# Retransmission $\neq$ Repeat: Simple Retransmission Permutation Can Resolve Overlapping Channel Collisions

Li Erran Li[⋆] Kun Tan[†] Harish Viswanathan[⋆] Ying Xu[×] Yang Richard Yang[‡]

Bell Labs[⋆] Microsoft Research, Asia[†] Yale University[‡]
Beijing University of Post and Telecommunications[×]
{erranlli,harishv}@research.bell-labs.com kuntan@microsoft.com
yang.r.yang@yale.edu xuying.bupt@gmail.com

## ABSTRACT

Collisions in overlapping channels can be a major problem in the deployment of high-speed OFDM networks. In this paper, we present Remap, a simple, novel paradigm for handling collisions in overlapping OFDM channels. Remap introduces a novel concept of retransmission permutation that permutes the bit-to-subcarrier assignment after each transmission, departing from the traditional, simply-repeat paradigm. Remap is simple to implement and able to exploit collision-free subcarriers to decode frames despite successive collisions in overlapping channels. We apply Remap to 802.11g to demonstrate that the diversity created by remapped frames can substantially improve decoding efficiency and improve wireless throughput. We implement our technique in software radio and demonstrate that it has potential to be deployed with simple software and firmware updates.

## Categories and Subject Descriptors

C.2.1. [**Computer-Communication Networks**]: Network Architecture and Design – Wireless communication

## General Terms

Algorithms, Design, Experimentation.

## Keywords

Collision Decoding, Interference Cancellation, OFDM, Time-Frequency Decoding, Loss of Orthogonality.

## 1. INTRODUCTION

As OFDM becomes the foundation of modern high-speed wireless networks, due to its advantages [19] such as lower symbol rate, effective usage of a large frequency band, and resistance to frequency-selective fading, collisions on overlapping OFDM channels may become a major problem in the deployment of such high-speed wireless networks. Specifically, in an OFDM network, each channel is allocated a set of subcarriers, and two channels overlap when the intersection of their sets of subcarriers is not empty. Consider 802.11g, which is becoming almost ubiquitously deployed in many residential neighborhoods. With only 3 orthogonal channels with disjoint sets of subcarriers but a large number of access points in a densely populated neighborhood, it is inevitable that many 802.11 access points in range of each other use overlapping channels, as observed by previous measurement studies (*e.g.*, [1]). In [17], the authors show that partially overlapping channels may improve network throughput even in managed 802.11 networks, when the number of orthogonal channels is limited, thus supporting the deployment of partially overlapping channels. Channel overlapping is also allowed in WiFi networks built on digital white spaces [2].

A challenge of using overlapping OFDM channels, however, is how to handle collisions during contention and/or in the presence of hidden terminals. Although much progress has been made recently in handling collisions (*e.g.*, Zigzag decoding [7]), the previous schemes are for single-channel collisions, not the overlapping channel settings.

In this paper, we present Remap, a simple, novel paradigm for handling collisions in OFDM networks with overlapping channels. Remap is different from the existing, passively repeat paradigm, and introduces a novel concept called retransmission permutation to permute the bit-to-subcarrier mapping after each transmission. Retransmission permutation is a powerful diversity technique [21] that can recover frequency selective losses from subsequent retransmissions when there is no collision. When there are collisions, it in essence provides channel-width adaptation and allows bootstrapping of the decoding of collided frames that may otherwise be impossible to decode.

Specifically, the foundation of Remap is based on a simple observation and a simple idea. The observation is that when two frames transmitted on overlapping channels collide, only the subcarriers in the intersection of the two channels collide; the bits in other subcarriers are clean and can be collected. However, the non-colliding subcarriers do not contain complete frame information. The idea of Remap is to introduce structured permutation on the mapping from bits to subcarriers after each collision to create structured diversity. This diversity allows either independent decoding or bootstrapping other decoding techniques such as Zigzag decoding [7].

In particular, we design 802.11g/Remap, which applies Remap to 802.11g to demonstrate its effectiveness. We show that by using the diversity created by remapped frames, an 802.11g receiver $a$ can decode any frame $P_a$ after 4 collisions with other transmissions in *adjacent* channels; the number of collisions reduces to 2 if the


*MobiCom'10,* September 20–24, 2010, Chicago, Illinois, USA.

other transmissions are on *non-adjacent* channels. These numbers do not make any assumptions on the frames collided with $P_a$. If the frames collided with $P_a$ are the same (*i.e.*, both collisions are between $P_a$ and another frame $P_b$), Remap can bootstrap decoding both frames with the bits on the collision-free subcarriers. Decoding both frames at a single access point (AP) is important so that the combiner (used in systems such as [18]) behind the AP can make use of the reception diversity. In contrast, without Remap, Zigzag cannot decode both frames. Furthermore, 802.11g/Remap is backward compatible with 802.11 MAC. Thus, 802.11g/Remap has potential to be deployed with simple software and firmware updates to the existing 802.11 networks.

We implement 802.11g/Remap using a software radio testbed and conduct experiments to evaluate the performance of Remap to demonstrate the benefits of Remap. For example, for non-adjacent channel collisions, when the signal strength of the primary data is weak, for example between -6.8 dB to 2.7 dB, a normal decoder can decode only between 0% to 8.6% collisions, while Remap can decode 90.2% to 98.4%.

To summarize, we have made the following contributions:

- We propose a simple, novel concept of retransmission permutation to handle collisions on overlapping OFDM channels.
- We design 802.11g/Remap. We rigorously show that through the diversity created by remapped frames, an 802.11g receiver can substantially improve decoding efficiency and improve wireless throughput.
- We implement our technique in software radio and demonstrate its feasibility.

The rest of the paper is organized as follows. In Section 2, we present the basic idea. In Section 3, we cover our techniques to handle key issues. In Section 4, we present evaluation results. In Section 5, we discuss our limitations and clarify several relevant issues. Section 6 compares Remap with related work, and we conclude in Section 7. In the appendix, we discuss several technical details for completeness.

## 2. BASIC IDEA

**802.11g Primer**

We use 802.11g to illustrate our basic idea. In standard 802.11g, data bits are assigned to subcarriers. We denote the first group of 16 subcarrier frequencies of 802.11g as $G_1$, the next group as $G_2$, etc. Each 802.11g channel consists of 64 consecutive subcarrier frequencies. [1] Thus, the first channel $C_1$ of 802.11g consists of four groups: $G_1$, $G_2$, $G_3$, and $G_4$; channel $C_2$ consists of $G_2$, $G_3$, $G_4$, and $G_5$; channel $C_3$ consists of $G_3$, $G_4$, $G_5$, and $G_6$; channel $C_4$ consists of $G_4$, $G_5$, $G_6$, and $G_7$, etc. Note that $C_1$ overlaps with $C_1$, $C_2$, $C_3$, and $C_4$. We say that $C_2$ is an *adjacent overlapping channel* of $C_1$; $C_3$ and $C_4$ are *non-adjacent overlapping channels* of $C_1$. [2]

Assume that a sender uses channel $C_1$ to send a frame $P$ consisting of four bit blocks $A_1$, $A_2$, $A_3$, and $A_4$. Let the bit-to-subcarrier assignment be that $A_1 \rightarrow G_1$, $A_2 \rightarrow G_2$, $A_3 \rightarrow G_3$, and $A_4 \rightarrow G_4$. In other words, the bits in bit block $A_1$ are assigned to be carried by subcarrier group $G_1$, those in $A_2$ by $G_2$, $A_3$ by $G_3$, and $A_4$ by $G_4$. If

[1] Only 48 subcarriers carry data bits.

[2] Note that non-overlapping channels do not equal to orthogonal channels; due to imperfect filtering, guard bands are needed to achieve orthogonality. Our decoding technique is subject to adjacent channel interference.

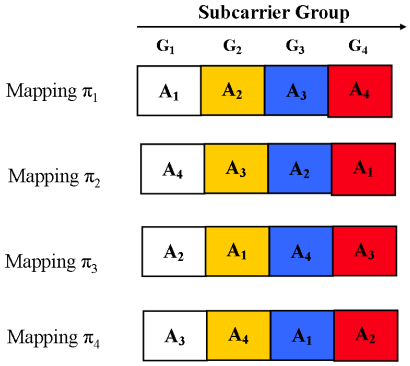



**Figure 1: Bit-to-subcarrier permutation table.**

the transmission for frame $P$ is not successful, 802.11g retransmits the frame $P$ where the bit-to-subcarrier assignment is the same.

**Retransmission Permutation**

The key novelty introduced by Remap is that during a retransmission, Remap uses a permutation scheduling of bit-to-subcarrier mapping. Note that, 802.11g interleaves in frequency to mitigate simultaneous fading of adjacent subcarriers. Remap is done after such interleaving. Since Remap is deterministic, the effectiveness of interleaving of 802.11g is not affected.

Figure 1 is a bit-to-subcarrier permutation table for 802.11g with desirable properties. Specifically, when transmitting a frame for the first time, the transmitter uses the first row of the table for bit-to-subcarrier mapping: bit blocks $A_1, \cdots, A_4$ are mapped to subcarrier groups $G_1,\cdots,G_4$ respectively. If the transmission encounters a collision and needs to be retried, the transmitter uses the second row of the permutation table for bit-to-subcarrier mapping: bit blocks $A_4$, $A_3$, $A_2$, $A_1$ are mapped to subcarrier groups $G_1,\cdots,G_4$ respectively. The third and fourth rows of the table are for the bit-to-subcarrier mapping of the second and third retransmissions. The transmitter cycles through these four rows if there is a need for more retransmissions.

With the basic idea, now we demonstrate the benefits of Remap using a simple example shown in Figure 2. In this example, two residential users, Alice and Bob, use 802.11g to connect to access points $AP_a$ and $AP_b$ respectively. Let the channel between Alice and $AP_a$ be $C_a$. The channel $C_b$ between Bob and $AP_b$ is an adjacent or non-adjacent overlapping channel of $C_a$.

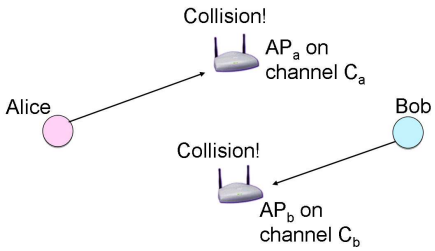



**Figure 2: $AP_a$ and $AP_b$ use overlapping channels. Frame from Alice to $AP_a$ and that of Bob to $AP_b$ collide.**

Due to hidden terminals or randomness, Alice may transmit a frame $P_a$ to $AP_a$ concurrently with Bob transmitting a frame $P_b$ to $AP_b$, causing collisions at $AP_a$ and $AP_b$. Without receiving an acknowledgment, Alice retransmits $P_a$, which may again collide with a transmission by Bob for frame $P'_b$. Note that $P'_b$ may be different from $P_b$, as Bob may schedule a different frame $P'_b$ after failing to receive an acknowledgment for $P_b$. Without Remap, frame $P_a$ cannot be decoded by the access point $AP_a$ so long there are collisions.

Remap, however, allows decoding of collided frames. To illustrate our idea, we consider how $AP_a$ decodes $P_a$. We will show that

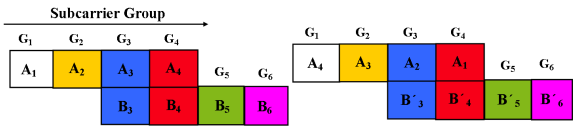



**Figure 3: Non-adjacent channel collisions.**

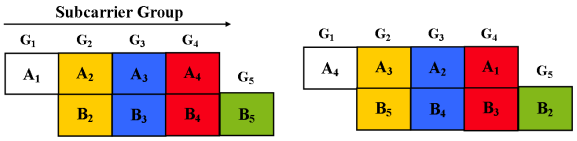



**Figure 4: Adjacent channel collisions.**

Remap allows $AP_a$ to decode $P_a$ after *at most* 3 retransmissions. We illustrate three cases.

- *Collisions in Non-adjacent Channels:* In this case, Alice and Bob transmit using overlapping but non-adjacent channels (*e.g.*, Alice uses $C_1$ and Bob uses $C_3$). Figure 3 illustrates the collided frames in the frequency domain. The left shows the first collision, and the right shows the second collision. After the first collision, $AP_a$ can decode bit blocks $A_1$ and $A_2$ because Alice encodes them in the collision-free subcarrier groups $G_1$ and $G_2$. When transmitting the frame for the second time, Alice uses Remap permutation to transmit $A_4$ and $A_3$ in the two collision-free subcarrier groups. Thus, despite collisions in both transmissions, $AP_a$ can decode all four bit blocks. In other words, Remap allows full decoding after at most one retransmission.
- *Collisions in Adjacent Channels:* The worst-case requiring 3 retransmissions happens when Alice and Bob use adjacent channels (*e.g.*, Alice uses $C_1$ and Bob uses $C_2$). Figure 4 shows the first two collisions. After the first collision, $AP_a$ can decode bit block $A_1$; after the second collision, $AP_a$ can decode bit block $A_4$ due to Remap permutation. Using the permutation table in Figure 1, one can verify that after the third collision, $AP_a$ can decode bit block $A_2$, and after the fourth collision, $AP_a$ can decode bit block $A_3$. Thus, Remap allows $AP_a$ to fully recover the frame despite persistent collisions.
- *Same-Pair Collisions in Adjacent Channels:* The preceding worst case does not make any assumption on the frames that collided with $P_a$. When the two frames that collided with $P_a$ are the same, Remap can achieve higher efficiency. We refer to this case as *same-pair collisions in adjacent channels* or *same-pair collisions* for short.

  Specifically, for the case of same-pair collisions, Remap uses bits that are mapped to collision-free subcarriers to bootstrap more advanced decoding motivated by the ZigZag decoding. Again consider Figure 4. We observe that bit blocks $A_1$ and $A_4$ are on collision-free subcarriers during the first and second collisions respectively. Let $y$ denote the time-domain signal from the first collision, and $y'$ from the second collision. Let $A_j(G_i)$ denote the time-domain signal of bit block $A_j$ encoded at subcarrier group $G_i$.

  From the first collision signal $y$, $AP_a$ decodes $A_1$ from subcarrier group $G_1$, since there is no collision on this subcarrier group. $AP_a$ then re-encodes $A_1$ onto subcarrier group $G_4$ to obtain $A_1(G_4)$. Subtracting $A_1(G_4)$ from $y'(G_4)$ (the signal at subcarrier group $G_4$ from the second collision), $AP_a$ obtains $B_3(G_4)$ and decodes $B_3$. $AP_a$ then re-encodes $B_3$ onto subcarrier group $G_3$ to obtain $B_3(G_3)$, subtracts $B_3(G_3)$ from $y(G_3)$ (the signal at subcarrier $G_3$ of the first collision), and then decodes $A_3$. With $A_3$, $AP_a$ decodes $B_5$ from the second collision.

  Similar to the preceding, from the second collision, $AP_a$ decodes $A_4$ from subcarrier group $G_1$, since there is no collision on this subcarrier group during collisions. $AP_a$ uses $A_4$ to decode $B_4$ from the signal of the first collision. $B_4$ causes $A_2$ to get decoded from the second collision and followed by $B_2$. Thus, $AP_a$ can decode all bits in all subcarriers, even the bits of the second frame transmitted in subcarriers outside of channel $C_a$ in the first transmission!

  We represent the decoding process using the decoding graph shown in Figure 5. Note that, for simplicity, we have presented a simplified view of 802.11g subcarrier structure. For 802.11g, there are unused subcarriers, so we need to define subcarrier groups in terms of used subcarrier groups. In this definition, the subcarrier groups are not aligned. However, this does not present any decoding problem. We give more details on 802.11g in the Appendix.

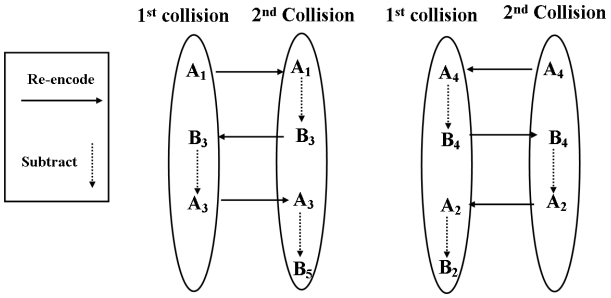



**Figure 5: Same-pair decoding graph.**

# 3. REMAP IN DEPTH

The preceding section presents the basic idea. In this section, we give more details on several key issues, and summarize with the complete algorithm and analysis. For additional technical details such as handling scramblers, 802.11 specific subcarrier structures, and channel estimation, please see Appendix.

## 3.1 Presentation Setting

We use 802.11g to be concrete. Our presentation setting is that an 802.11g transmitter Alice transmits frames (denoted $P_a$, $P'_a$) to a receiver $AP_a$. We refer to the frames from Alice as the primary frames. We refer to Alice' transmission channel as the primary channel. We assume that the primary channel is $C_1$ using subcarrier groups $G_1$ to $G_4$. A second transmitter Bob transmits other frames (denoted $P_b$ and $P'_b$) to another receiver concurrently, and may cause collisions with the transmissions by Alice. We refer to the channel used by Bob as the secondary channel, and the frames from Bob as the secondary frames. We focus on the case of one secondary channel on $C_2$, $C_3$ or $C_4$. Our scheme can be extended to other secondary channels.

## 3.2 How to Encode Bit-to-Subcarrier Remapping?

Remap may dynamically change the bit-to-subcarrier mapping for retransmissions. Therefore, it is necessary to notify such mapping information to the receiver before it can correctly decode the frame. One simple way is to encode such mapping information in the PHY frame header. However, there are two issues need to be considered. First, in the current 802.11 standards, the PLCP

header is compact, and there are no reserved bits for additional information. One possible solution is to extend the header by another OFDM symbol. But this will add additional overhead. Second, the PLCP header may also subject to collisions. Therefore, it requires a more robust way to encode the remapping information.

In this paper, we propose to encode the remapping information in the long training symbols of 802.11. Specifically, we design four distinct, pseudo-random (PN) long training symbols to signal the four remapping schemes. The new PN long training symbols are designed for good self-correlation. Following the preamble format of 802.11, Remap repeats a PN long training symbol twice to facilitate fine-grained frequency offset estimation and channel estimation. Note that changing the content of the long training symbol does not affect its ability for channel estimation, as long as it is known.

During the deployment in a WLAN, a Remap-capable AP announces that it supports Remap in its beacon frames. The AP can use the reserved bits in the CAPABILITY field of its beacon frame or the vendor specific information element of the beacon frame. The Remap scheme is utilized only when both the AP and the client are Remap-capable; otherwise, the conventional 802.11 format is used for backward capability.

During reception, the receiving AP will first use the 10 short symbols (which we kept intact) to conduct timing acquisition and coarse frequency offset estimation. Then the AP will conduct a search to determine whether the frame is a legacy frame or a Remap frame. In particular, for a Remap frame, the AP uses correlation to search for each of the four long PN training symbols in parallel. It will lock onto the one with the maximal peak and decide the remapping scheme.

## 3.3 Is There a Collision?

A collision is detected if a receiver finds the preamble of one frame inside another frame. One particular challenge here is that the interfering transmission can come from a partially overlapped channel, so that the traditional signal cross-correlation technique will fail.

To understand the reason, assume that a transmission of frame $P_b$ on channel $C_2$ collides with a transmission of frame $P_a$ on channel $C_1$. Assume that the preamble used for $P_b$ is $s_b$.[3] Since frame $P_b$ is transmitted on channel $C_2$, all samples taken on channel $C_1$ will rotate by $e^{j2\pi i\delta fT}$, where $\delta f$ is the difference between the central frequencies of channels $C_1$ and $C_2$, $T$ is the sampling period, and $i$ is the index of samples. The correlation result will be

$$\Gamma(\Delta) = \sum_{k=0}^{L} H_2 |s_b[k]|^2 e^{2jk\pi\delta fT},$$

where $H_2$ is the the channel gain measured on channel $C_2$. This may not be a peak even though the signal and the correlation pattern are matched. Thus, Remap should compensate this frequency difference before it can apply the correlation algorithm.

Another issue is that a signal of $P_b$ transmitted on channel $C_2$ can be distorted when received at $AP_a$, which is tuned to channel $C_1$ and thus will apply the matching filter for channel $C_1$ during reception. Specifically, the frequency components of the signal beyond the high edge of channel $C_1$ are filtered out. Thus, if Remap continues using $s_b$ as the template, it will also result in a reduced correlation peak.

To address the preceding issues, $AP_a$ precomputes templates to be used to detect collisions in overlapping channels. A template is the correlation samples used to detect the collision caused by a transmission on an overlapping channel $C_2$ with a given preamble. When computing the template, $AP_a$ applies a filter to remove the high frequency components. Then, $AP_a$ also compensates each sample in the template by a factor of $e^{-2jk\pi\delta fT}$ to remove the effects of the central frequency difference.

The detection process conducts a simultaneous search of collisions using the precomputed templates. A transmission is detected when a peak is found during correlation, and the signal energy is larger than a threshold. Remap will report a collision when the following two cases occur: 1) the receiver detects a preamble on the primary channel, and then it finds a preamble on a secondary channel while the transmission on the primary channel is still going on (*i.e.*, the energy level is still high); or 2) the receiver detects a preamble on a secondary channel, and then it finds a preamble on the primary channel while the transmission on the secondary channel is still going on.

## 3.4 Which Channel is the Secondary Channel?

After $AP_a$ detects a collision, it also needs to determine the channel that Bob uses. We identify two approaches.

One approach is that $AP_a$ uses a standard energy-detection technique. Specifically, each 802.11g channel (20 MHz) consists of 64 subcarriers. As we have already discussed, these subcarriers are partitioned into 4 groups. The energy-detection technique identifies each group that has a significant change in energy before and after the correlation peak. From the subcarrier groups that have experienced collisions, $AP_a$ can infer the secondary channel by utilizing the channel structure of 802.11g.

Specifically, $AP_a$ first conducts FFT on the input samples. Let $Y[k,m]$ be the output of the FFT for the $m$-th subcarrier symbol of the $k$-th OFDM symbol. Let the sample index of the correlation peak be $\Delta$. $AP_a$ then takes $L$ symbols ($80L$ samples) before the correlation peak and computes the energy of each subcarrier group $G_i, i = 1,2,3,4$ before the peak:

$$E_1(G_i) = \sum_{k=\Delta-L}^{\Delta-1} \sum_{m\in G_i} |Y[k,m]|^2. \quad (1)$$

$AP_a$ also uses $L$ symbols after the correlation peak to compute the energy of each subcarrier group $G_i, i = 1,2,3,4$:

$$E_2(G_i) = \sum_{k=\Delta}^{\Delta+L} \sum_{m\in G_i} |Y[k,m]|^2. \quad (2)$$

$AP_a$ compares $E_1(G_i)$ and $E_2(G_i)$. If $E_2(G_i)$ is greater than $E_1(G_i)$ by a threshold percentage, $AP_a$ decides that a collision happens on subcarrier group $G_i$.

The preceding energy-detection approach is effective only when the secondary frame has a comparable strength to the primary frame; it may not be accurate if the secondary frame has a significantly smaller signal strength (but still strong enough to cause a collision). To address this, we use another approach that again utilizes the correlation of different preamble templates on each channel. As explained earlier in Section 3.3, because preambles are transmitted on different channels, they will generate different patterns even though they contain the same signal. Thus, only the pattern with the matched channel will have the highest peak. This provides us a more reliable means to discern the interference channel.

## 3.5 How to Identify Same-Pair Collisions?

Once $AP_a$ determines that the received signal $y$ is the result of a collision in an adjacent channel, it tries to search for a signal $y'$

[3] As mentioned in Section 3.2, Remap conducts correlations for four possible preambles simultaneously.

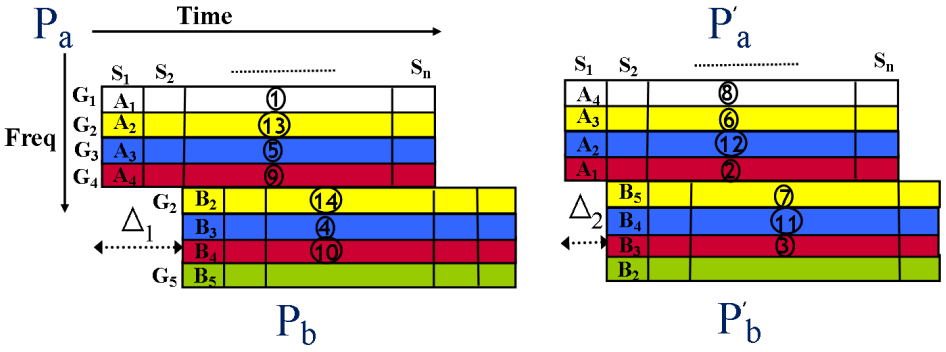



**Figure 6: Time and frequency domain decoding process of same-pair collisions.**

involving the same-pair of frames $P_a$ and $P_b$. Because senders in 802.11 typically keep retransmitting a failed transmission as soon as the medium is free, it is sufficient to store the signals of only a few most recent collisions.

$AP_a$ again uses correlation to do the search. Assume that $y$ contains the collision of frames $(P_a, P_b)$, and $y'$ the collision of $(P'_a, P'_b)$. Although $AP_a$ cannot decode the frames yet, it knows the channels that each of these four frames are transmitted on and the timing offsets. Because Remap maps bits to different subcarriers during retransmissions, $AP_a$ cannot simply perform a correlation in the time domain.

First, consider whether the frames on the primary channel are the same (*i.e.*, $P_a = P'_a$). $AP_a$ performs FFT on $y$ and $y'$ using the timing of Alice's frame. $AP_a$ also corrects frequency offsets. Let $M$ be the number of subcarriers and $N$ the number of symbols in each frame. Recall that $Y[k,m]$ is the output of the FFT (FFT of $y$) for the $m$-th subcarrier symbol of the $k$-th OFDM symbol. Similarly, $Y'[k,m]$ is the counterpart for $y'$. Let $Y^*[k,m]$ denote the complex conjugate of $Y[k,m]$. Let $H_a[m]$ and $H'_a[m]$ be the channel gains associated with the $m$-th subcarrier of Alice's channel to $AP_a$. Similarly, $H'_b[m]$ and $H_b[m]$ are for Bob. Let $\hat{Y}'[k,m]$ be the reverse-mapped samples; that is, if subcarrier $m$ is mapped to subcarrier $i$ in $P'_a$, then $\hat{Y}'[k,m] = H'_a[m]Y'[k,i]/H'_a[i]$. Let the transmitted bits on subcarrier $m$ for symbol $k$ from Alice be $X_a[k,m]$.

Then $AP_a$ computes the following correlation:

$$\begin{aligned}\Gamma(P_a, P'_a) &= \Sigma_{k=1}^{N}\Sigma_{m=1}^{M} Y^*[k,m]\hat{Y}'[k,m] \\ &= \Sigma_{k=1}^{N}\Sigma_{m=1}^{M} H^*{}_b[m]H'_a[m]|X_a[k,m]|^2. \end{aligned} \quad (3)$$

Note that, the preceding technique assumes that the channel of each subcarrier for the retransmitted frame does not change much between the two transmissions. Note that, if $AP_a$ does not know the bit-to-subcarrier mapping of $P'_a$, it may need to try out all four mappings.

Next consider whether the two frames on the second channel are the same (*i.e.*, $P_b = P'_b$). For the second channel, $AP_a$ cannot take all subcarrier groups into account, since subcarriers outside the primary channel are filtered. In this case, $AP_a$ conducts correlation similar to the preceding but with only a subset of the subcarrier groups.

## 3.6 How to Decode Same-Pair Collisions?

If signals $y$ and $y'$ are due to collisions of the same pair of frames on adjacent channels, Remap allows decoding of both, as we have seen in the preceding section on decoding same-pair collisions. However, the description in the preceding section focuses on the main ideas using a frequency-domain representation. It leaves multiple important issues unaddressed. We now give more details on how Remap addresses the issues using simple, nice techniques.

We start with a more detailed time and frequency domain representation. Figure 6 is a more detailed representation of the example in Section 2 to illustrate how to decode same-pair collisions. Recall that the collision-free subcarrier group for $AP_a$ is $G_1$. Thus, $AP_a$ can decode bit blocks $A_1$ from the first collision and $A_4$ from the second collision.

As we discussed in the preceding section, after decoding $A_1$ encoded on $G_1$ from the first collision signal, $AP_a$ re-encodes $A_1$ to subcarrier group $G_4$ to obtain $A_1(G_4)$, and subtracts $A_1(G_4)$ from $y'$ (the signal of the second collision) starting at offset $\Delta_2$ in order to recover the signal of $B_3$.

A key issue in the preceding subtraction step is that $AP_a$ needs to compute $A_1(G_4)$ in the time-domain. Remap computes $A_1(G_4)$ from $A_1(G_1)$ efficiently. In particular, $A_1(G_1)$ in the frequency domain consists of a group of subcarrier symbols, one on each subcarrier in $G_1$. If a symbol is carried by subcarrier $i$ in the first transmission of $P_a$, the symbol will be carried by subcarrier $F_i$ in the second transmission of $P_a$. Thus, re-encoding $A_1$, in the frequency domain, involves simple permutations and channel scaling of subcarrier symbols. After this operation, $AP_a$ conducts efficient IFFT to convert from the frequency domain to the time domain to obtain $A_1(G_4)$.

At this point, in the preceding section, we say that $AP_a$ decodes $B_3$ from the signal obtained by $y'(G_4) - A_1(G_4)$, re-encodes $B_3$ to subcarrier group $G_3$ to obtain $B_3(G_3)$, and then subtracts the (time-domain) $B_3(G_3)$ in order to recover $A_3$. However, this is not straightforward. In particular, although $AP_a$ obtains the time-domain signal of $B_3(G_4)$, it does not have the PLCP header of $P'_b$. Thus, $AP_a$ does not know the modulation scheme of $P'_b$ and consequently cannot actually decode $B_3$ from $B_3(G_4)$. *Remap solves this problem nicely*. In particular, if $AP_a$ can validate that the channel has not changed much between the two collisions, which is likely given that 802.11 is used primarily in indoor environments, it can compute the re-encoded time-domain signal of $B_3(G_3)$ from $B_3(G_4)$, using the technique from the preceding paragraph, without the need to fully decode $B_3(G_4)$. For better accuracy, once $AP_a$ obtains the PLCP header from the first pass, it can then apply the normal Remap decoding process. For subsequent bits of the second frame, it decodes using the modulation scheme encoded in its PLCP header's signaling field.

The preceding process continues, in order as labeled in the figure. After decoding the bits on all subcarriers, $AP_a$ proceeds to decode OFDM symbol by symbol, and reconstruct $P_a$. If needed, it can also reconstruct $P_b$. Note that, $AP_a$ does not have to wait until all bits on all subcarriers before decoding OFDM symbols.

```
Remap_decode(col_set, y, channel) // col_set is the set of collisions
01. col_channel = Get_collided_channel(y)
02. if (col_channel == channel ) // co-channel collision
03.   Zigzag_decode(col_set, y) // account for bits-to-subcarrier remapping
04. else
05.   collect_collision_free_group_bits(y, channel, col_channel)
06.   foreach y' ∈ col_set
07.     if (same_pair_collision(y, y', channel, col_channel))
08.         decode_frame_pair(y, y', channel, col_channel)
09.     elseif (same_primary_frame(y, y', channel, col_channel))
10.         collect_bits_in_same_frame(y, y', channel, col_channel)
11.     endif
12.   end foreach
13. endif
14. col_set = col_set ∪ y
```

**Figure 7: Remap decoding algorithm.**

It can decode an OFDM symbol as soon as all of its bits in all corresponding subcarriers are decoded. Thus, Remap can decode OFDM symbols as time progresses.

## 3.7 Complete Algorithm

We now give the complete Remap decoding algorithm. The algorithm is shown in Figure 7 as *Remap_decode*. The algorithm takes as input a new collision signal (*y*), the set (*col_set*) of saved collisions, and the channel (*channel*) that $AP_a$ tunes to.

The algorithm first determines whether the collision in the received signal *y* is co-channel or overlapping channel. The algorithm (line 1) computes the overlapped channel on which the interference transmission occurs. If the collision happens on the same channel of $AP_a$, it is co-channel, and the algorithm (line 4) invokes the Zigzag decoder (modified to deal with bit-to-subcarrier remapping). Otherwise, the collision is on overlapping channels.

If the collision in *y* is due to overlapping channels, the algorithm (line 5) first decodes and collects bit blocks in the collision-free subcarrier groups. For data structure simplicity, the collected bit blocks are still saved in *y*.

The algorithm then iterates over the set of saved collisions *col_set*. For each $y'$ in *col_set*, the algorithm detects whether *y* and $y'$ contain the collisions of the same-pair of frames. If so, the algorithm (line 8) decodes the pair of frames and adds them to the set of decoded frames. Otherwise, the algorithm (line 9) determines whether $y'$ and *y* contain the same frame on the primary channel. If so, the bit blocks decoded from their collision-free subcarriers belong to the same frame and can be combined. If all four bit blocks of a frame are collected and the combined frame passes the CRC check, the frame is inserted into the set of decoded frames.

## 3.8 Analysis and Extensions

We characterize the effectiveness of Remap under same-pair collisions. Specifically, let $AP_a$ be the intended receiver of frame $P_a$ from Alice on channel $C_a$; $AP_b$ the intended receiver of $P_b$ from Bob on channel $C_b$. Assume that Alice uses bit-to-subcarrier mappings $\pi_{a,i}$ and $\pi_{a,i'}$ for the first and second transmissions of $P_a$ on channel $C_a$ respectively. Bob uses mappings $\pi_{b,j}$ and $\pi_{b,j'}$ for the two transmissions of $P_b$ on $C_a$'s adjacent channel $C_b$. If the mappings are chosen from the Remap table shown in Figure 1, we have the following theorem:

THEOREM 1. *If $\pi_{a,i} \neq \pi_{a,i'}$ and $\pi_{b,j} \neq \pi_{b,j'}$, then Remap can decode both frames $P_a$ and $P_b$ at $AP_a$ or $AP_b$.*

PROOF. Due to the symmetry of the four mappings, w.l.o.g., we assume that during the first collision, Alice uses $\pi_{a,1}$ and Bob uses $\pi_{b,1}$. We can identify a total of 9 cases where Figure 4 is one example. One can verify that 6 out of the 9 cases can be decoded solely in the frequency domain iteratively, while 3 out of the 9 cases can be decoded by decoding in the time domain iteratively for a subset of subcarrier groups. □

*Remark* We can extend the bit-to-subcarrier mapping table in Figure 1 to non-802.11 channel structures where each channel consists of *k* adjacent subcarrier groups (in the case of 802.11g, $k = 4$).

## 3.9 Loss of Orthogonality

When $AP_a$ tries to decode Alice's frame on the primary channel, it may lose orthogonality for the following two reasons:

- Frequency offset of the second frame; and
- If symbols are misaligned between Alice and Bob's frames, during FFT, the energy of Bob's frame on subcarrier *i* may spread to Alice's subcarriers (not limited to subcarrier *i*), as Alice's FFT window will not contain complete OFDM symbols of Bob's frame.

Because of loss of orthogonality, $Y[i] = HX[i] + ICI[i] + w[i]$, where $ICI[i]$ is the interference effect of the aforementioned reasons, $H$ the channel gain, $X$ the data, and $w$ the noise.

We first look at the impact of misalignment between Alice's and Bob's symbols. Let Alice's *i*-th subcarrier signal be $x_i(t) = e^{2\pi it/T}$. Let the interfering subcarrier from Bob be $i+m$ and its signal is $x_i(t) = e^{2\pi(i+m)t/T}$. Suppose the symbols are unaligned by $\delta$. The correlation is then:

$$I_m(\delta, T) = \int_{\delta}^{T_N} x_i(t) \times x_{i+m}^*(t+\delta)dt \quad (4)$$

$$= \beta/\alpha(1 - e^{\alpha\delta}), \quad (5)$$

where $\beta = e^{-j2\pi(i+m)\delta/T}$ and $\alpha = -j2\pi m/T$.

The power is $|I_m(\delta,T)|^2 = (\frac{T}{2\pi m})^2(2 - 2cos(\frac{2\pi m\delta}{T}))$. Taking the max, we have the total interference power from the $i+m$-th subcarrier [6]:

$$|I_m|^2 = \frac{T^2}{(\pi m)^2}. \quad (6)$$

We obtain the sum of interferences from all subcarriers (recall that *M* is the total number of subcarriers) that are at least distance *k* from *i*:

$$I_{|m'-i|\geq k}(\delta, T) = \sum_{m'=0,|m'-i|\geq k}^{M} I_{|m'-i|}(\delta, T). \quad (7)$$

Assume that the interfering symbols at different subcarriers are independent. We have

$$|I_{|m'-i|\geq k}(\delta, T)|^2 = \sum_{m'=0,|m'-i|\geq k}^{M} |I_{|m'-i|}(\delta, T)|^2. \quad (8)$$

We then get the upper bound $|I^{max}_{|m'-i|\geq k}|^2 = \sum_{m'=0,|m'-i|\geq k}^{M} |I_{|m'-i|}|^2$. If $k = 4$, $I = 0.056T^2$ which 12.5 dB below the *i*-th signal power. If $k = 3$, then $I$ is 11.3 dB below the *i*-th signal power. Note that these interference powers are upper bounds. Because $A_1$ of the first collision and $A_4$ of the second collision are separated by 4 subcarriers, $A_1$ and $A_4$ can be decoded as long as the signal can tolerate a noise level that is 12.5 dB lower than the signal power. There are a few subcarriers of $A_2$ and $A_3$ that get interfered by nearby subcarriers with a distance smaller than 3. The following iterative solution should help.

After Remap decodes $P_a$ and $P_b$ with loss-of-orthogonality interference, we can reconstruct $ICI[i]$. We then subtract $ICI[i]$ to better decode $X[i]$ ($P_a$). This can be done similarly for $P_b$. In theory, this process can iterate. The quality of this iterative interference cancellation technique will depend on the decoding quality of the first step. We do not investigate these kinds of techniques in the experiments of this paper.

# 4. DISCUSSIONS

## 4.1 Limitations

**Rate adaptation**: Remap will not work if retransmission uses a different rate. This is also true for Zigzag. Some current rate adaptation algorithms adapt to collisions. For example, recent versions of madwifi [16] change the bit rate when transmissions are not acknowledged. However, as a recent study shows [25], rate should be adapted to noise rather than interference caused by other transmissions. Thus, a rate adaptation scheme like this will not cause problem to Remap or Zigzag.

**Asymmetric received power**: Remap is sensitive to asymmetric received powers of the collided frames. If one is significantly stronger than the other, then automatic gain control might lock to one and push the other into saturation at the ADCs or down into the quantization noise. That is, depending on the SNR difference, to decode the collisions, Remap can require a better ADC scheme.

## 4.2 Clarifications

**Pilot tone corruption**: It is possible that pilot tones might be corrupted. However, Remap does not use the corrupted pilot tones for channel estimation and frequency offset estimation. In the iterative decoding process, Remap uses pilot tones with interference subtracted out. In more detail, Remap first uses pilot tones of collision free subcarriers. In the iterative decoding process, once Remap subtracts the interference from collision free subcarriers, it is left with the next subcarrier group with interference removed. Hence, Remap decoding will have uncorrupted pilot tones in the next subcarrier group.

**Different modulations on different subcarriers**: Some recent studies, such as FARA [20], have proposed using different modulations on different subcarriers. Remap can work with these schemes as long as it knows the modulation scheme that each subcarrier uses.

**Different frame sizes**: Remap has no problem with frames of different sizes as long as the retransmission of the same frame uses the same data rate (but two frames can use different rates).

**Rateless erasure code**: Rateless measurement code (*e.g.*, [22]) cannot solve the problem that Remap solves. The problem with rateless erasure code is that, Remap cannot match the second transmitted frame with the first frame (*e.g.*, if the header of the second transmission collides with the other frame). If Remap cannot match the frames, it cannot put the relevant bits together.

**What gets permuted?** Only the preamble is not subject to bit-to-subcarrier remapping. Thus, the Remap iterative decoding process will take care of MAC header collisions. Since the preamble is known, it can be subtracted.

# 5. EVALUATIONS

## 5.1 Experimental Setup

We evaluate Remap using the Sora software radio platform [14].

- Hardware and software environment. Sora has a full-fledged implementation of 802.11g. Remap follows the standard 802.11g preamble structure, but we apply new long training symbols for remapped retransmissions. We have implemented Remap decoder for both non-same-pair collisions and same-pair collisions.

- Modulation. Remap can work with a variety of modulation schemes. In this section, we may focus on Binary Phase Shift Keying (BPSK). BPSK is used for the 6 Mbps data rate of 802.11g.

- Configuration parameters. We conduct experiments inside a large lab room in an office building. We choose to conduct our experiments in weekends or deep nights. So there are rare background 802.11b/g activities. We obtain different signal noise ratios (SNR) by changing the location and/or power. Unless otherwise mentioned, we use frames with 800 bytes in our experiments.

We set up one Sora as the receiver. Two other Sora machines are used as Alice and Bob. We set Alice to transmit on channel 3 and the receiver is also receiving on channel 3 as well. Bob can transmit on either channel 4 or channel 5.

We use BER (Bit Error Rate) and *normalized throughput* as performance metrics. BER is computed before the Viterbi channel decoder. Since we know the exact content of each transmitted frame, we can easily compute BER by a side-by-side comparison between the demodulated bits and the bits that have been transmitted. To compute the *normalized throughput*, we perform Viterbi decoding on each received frame. If a frame can be decoded by Viterbi decoder without any bit error, we count it a successfully decoded frame. Otherwise, it is counted as an error. We compute the *normalized throughput* as the ratio between all successfully decoded frames and the ideal case where all collisions can be decoded.

In our experiments, we first set all nodes on channel 3 and ensure both links, from Alice/Bob to the receiver, have good quality; that is, the frames loss rate is close to zero if there is only one transmitter being active. Then, we set Bob to channel 4 or 5.

## 5.2 Experimental Results

### *5.2.1 Detecting Collisions and Matching*

We start by evaluating the effectiveness of our collision detection and same-pair detection algorithms.

**(a) Detecting Collisions**

We first evaluate how well we can detect a collision using the algorithm proposed in Section 3.3. We record 4,000 collision pairs transmitted on different channels (*e.g.*, channels 3 and 4; and channels 3 and 5) respectively.

To precisely locate the collision positions, we use the following technique. Since our software radio platform provides us the flexibility to transmit arbitrary signals, we form a signal that contains a customized PN-marker and a sequence of eight experiment frames, with sufficient silence period between one another. This way, we precisely control the latency between the leading PN-marker and each following frame at the sample-level. We can detect the PN marker quite easily and robustly, and hence we treat this as the ground truth of our experiment.

Then, we use our algorithm to find the collision location of the second frame, and compare it to the result we get using the artificial PN marker. We call it is a false positive, if our algorithm reports a collision when it is presented with a signal with no-collision; while

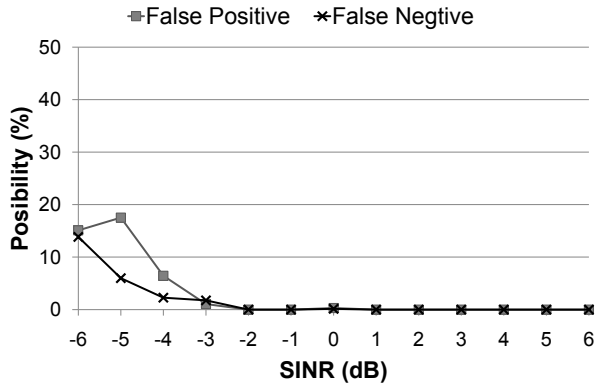



**Figure 8: False positive and false negative probabilities of detecting collisions under different interference levels.**

we call it a false negative, if the algorithm actually misses the true collision location or it reports a collision but with a wrong channel.

Figure 8 shows the average probabilities of false positive and false negative collision detection. In this figure, the x-axis is the signal-to-interference-plus-noise ratio (SINR) of the experimental settings. We do not separate channel 3 and 4 collisions from channel 3 and 5 collisions. In other words, Figure 8 shows the aggregate collision detection results. We can observe that when the interference signal is weak compared with the data signal (SINR > 0 dB), both false positive and false negative are very low. As the interference level increases, false positive and false negative start to increase slightly. The reason is that interference injects noise into the correlation computation. However, even at low SINR, the error probabilities are still low. For example, the false positive probability is below 20% when SINR is as low as -6 dB.

**(b) Detecting Same-Pair Collisions**

We next evaluate our algorithm in Section 3.5 to detect collisions involving the same pair of frames. In particular, the decoding algorithm needs to know if two collisions contain the same pair of frames before it can perform same-pair decoding. Since we do not need collision matching for channel 3 and channel 5, we collect 2,400 collided frames sent on channel 3 and channel 4 only. We classify the recorded frames into two groups. Within one group, all collision pairs contain only the same pair of frames (possibly being remapped); while in the other group, all collision pairs contain different random frames. We randomly sort the collision pairs in each group, and try to match any two adjacent collision pairs. We report a false negative if it fails to match a collision pair in the first group, containing all same-pairs; while it is a false positive if it is reported to match a collision pair in the latter group.

| Channel | False Positive | False Negative |
|---|---|---|
| 3 | 0 | 0.24 |
| 4 | 0 | 0.3 |

**Table 1: Performance of detecting collisions involving the same pair of frames on channels 3 and 4.**

Table 1 reports false positive and false negative probabilities from frames in channel 3 and channel 4. We can see that false positive is zero in our experiments. Thus, the detection algorithm does not falsely report same-pair collisions and thus will not unnecessarily trigger the same-pair decoding process. On the other hand, the algorithm does report false negatives due to interference. We note that false negatives in this case miss the same-pair decoding opportunities, but the collided frames may still be able to be decoded using the general adjacent decoding algorithm.

### *5.2.2 Decoding Performance*

Next we evaluate how well our decoder works under a variety of SNR settings. We use the similar technique described in Section 5.2.1 to get collision frames. We present results for non-matching collision decoding only. We leave matching-collision decoding evaluation for future work. In our experiments, for each frame, Alice always sends the original frame and the three Remapped versions; while Bob continuously sends randomly generated frames. In the third *same-pair collision* experiment, both Bob and Alice will repeat each frame with four different remapping versions. Thus, we can obtain two collision pairs, each of which contains the same frames with different remapping schemes. Unless otherwise mentioned, for each experiment, we report the results with over 2,000 captured collision frames.

**(a) Decoding Non-adjacent Channel Collisions**

We start with decoding collisions in non-adjacent channels. Table 2 shows the throughput at three SNR settings. Since the noise is small in our environment, we denote each setting using the SNR difference between the received signal on the primary channel and that on the secondary channel. We see that, when the SNR of the primary channel signal increases relative to that of the secondary channel from -6.8 dB to 2.7 dB, the BER of decoding on the primary channel decreases (BER is calculated only for those frames that decoded correctly) from $1.5 \times 10^{-2}$ to $8.8 \times 10^{-5}$. Remap decodes more than 90% collision pairs correctly in all cases while normal decoder decodes only between 0 to 8.6% of the collision frames.

| Scenario → | Non-adjacent channel | |
|---|---|---|
| SNR Diff ↓ | BER | Throughput (Remap, Normal) |
| -6.8 | $1.5 \times 10^{-2}$ | (92.7, 0) |
| -1.6 | $2.1 \times 10^{-3}$ | (90.2, 1.8) |
| 2.7 | $8.8 \times 10^{-5}$ | (98.4, 8.6) |

**Table 2: BER and normalized throughput with respect to SNR diff.**

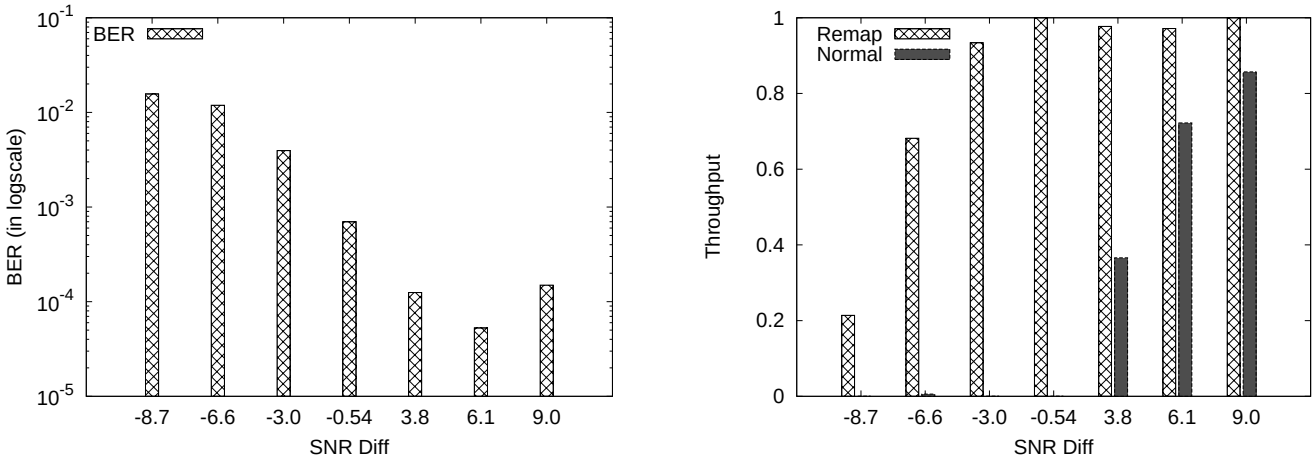
We take a closer look at the scenario where the SNR difference is -1.6 dB. Among the 9.8% (1-90.2%) that Remap is not able to decode, 76% are due to the fact that Remap cannot decode the SIGNAL field of 802.11g preamble when the frames transmitted on the secondary channel arrive first. The reason is most likely that the interference from the secondary channel signal is so strong some times that the SIGNAL field is corrupted. In this setting, in 40% collisions, the frames on the secondary channel arrived first.

As a comparison, normal decoder decodes 1.8% collisions. A closer look shows that the collision frame is mostly collision free, either 3 or 10 bytes overlap at the end.

**(b) Decoding Adjacent Channel Collisions**

Next we evaluate decoding collisions in adjacent channels. Figure 9 shows the BER and throughput of seven settings. We see that the BER (calculated before the Viterbi process) of correctly decoded frames decreases from $1.6 \times 10^{-2}$ to below $1.5 \times 10^{-4}$ when the SNR difference increases from -8.7 dB to 9 dB. We see that, when the SNR difference is below -0.5 dB, normal decoder decodes almost no frames correctly. When the SNR difference goes from 3.8 dB to 9 dB, normal decoder's decoding fraction increases from 37% to 86%. We also see that Remap's decoding fraction will significantly reduce if the signal received on the secondary channel is stronger than the signal on the primary channel by more than 9 dB.

## 6. RELATED WORK




(a) BER vs SNR diff between received channel 3 and 4 signal

(b) Throughput

**Figure 9: Decoding non-adjacent channel collisions (channels 3 and 5).**

Remap's key contribution is a novel bit-to-subcarrier remapping technique for resolving collisions in overlapping channels. It is different from prior techniques on interference cancellation and joint decoding (*e.g.*, [11, 15, 24]). These techniques operate on a single collision, and their decoding capability is limited by the channel capacity. Remap is related to analog network coding [13]. However, a receiver capable of analog network coding can decode a collision only if it knows one of the collided frames.

Zigzag is designed in the context of no bit-to-subcarrier remapping. Remap builds on top of Zigzag. Remap works for overlapping channel collisions and is designed to take advantage of bit-to-subcarrier remapping. Unlike Zigzag, Remap does not rely on matching collisions. In the case of two matching collisions, Remap enables diversity combining from receptions at multiple access points (APs) as used in [18]. The reason is that Remap can decode both frames involved in the collisions as long as the number of overlapping subcarriers is no fewer than half of the total.

Remap differs from prior work on MAC layer techniques (*e.g.*, [3, 12]) to avoid collisions. Remap decodes collisions and works with 802.11 MAC without modifications.

The concept of decoding interfering users has been an area of intense study in communication and information theory. Techniques such as MIMO [23] and interference alignment (*e.g.*, [4, 8, 9]) require synchronization, coding and scheduling. In contrast, Remap resolves collisions without these requirements.

Although one may try to alleviate the shortage of orthogonal channels by using variable bandwidth channels, as advocated in [5, 10], bursty or time varying workload can pose a problem for channel width adaptation.

# 7. CONCLUSIONS AND FUTURE WORK

We presented Remap, a simple, novel paradigm that uses retransmission permutation for decoding collisions in overlapping OFDM channels. Our prototype implementation of applying Remap to 802.11g demonstrates that Remap can be simple to implement and able to exploit collision-free subcarriers to decode frames despite successive collisions.

We believe that Remap can offer key insight in resolving collisions in other OFDM based wireless systems that have different subcarrier and channel structures. We are investigating techniques inspired by Remap in the context of future dynamic spectrum access networks, where different nodes may select overlapping frequency bands.

# 8. ACKNOWLEDGMENTS

We are grateful to our shepherd Kyle Jamieson and the anonymous reviewers whose comments and suggestions helped to improve the paper. We thank Junliang Liu for conducting initial experiments. We also thank Prof. Zhiyong Feng for the support during this project. The research of Y. Richard Yang is supported by part by NSF Grant CNS-1018502.

# 9. REFERENCES

[1] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployments. In *Proceedings of MobiCom*, Cologne, Germany, Aug. 28-Sept. 2 2005.

[2] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh. White space networking with Wi-Fi like connectivity. In *Proceedings of SIGCOMM*, Barcelona, Spain, Aug. 2009.

[3] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A media access protocol for wireless LANs. In *Proceedings of ACM SIGCOMM*, London, England, Aug. 1994.

[4] V. R. Cadambe and S. A. Jafar. Interference alignment and the degrees of freedom for the K user interference channel. Preprint, 2007.

[5] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl. A case for adapting channel width in wireless networks. *SIGCOMM Comput. Commun. Rev.*, 38(4):135–146, 2008.

[6] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[7] S. Gollakota and D. Katabi. Zigzag decoding: combating hidden terminals in wireless networks. *SIGCOMM Comput. Commun. Rev.*, 38(4), 2008.

[8] S. Gollakota, S. D. Perli, and D. Katabi. Overcoming the antennas-per-node throughput limit in MIMO LANs. Technical Report MIT-CSAIL-TR-2009-007, MIT, 2009.

[9] K. S. Gomadam, V. R. Cadambe, and S. A. Jafar. Approaching the capacity of wireless networks through

distributed interference alignment. In *Proceedings of IEEE GLOBECOM*, Miami, FL, Dec. 2008.

[10] R. Gummadi, R. Patra, H. Balakrishnan, and E. Brewer. Interference Avoidance and Control. In *Proceedings of Hotnets-VII*, Calgary, Alberta, Oct. 2008.

[11] D. Halperin, T. Anderson, and D. Wetherall. Taking the sting out of carrier sense: interference cancellation for wireless LANs. In *Proceedings of MobiCom*, San Francisco, CA, Sept. 2008.

[12] P. Karn. MACA - a new channel access method for packet radio. In *Proceedings of ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, Sept. 1990.

[13] S. Katti, S. Gollakota, and D. Katabi. Embracing wireless interference: Analog network coding. In *Proceedings of ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.

[14] Kun Tan et al. Sora: High performance software radio using general purpose multi-core processors. In *Proceedings of NSDI*, Boston, MA, Apr. 2009.

[15] L. E. Li, R. Alimi, R. Ramjee, J. Shi, Y. Sun, H. Viswanathan, and Y. R. Yang. Superposition coding for wireless mesh networks. In *Proceedings of MobiCom*, Montreal, QC, Sept. 2007.

[16] Madwifi. Madwifi. Available at `http://madwifi.org/`.

[17] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaugh. Partially overlapped channels not considered harmful. *SIGMETRICS Perform. Eval. Rev.*, 34(1):63–74, 2006.

[18] A. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proceedings of MobiCom*, Cologne, Germany, Sept. 2005.

[19] J. Proakis and M. Salehi. *Digital Communications*. McGraw-Hill, 2008.

[20] H. Rahul, F. Edalat, D. Katabi, and C. G. Sodini. Frequency-aware rate adaptation and MAC protocols. In *Proceedings of MobiCom*, Beijing, China, Sept. 2009.

[21] H. Samra and Z. Ding. Retransmission diversity schemes for multicarrier modulations. *Wireless Communications, IEEE Transactions on*, 5(5):1142–1147, 2006.

[22] A. Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52:2551–2567, 2006.

[23] D. Tse and P. Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, May 2005.

[24] S. Verdu. *Multiuser Detection*. Cambridge University Press, New York, NY, 1998.

[25] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *Proceedings of SIGCOMM*, Barcelona, Spain, Aug. 2009.

# APPENDIX

In this appendix, we discuss a few additional technical details about implementing Remap, in particular implementing Remap for 802.11g.

## A. DEALING WITH SCRAMBLER

In the IEEE 802.11g standard, all bits in a frame are scrambled before coding and modulation. The initial state of the scrambler, also called seed, is randomized each time a frame is transmitted, including retransmission. This behavior creates an additional challenge for Remap. If a different scramble seed is used for the retransmission, it is impossible for Remap to collect bit blocks from retransmitted frames. Thus, in our implementation, we simply use the same seed for all retransmissions and only change it for a new frame.

## B. 802.11 SPECIFIC SUBCARRIER GROUPS

In the main text, we do not go into more specifics of the 802.11 subcarrier structure. 802.11 subcarriers are numbered from -32 to +31. Subcarriers [-32,-27],[27,31] and 0 are unused. Subcarriers -21,-7, +21, +7 are used as pilots. Non-pilot subcarriers in [-26,-1] and [1,26] are mapped with information bits.

In 802.11g/Remap, we divide subcarriers into four groups ($G_1$, $G_2$, $G_3$, $G_4$) with an equal number of subcarriers in each group: [-26,-14], [-13,-1], [1,13] and [14,26]. However, in this assignment, the adjacent channel subcarrier groups do not align with the primary channel. In particular its subcarrier groups are mapped into [-10,2], [3,15], [17,29] and [30,42] as shown in Figure 10.

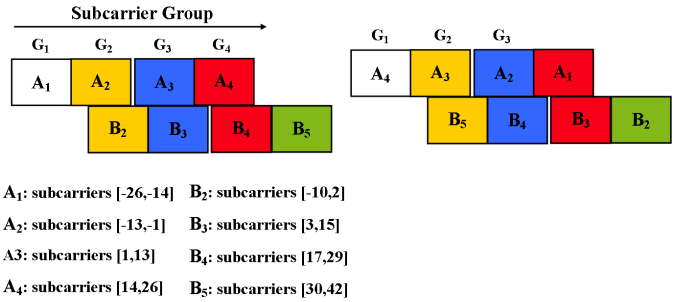



**Figure 10: Subcarrier view of collision with adjacent channel**

- Bit block $A_1$ (12 bits) maps to subcarrier group $G_1$: [-26,-14] (from subcarrier -26 to -14, 13 subcarriers, -21 is the pilot)
- Bit block $A_2$ (12 bits) maps to subcarrier group $G_2$: [-13,-1]
- Bit block $A_3$ (12 bits) maps to subcarrier group $G_3$: [1,13]
- Bit block $A_4$ (12 bits) maps to subcarrier group $G_4$: [14,26]
- When remap $A_1$ from $G_1$ to $G_4$, subcarrier -26 maps to subcarrier 14, subcarrier -14 maps to 26; that is, subcarrier j in $G_1$ maps to 40+j in $G_4$
- When remap $A_2$ from $G_2$ to $G_3$, subcarrier -13 maps to subcarrier 1,
- Subcarrier -1 maps to 13; that is, subcarrier j in $G_2$ maps to 14+j in $G_3$
- When remap A3 from $G_3$ to $G_2$, subcarrier 1 maps to subcarrier -13, subcarrier 13 maps to -1; that is, subcarrier j in $G_3$ maps to -14+j in $G_2$
- When remap A4 from $G_4$ to $G_1$, subcarrier 14 maps to subcarrier -26, subcarrier 26 maps to -14; that is, subcarrier j in $G_4$ maps to -40+j in $G_1$

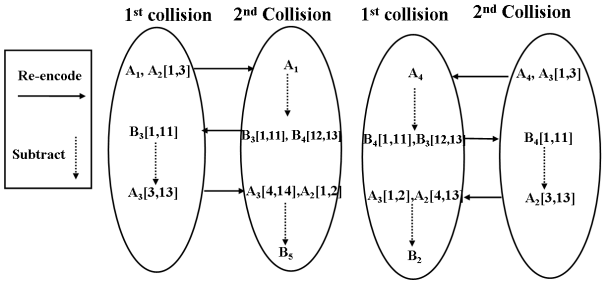



**Figure 11: Same-pair decoding graph for more complex 802.11.**

We modify Remap same-pair decoding. Figure 11 shows the actual decoding process. Note that, we should try to avoid decoding

the adjacent channel subcarriers if they fall into the unused subcarriers of the primary channel. In our decoding process, we only make use of one unused subcarrier, +27. $B_i[j,k]$ denotes the subcarrier indexed from $j$ to $k$ within the subcarrier groups where $B_i$ uses. For example, $B_3[1,11]$ refers to subcarriers [3,13] for the first collision. We need subcarrier +27 to decode $B_3[11]$ and $B_4[11]$. Since it is adjacent to used subcarriers, the filter response should still be sufficient.

## C. CHANNEL ESTIMATION IN SAME-PAIR DECODING

For Bob's frame, the FFT is performed using the timing of Bob's frame. We have two techniques corresponding to two cases. First, we estimate channel on subcarrier group $G_i$ when we have already subtracted the interference signal on subcarrier group $G_i$. As an example, in Figure 6, at step 3, we have already subtracted the subcarrier symbols corresponding to $A_1$ on subcarrier group $G_4$ from the samples.

We can use the same Equation (3). Second, we estimate channel on subcarrier group $G_i$ when we have already known the bits encoded on subcarrier group $G_i$. For example, in Figure 6, at step 4, we need to subtract $B_3$ from the first collision samples. However, we have already decoded $B_3$ now. We do a correlation in the frequency domain:

$$\begin{aligned} \Gamma &= \sum_{l=1}^{M} Y[l,i]{X_b}^*[l,i] \\ &= \sum_{l=1}^{M} (H_b[i]X_b[l,i] + H_a[i]X_a[l+\Delta] + w[i]){X_b}^*[l,i] \\ &= H_b[i]\sum_{l=1}^{M} |X_b[l,i]|^2. \end{aligned} \quad (9)$$