'=1}^{S}\sum_{k'=1}^{L}g_{s'}m_{1,k'}^* m_{s',k'}\right)\\&=\frac{1}{L^2}\sum_{s=1}^{S}\sum_{k=1}^{L}\sum_{s'=1}^{S}\left(\sum_{k'=1,k'\neq k}^{L}g_s g_{s'}\mathsf{E}(m_{1,k}^* m_{s,k})\right.\\&\quad\left.\times\mathsf{E}(m_{1,k'}^* m_{s',k'})+g_s g_{s'}\mathsf{E}(m_{1,k}^* m_{s,k}m_{1,k}^* m_{s',k})\right).\\&=g_1^2\sigma^4+\frac{1}{L}\Xi_1,\end{aligned} \quad (12)$$

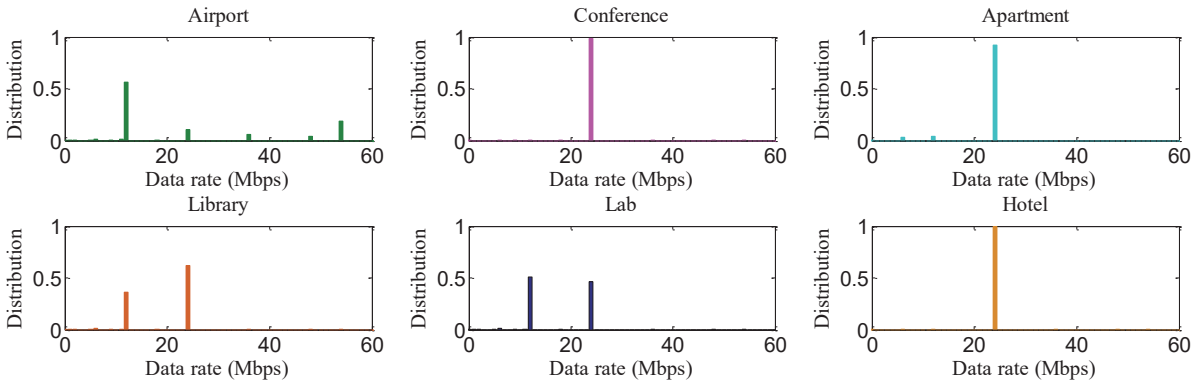

Figure 25: RTS data rate distributions.

where

$$\Xi_1 = \left(g_1^2(\mathsf{E}(|m_{1,k}|^4) - \sigma^4) + \mathsf{E}((m_{1,k}^*)^2)\mathsf{E}(m_{1,k}^2) \sum_{s=2}^{S} g_s^2 \right).$$

From (11) and (12), we obtain the variance of z_1 as

$$\mathsf{Var}(|z_1|) = \mathsf{E}(|z_1|^2) - (\mathsf{E}(|z_1|))^2 = \frac{1}{L}\Xi_1. \tag{13}$$

Replacing (13) into (10), we have

$$\mathsf{P}(|z_1| > h) \geq 1 - \frac{\Xi_1}{\Xi_1 + L(h - g_s\sigma^2)^2}. \tag{14}$$

Then, (9) can be rewritten as

$$\begin{aligned} \mathsf{P}\left(\bigcap_{s=1}^{S} \{|z_s| > h\} \right) &\geq 1 - \sum_{s=1}^{S} \frac{\Xi_s}{\Xi_s + L(h - g_s\sigma^2)^2} \\ &\geq 1 - S\frac{\Xi_{\max}}{\Xi_{\max} + L(h - g_{\max}\sigma^2)^2}, \end{aligned} \tag{15}$$

where $\Xi_{\max} = \max\{\Xi_s\}_{s\in[1,S]}$, and $g_{\max} = \max\{g_s\}_{s\in[1,S]}$. When $L \to \infty$, the probability converges to 1.

Step 2:

Letting $y = \gamma M - S$, by Markov's inequality, we have

$$\mathsf{P}(Y > y) \leq \frac{1}{y}\mathsf{E}(Y), \tag{16}$$

then from (6), we have

$$\mathsf{E}(Y) = \mathsf{E}\left(\sum_{m=S+1}^{M} \mathbf{1}_{\{|z_m|>h\}} \right) = \sum_{m=S+1}^{M} \mathsf{P}(|z_m| > h). \tag{17}$$

According to Chebyshev's inequality, we can obtain

$$\mathsf{P}(||z_m| - \mathsf{E}(|z_m|)| > h) \leq \frac{\mathsf{Var}(|z_m|)}{h^2}. \tag{18}$$

Next, we derive $\mathsf{E}(|z_m|)$ and $\mathsf{Var}(|z_m|)$. Without loss of generality, we consider the last element z_M. Similarly, we have

$$\mathsf{E}(|z_M|) = \frac{1}{L}\mathsf{E}\left(\sum_{s=1}^{S}\sum_{k=1}^{L} g_s m_{M,k}^* m_{s,k} \right) = 0, \tag{19}$$

and

$$\begin{aligned} \mathsf{E}(|z_M|^2) &= \frac{1}{L^2}\mathsf{E}\left(\sum_{s=1}^{S}\sum_{k=1}^{L} g_s m_{M,k}^* m_{s,k} \right)^2 \\ &= \frac{1}{L^2}\sum_{s=1}^{S}\sum_{k=1}^{L}\sum_{s'=1}^{S} g_s g_{s'} \mathsf{E}(m_{M,k}^* m_{s,k} m_{M,k}^* m_{s',k}) \\ &= \frac{1}{L}\sum_{s=1}^{S} g_s^2 \mathsf{E}((m_{M,k}^*)^2)\mathsf{E}(m_{s,k}^2). \end{aligned} \tag{20}$$

Let $\Psi_M = \sum_{s=1}^{S} g_s^2 \mathsf{E}((m_{M,k}^*)^2)\mathsf{E}(m_{s,k}^2)$, then we can obtain

$$\mathsf{Var}(|z_M|) = \mathsf{E}(|z_M|^2) - (\mathsf{E}(|z_M|))^2 = \frac{1}{L}\Psi_M. \tag{21}$$

Replacing (19) and (21) into (18), we have

$$\mathsf{P}(|z_M| > h) \leq \frac{\Psi_M}{Lh^2}. \tag{22}$$

Thus (17) can be rewritten as

$$\mathsf{E}(Y) \leq \sum_{m=S+1}^{M} \frac{\Psi_M}{Lh^2} \leq (M - S - 1)\frac{\Psi_{\max}}{Lh^2}, \tag{23}$$

where $\Psi_{\max} = \max\{\Psi_m\}_{m\in[S+1,M]}$. Finally,

$$\mathsf{P}(Y > y) \leq \frac{1}{y}\mathsf{E}(Y) \leq \frac{(M - S - 1)\Psi_{\max}}{Lyh^2}. \tag{24}$$

When $L \to \infty$, $\mathsf{P}(Y > y)$ converges to 0, which completes the proof. □

Lemma 1 *Given tooth vectors* $\mathbf{m}_i$ *and* $\mathbf{m}_j$*, for all* $m_{k,i} \in \mathbf{m}_i$ *and* $m_{s,j} \in \mathbf{m}_j$ *satisfying* $\mathsf{E}(m_{k,i}) = \mathsf{E}(m_{s,j}) = 0$ *and* $\mathsf{E}(|m_{k,i}|^2) = \mathsf{E}(|m_{s,j}|^2) = \sigma^2$*, the following two statements are true: (i) if* $i \neq j$*,* $\mathsf{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j) = 0$ *and* $\mathsf{E}(|\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j|^2) = \frac{1}{L}\sigma^4$*; (ii) if* $i = j$*,* $\mathsf{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_i) = \mathsf{E}(\frac{1}{L}\mathbf{m}_j^H\mathbf{m}_j) = \sigma^2$ *and* $\mathsf{E}(|\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_i|^2) = \frac{1}{L}(L-1)\sigma^4 + \frac{1}{L}3\sigma^4$.

Proof: Let $z = \frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j = \frac{1}{L}\sum_{k=1}^{L} m_{i,k}^* m_{j,k}$.

For statement (i), since $\mathsf{E}(m_{i,k}) = \mathsf{E}(m_{j,k}) = 0$, we have

$$\mathsf{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j) = 0.$$

Furthermore, as we know $\mathsf{E}(|m_{i,k}|^2) = \mathsf{E}(|m_{s,k}|^2) = \sigma^2$, we can obtain

$$\begin{aligned}\mathsf{E}(|\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j|^2) &= \frac{1}{L^2}\mathsf{E}\left(\sum_{k=1}^{L} m_{i,k}^* m_{j,k} \sum_{q=1}^{L} m_{i,q}^* m_{j,q}\right) \\ &= \frac{1}{L}\sigma^4\end{aligned} \tag{25}$$

For statement (ii), it is easy to know that

$$\mathsf{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j) = \mathsf{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_i) = \sigma^2$$

and

$$\begin{aligned}\mathsf{E}(|\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_i|^2) &= \mathsf{E}\left(\frac{1}{L}\sum_{k=1}^{L} m_{i,k}^* m_{i,k}\right)^2 \\ &= \frac{1}{L^2}L(L-1)\sigma^4 + \frac{1}{L^2}\sum_{k=1}^{L}\mathsf{E}(|m_{i,k}|^4) \\ &= \frac{1}{L}(L-1)\sigma^4 + \frac{1}{L}3\sigma^4.\end{aligned} \tag{26}$$

Therefore, we complete the proof. □