CRMA: Collision-Resistant Multiple Access

Tianji Li[1,2] Mi Kyung Han[1] Apurv Bhartia[1] Lili Qiu[1] Eric Rozner[1] Yin Zhang[1] Brad Zarikoff[2]
The University of Texas at Austin[1], National University of Ireland Maynooth[2]
{tianji,hanmi2,apurvb,lili,erozner,yzhang}@cs.utexas.edu, brad.zarikoff@nuim.ie

Abstract– Efficiently sharing spectrum among multiple users is critical to wireless network performance. In this paper, we propose a novel spectrum sharing protocol called Collision-Resistant Multiple Access (CRMA) to achieve high efficiency. In CRMA, each transmitter views the OFDM physical layer as multiple orthogonal but sharable channels, and independently selects a few channels for transmission. The transmissions that share the same channel naturally add up in the air. The receiver extracts the received signals from all the channels and efficiently decodes the transmissions by solving a simple linear system. We implement our approach in the Qualnet simulator and show that it yields significant improvement over existing spectrum sharing schemes. We also demonstrate the feasibility of our approach using implementation and experiments on GNU Radios.

Categories and Subject Descriptors

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

General Terms

Experimentation, Performance

Keywords

MAC Protocol, CSMA, CDMA, Collision Decoding, OFDM

1. INTRODUCTION

Motivation: The design of spectrum sharing and multiple access is critical to wireless network performance. Traditional approaches share spectrum by slicing it across time or frequency, and allocate each slice to at most one user at any time. When multiple transmissions overlap on the same frequency, a collision occurs and no useful data is delivered. To reduce collisions, various techniques are proposed. For example, in the time domain, this can be achieved by using centralized scheduling (*e.g.*, PCF [25]), carrier sense (*e.g.*, IEEE 802.11 DCF [25]), or hybrids (*e.g.*, [36]); in the frequency domain, static or dynamic channel assignment and channel hopping can be employed. We can further combine time and frequency division multiplexing (*i.e.*, allowing a node to use a given frequency at a given time).

However, all the efforts aiming to resolve collisions can lead to considerable overhead. It is well-known that slicing spectrum across frequency alone (FDMA) is inefficient because nodes may occupy frequency without generating traffic [34]. On the other hand, slicing spectrum across time has to meet the challenging requirement that no more than one user can access the same spectrum at any time. The latter either requires global coordination and centralized scheduling, which is hard to implement in a distributed network, or resorts to carrier sense, which has several well-known issues, such as hidden terminals [5, 12, 14], exposed terminals [12], and significant overhead. The overhead is especially significant for small frames, high data rates [37], or long-distance networks in order to sense a far-away transmitter and avoid collisions [24, 26, 32]. For example, the carrier sense overhead is 75.2% for TCP ACKs in an 802.11a network with 54 Mbps and similar for VoIP packets. The overhead increases to 82.7% in an 802.11n network at 600 Mbps, and the carrier sense time further increases by 167% in order to carrier sense a transmitter that is 3000 meters away (instead of 300 meters away) [18].

Our approach: Motivated by the limitations of the existing spectrum sharing approaches, we propose a new direction of spectrum sharing, called Collision-Resistant Multiple Access (CRMA). In CRMA, every transmitter views the OFDM physical layer as multiple orthogonal but sharable channels, and randomly selects a subset of the channels for transmission. [1] When multiple transmissions overlap on a channel, these signals will naturally add up in the wireless medium. The receiver j can construct a simple linear system

$$A(i, j, f)x(i) = R(j, f).$$

$x(i)$ denotes the transmitter i's signals, $R(j, f)$ denotes the receiver j's received signal on channel f, and the matrix $A(i, j, f) = c(i, f) \times h(i, j, f)$, where $h(i, j, f)$ is the channel distortion from transmitter i to receiver j on channel f and $c(i, f)$ is a code chosen by the transmitter to scale the signal. The code $c(i, f)$ can be either binary (*i.e.*, 1 when the signal $x(i)$ is present on channel f and 0 otherwise) or non-binary. The receiver then decodes x based on A and R by solving the linear system. By using coding, transmitters no longer need to avoid collisions. Upon successful decoding, it can continuously send without incurring carrier sense overhead. Figure 1 shows an example of two senders transmitting signals on two channels. A receiver can extract the original signals by solving a simple linear system of two equations.

CRMA shares the same spirit as CDMA in that both use coding to share spectrum, but differs significantly from CDMA in the follow-

[1]Transmitting on non-consecutive frequency bands is feasible as demonstrated by the implementations in previous works [31, 43].

MobiCom'11, September 19–23, 2011, Las Vegas, Nevada, USA.

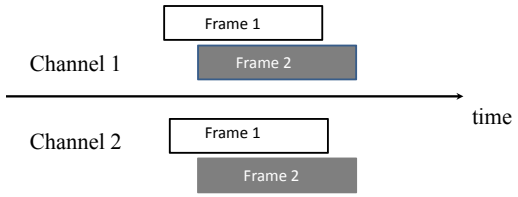


Figure 1: An example of two senders sharing two channels. The signals received on the two channels at node j are $R(j,1) = h(1,j,1) \times x1 + h(2,j,1) \times x2$ and $R(j,2) = h(1,j,2) \times x1 + h(2,j,2) \times x2$, respectively, where $x1$ and $x2$ are transmitted signals and $h(i,j,f)$ are the channel coefficients. We can decode $x1$ and $x2$ based on R and h.

ing ways. There are two classes of CDMA: synchronous CDMA and asynchronous CDMA. CRMA differs from synchronous CDMA in that it does not require that all transmissions start at the same time, and differs from asynchronous CDMA in that the latter suffers from Multiple Access Interference (MAI) whereas we explicitly incorporate the other transmissions involved in collisions when constructing the linear system for decoding so that we can cancel out the interference and achieve high reception rate.

Our use of random coding is related to network coding, where the receiver receives random linear combinations of the original frames and decodes the frames by solving a linear system. Different from existing network coding literature, we exploit the flexibility of random linear coding to develop an efficient spectrum sharing protocol. This protocol is general and its benefits are especially significant for environments where carrier sense is inefficient, such as hidden terminals, exposed terminals, networks with long-distance communication, high data rates, or short frames.

We implement our approach in the Qualnet simulator [28] and compare it with several existing spectrum sharing protocols, including CSMA, random access, and Wi-Fi. Our results show that CRMA significantly out-performs the other schemes. We further demonstrate the feasibility of our approach using experiments on GNU Radios.

In summary, this paper proposes a new direction of spectrum sharing. It makes three major contributions: (1) a novel encoding and decoding algorithm that enables simultaneous transmissions of interfering signals on the same frequency band, (2) a new spectrum sharing protocol that builds on top of the new coding scheme, and (3) experimental evaluation of the protocol to show this is a promising approach.

Paper outline: The remainder of the paper is organized as follows. In Section 2, we give a brief background on wireless communication. We present our approach in Section 3. In Section 4, we compare the efficiency of our approach with the existing schemes using Qualnet simulation. We demonstrate the feasibility of our approach using testbed experiments in Section 5. We review related work in Section 6, and we conclude in Section 7.

2. BACKGROUND

To transmit a frame, a transmitter uses modulation to convert bits into a series of discrete complex numbers. For example, BPSK modulation uses $e^{j\pi}$ to represent a "0" bit and uses e^{j0} to represent a "1". Let x denote transmitted symbols, and y denote received symbols. They have the following relationship:

$$Y[n] = h[n]X[n] + w[n], \tag{1}$$

where the channel coefficient $h[n] = \alpha[n]e^{-j\theta[n]}$ is a complex number that consists of the channel attenuation $\alpha[n]$ and phase shift $\theta[n]$, and $w[n]$ is additive white Gaussian channel noise. A traditional receiver decodes $X[n]$ by estimating $h[n]$ using the training sequence, which is a known sequence of symbols and included in every frame.

When multiple signals transmit simultaneously, their signals add up as follows:

$$Y[n] = \sum_i Y_i[n] + w[n], \tag{2}$$

where $Y_i[n] = h_i X_i[n]$ denotes the signal after channel distortion.

In this paper, we consider an OFDM physical layer, which is widely used today, such as in IEEE 802.11a/g/n, WiMax, and LTE. In OFDM, each channel consists of multiple orthogonal subcarriers, each of which is used to carry data independently. When a data symbol $x(i)$ in CRMA is transmitted onto K channels, it is replicated to one subcarrier in each of the K channels. Each subcarrier has a channel distortion $h(i,j,f,s)$, which denotes the channel distortion from transmitter i to receiver j on channel f at subcarrier s. Since each symbol is mapped to exactly one subcarrier in each channel, s is unique for a given channel f and a given symbol in the frame (*e.g.*, the first symbol in a frame is always mapped to 10-th subcarrier on channel 1). Therefore to simplify the notations without introducing ambiguity, we *drop* s in the rest of the paper.

3. OUR APPROACH

In CRMA, a sender selects K channels and sends copies of its frame onto all the selected channels. A receiver receives the sum of all the signals transmitted on each channel and constructs a linear system that reflects the relationship between the original signal x, received signal R, code c, and channel coefficients h. It can then decode x based on the knowledge of R, c, and h. In order to realize this protocol, several important research issues must be addressed:

- How to select a code?
- How to establish the code between a sender and receiver?
- How to encode and decode transmissions?
- How to limit the number of transmissions involved in a collision to allow successful decoding?
- How to enhance spectrum utilization?
- How to handle decoding failures?
- How to deal with misaligned collisions?

Below we address each of these issues. In our description, we define a flow as a pair of communicating nodes. In Section 3.1 – Section 3.6, we assume OFDM symbols from different transmissions are synchronous so that one signal's FFT window contains complete symbols from other signals. We relax this assumption in Section 3.7.

3.1 What is a Code?

In this paper, we use a binary code, where the code $c(i,f) = 1$ when transmitter i uses channel f and $c(i,f) = 0$ otherwise. In general, the code can be non-binary, and the transmitter can scale its signal according to the code, *e.g.*, $c(i,f)$ is the scaling factor that transmitter i uses on channel f. Non-binary codes can be used to increase the chance of linear independence in matrix A. For simplicity, we use binary codes and show how to select binary codes that maximize decodable collisions.

Let us ignore the channels on which there are no transmissions (*i.e.*, the matrix $A(i,f) = c(i,f) \times h(i,j,f)$ does not have columns

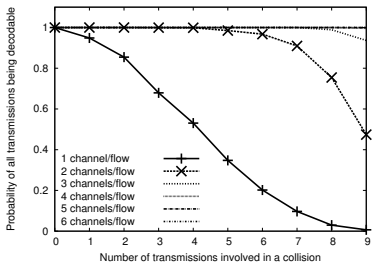


Figure 2: Probability of all N transmissions being decodable as we vary the number of channels per flow (K) and the number of transmissions involved in collisions (N). (20 flows compete for 10 channels. Each flow selects the K least-used channels out of 10 channels. At any time a random set of N transmissions are active.)

with all zeros.). In order for a collision to be decodable, the matrix A should be full rank. This requires (1) the total number of transmissions involved in collision should not exceed the total number of channels being used, and (2) the rows of matrix A should be linearly independent. We will address (1) in Section 3.4. Condition (2) can be satisfied when (i) different flows that select the same channel see different channel coefficients on the channel or (ii) different flows select different sets of channels.

To see how (i) can hold, we note that an OFDM channel consists of multiple subcarriers. Channel coefficients vary significantly across subcarriers [2, 11, 29]. Moreover, several previous works (*e.g.*, [17, 27, 45]) show the channel coefficients across subcarriers (within a channel) can be used as a unique signature to distinguish one link from another. Other work uses channel coefficients across subcarriers to bootstrap keys for secure communication [19]. The property that all these works leverage is that different node pairs have different channel coefficients across the same set of subcarriers. Therefore, as long as the number of flows is no more than the number of their shared channels, the matrix A is likely to be linearly independent.

To further increase the chance of linear independence in the matrix A, we let each flow select the least-used channels. In this selection, each node monitors the total number of flows on each data channel (as described in Section 3.3), and selects the channels that have the smallest number of flows. We perform an analysis with 20 flows competing over 10 channels without carrier sense. We vary the number of active flows, denoted as N, from 1 to 10. We also vary the number of channels that each flow uses, denoted as K, from 1 to 6. Figure 2 plots the decoding probability (*i.e.*, the probability that the linear system has a unique solution). As we can see, when each flow selects 3 channels, the decoding probability is 94% even when the number of transmissions is equal to the number of channels ($N = K = 10$). When there are fewer transmissions, the decoding rate is very close to 100%. In comparison, the traditional scheme uses one channel per flow, and its decoding probability is much lower without carrier sense. This confirms our intuition that using multiple channels per flow as in CRMA is more robust and can achieve a higher decoding ratio than the existing schemes, which use only one channel at a time and have to ensure that no transmissions overlap on any channel. We also analyze random channel selection, and the relative performance is similar and omitted in the interest of brevity.

3.2 Code Establishment

We develop two approaches for code establishment. The first approach exchanges code on the same channel as used by data transmissions. It does not require a separate control channel but incurs more processing for a receiver to detect a flow destined to it. The second approach exchanges the code on a separate control channel so that a receiver does not require additional in-band processing on the data channels. For ease of explanation, we describe our approach for binary codes, but the approach can be easily extended to non-binary codes. We evaluate the feasibility of in-band notification using a GNU Radio implementation, and report its accuracy in Section 5.2. We implement the control channel based code selection in Qualnet, and the performance in Section 4 is based on this code selection.

In-band notification: A source s selects K channels and sends copies of its data on all the selected channels. Since the frame can be involved in a collision, simply including the source and destination in the frame as usual does not work. Instead we use PN sequences to encode the source and destination addresses so that the receiver can detect them even under collisions. Specifically, a receiver first correlates the received signal with P, a known PN sequence corresponding to its own address. The correlation is close to zero except when the incoming signal containing P is perfectly aligned with P. A spike in the correlation indicates the frame contains the receiver's address. Only then does the receiver attempt to run another correlation between a received signal and the PN sequences corresponding to possible source addresses. Since the source sends copies of its data on all the selected K channels, the receiver expects to see spike in correlation on K channels, which reveals the set of channels the source uses for communication.

To further improve resilience (*e.g.*, not all K channels see correlation spikes), the sender could use consistent hashing to map $(senderID, receiverID)$ to a set of K channels. Specifically, consider there are M total channels and each flow uses K channels. Then there are C_K^M possible channel selections for a binary code, and we can number these selections from 1 to C_K^M. When a flow starts, the source simply hashes ($SenderID$, $ReceiverID$) to an integer between 1 and C_K^M. As soon as the receiver finds out the sender address on at least one of the K channels using correlation, it computes the same hash function independently to identify all the K channels used by the source. We plan to implement and evaluate this consistent hash based enhancement in the future.

Control channel based code selection: Alternatively, we can use a control channel to negotiate the code. Code negotiation is performed either (i) periodically or (ii) whenever the loss rate on the data channels exceeds a threshold. A number of approaches have been proposed in the literature to establish a control channel (*e.g.*, [3, 23, 44]). We can apply them to our context so that we can focus on how to use a control channel for code negotiation. We assume that users know which channel is used as the control channel and which channels are used as data channels.

Different users use CSMA/CA to share the control channel. To enhance the reliability of control messages, we use a high transmission power to increase the communication range and a conservative clear channel assessment (CCA) threshold to reduce collisions. Since the control traffic is infrequent, the overhead of CSMA and the performance loss from potential exposed terminals due to the conservative CCA threshold is insignificant.

A sender S that wishes to start communication with receiver R sends a *request for code* on the control channel. R responds by sending the selected set of K data channels to use. We further leverage link-layer ACKs and retransmissions to provide reliability as in IEEE 802.11 (*i.e.*, a sender will retransmit its request for a code if it does not receive a reply within a timeout).

Upon receiving the code, the sender and receiver broadcast the selected code multiple times to inform their neighbors of their code.

Within this broadcast message, the sender and receiver also include the codes selected by other nearby ongoing flows that they overhear. In this way, even when the message containing the code is lost, it will be re-broadcast later by other nodes, thereby enhancing the delivery probability of the selected code.

3.3 Encoding and Decoding Transmissions

For ease of explanation, we focus on decoding one symbol. To decode the symbol, a receiver j extracts the incoming signals from all channels, where $R(j, f)$ denotes the signal at receiver j on channel f. The receiver then constructs the following linear system. For any $f \in [1..M]$, where M is the total number of data channels, we have

$$\forall j, f : \sum_i h(i, j, f)c(i, f)x(i) = R(j, f) \quad (3)$$

where $h(i, j, f)$ is the channel coefficient from sender i to receiver j on channel f, $c(i, f)$ is the code sender i uses on channel f, and $x(i)$ is the signal that transmitter i sends. The left-hand side is the sum of all received signals, and the right-hand side is the final received signal at receiver j on the channel f. Our decoding algorithm extracts $x(i)$ based on $R(j, f)$. In order to achieve this goal, we should address the following important questions: (i) how to detect frame arrival and departure, (ii) how to find out which transmitter's signals are present on a given data channel in order to construct the above linear system, (iii) how to estimate the channels, and (vi) how to efficiently solve the linear system. Below we address each of these questions in turn.

Detecting frame arrival and departure: We use a preamble and a postamble to detect a frame arrival and departure. Each frame starts with a preamble, which contains a known sequence of symbols. Each frame ends with a postamble, which is another known sequence of symbols. We focus on detecting a frame arrival using a preamble, since detecting departure using a postamble is similar. We first describe how to detect frame arrival using a clean preamble, which does not collide with other signals. We then extend to the case when the preamble collides with other signals.

When a receiver receives a clean preamble, it stores the received preamble. It runs correlation between the stored preamble and the new incoming signals. To maximize the detection accuracy, we use PN sequences as the preamble. The correlation is close to zero except when the training sequence is perfectly aligned with the beginning of a frame transmitted by the same sender [8]. We observe a spike in the correlation whenever a new frame arrives. Note that spikes at true collision positions might sometimes be low due to the frequency offset between the sender and the receiver. In such a case, the receiver needs to compensate for the frequency offset to get a good correlation spike [8]. After compensating for the frequency offset, we can accurately detect frame arrival even when SNR is -6 dB, as shown in Section 5.2.

Correlation-based detection is highly robust. As shown in [35], a transmitter can use signal correlation to detect the presence of a signature in the received signal even if it is in transmission mode. The detection accuracy is high even under -32 dB. Therefore a node can count the number of on-going transmissions based on the arrivals of preambles and postambles even while it is transmitting.

Detecting transmitters involved in collisions: A receiver needs to know if there is a frame destined to it to invoke the decoding algorithm. As described in Section 3.2, a receiver can run correlation with the PN sequence corresponding to its address, and a spike in the correlation indicates that a frame contains its address. Similarly, it can use correlation to identify the addresses of the transmitters in order to construct a linear system for decoding. More specifically, since it is very rare to have multiple nodes start a frame transmission at the same time, the number of spikes in the correlation between the received signal and training preambles from other nodes indicates the number of signals involved in the collision. To further identify the set of transmitters involved in a collision, we use correlation to identify the source address encoded in the PN sequence, which are transmitted after the preamble and before the postamble. In this way, the receiver knows the set of transmissions involved in collisions and can use the channel coefficients corresponding to the flows to construct a linear system for decoding.

One potential concern with the correlation based detection is the time complexity. Given N flows (*i.e.*, source destination pairs) in the network, a naive algorithm will correlate the incoming signal on each channel with each of the N flow IDs, which takes $O(N)$ per channel. We can speed up the correlation by encoding the flow ID using b sequences, each representing one of the $N^{\frac{1}{b}}$ PN sequences. For example, when $b = 2$, 100 flow IDs can be encoded using two PN sequences and each of them takes 10 different values, where the first PN sequence represents the most significant digit and the second one represents the least significant digit. So we just need to correlate with 10 sequences instead of 100. When $b = 4$, we can encode 100 flow IDs using 4 PN sequences, each takes 4 different values, and we can correlate with only 4 PN sequences to identify the flow ID. Increasing b reduces the number of PN sequences to be correlated and hence the correlation time, but this increases the preamble length. Therefore it is desirable to select the smallest b that the system can afford in order to limit the preamble length.

Obtaining channel estimates: First, we consider how a receiver estimates the channel using preambles. Then we examine how a receiver continuously updates the channel estimation using the frame data (*i.e.*, anything but a preamble). The latter is especially important when the frame is so large that the channel may change during the transmission time of a frame.

A receiver can use the standard approach to estimate the channel if it receives a clean preamble. So we now focus on the case when the received preamble collides with other transmissions. If multiple preambles collide, we can apply the technique used in MIMO, where different transmitters send preambles coded with different PN sequences so that a receiver first extracts a clean preamble from each transmitter using its corresponding PN sequence and applies the standard channel estimation on the clean preamble. Since the probability of preamble collision is low, we have not implemented this approach.

Next we consider when a preamble collides with data from other transmissions. We first construct a linear system where all the channel coefficients are known from the previous estimates and the only unknowns in the linear system are those transmissions' data involved in the collision. Since the decoded data symbols will take discrete values (*e.g.*, either +1 or -1 in BPSK), we can decode the data as the ones that give the closest match to the received signal. Then we construct a new linear system by plugging the decoded values back into the linear system, treating the channel coefficients as unknown, and solving the new linear system. The new solutions give new channel estimates. We then use an exponentially weighted moving average (EWMA) to combine the new channel coefficients with the previous estimates. That is, $h(i, j, f) = w \cdot h^{new}(i, j, f) + (1 - w)h(i, j, f)$. The weight w is a tunable parameter, and is set to 0.2 in our testbed evaluation.

We can apply a similar approach to keep track of the channel changes during decoding of frame data. Specifically, we first use the previous channel estimation to construct a linear system and solve for the unknown data. Then we plug the decoded data symbols back to the system, and solve for the channel coefficients by

treating these coefficients as unknown. We can update the channel estimation every N_s data symbols, where N_s controls the computation time and freshness of the channel estimates. A large N_s gives faster computation time since (as described below) we only need to build a lookup table once for N_s data symbols, but a large N_s can cause channel estimates to become stale and degrade performance. Our experiments in Section 5.2 evaluate the decoding rate and computation time for different values of N_s.

Solving the linear system: The linear system has the form of

$$A \cdot x = R$$

where x belongs to a set of valid symbols (*e.g.*, +1 or -1 in BPSK). A simple way is to solve the linear system is to enumerate all values of x and find the combination of x that minimizes the fitting error, *i.e.*, find x such that $\|A \cdot x - R\|_2$ is minimized. To speed up the computation, for a given A, we build a lookup table, which computes $A \cdot x$ for all possible combinations of x. Then for a given R, we simply perform a table lookup to find the x that minimizes the fitting error. The same lookup table can be used until A changes, which occurs when the set of transmissions involved in a collision changes or the channel condition changes considerably. Section 5.2 further quantifies the computation overhead.

3.4 Limiting Overlapping Transmissions

To allow correct decoding, we should limit the number of transmissions involved in a collision to be within the number of channels being used. We develop two approaches to address this challenge: a sender-side approach and a receiver-side approach. We implement and evaluate the sender-side approach in Section 4 and plan to quantify the effectiveness of the receiver-side approach as part of the future work.

Sender-side approach: A sender usually immediately transmits whenever it has data to send (*i.e.*, $CW = 0$). However, if it does not receive an ACK after it transmits a data frame, this indicates a potential collision. So it enters a backoff mode (*i.e.*, $CW = CW_s > 0$), where CW_s is the smallest positive backoff value and set to 15 in our evaluation. During the backoff mode, it waits for a random time chosen from $[0, CW]$ slots, as in IEEE 802.11. When the backoff is over, the sender transmits immediately with a probability Q. Q is chosen to maximize the expected number of decodable transmissions based on the expected number of new simultaneous transmissions, which can be estimated using the difference between the total number of flows seen from the code establishment in Section 3.2 and the total number of existing transmissions. [2] With a probability of $1 - Q$, it defers and updates $CW = min(\alpha \cdot CW, CW_{max})$, where $\alpha > 1$ further reduces collisions and is set to 1.5 in our evaluation, and CW_{max} is the maximum CW and is set to $15 \cdot \alpha^{12}$. Eventually when it transmits and its frame is lost again, it updates $CW = min(\alpha \cdot CW, CW_{max})$. Whenever an ACK is received, $CW = CW/\alpha$ and if $CW < CW_s$, $CW = 0$. In this way, a sender can continuously send if its data frame is acknowledged and only pays for the backoff cost when necessary (*i.e.*, frames are lost).

Receiver-side approach: Alternatively, we can let the receivers decide who to transmit. Whenever a receiver receives a frame destined to it (which can be detected using correlation in Section 3.3), it determines whether its sender can continue sending the next frame. If so, it informs the sender by sending a notification encoded as a PN sequence so that the sender can detect it using correlation even in presence of collisions.

The receiver will send the notification to its sender when one of the following conditions holds: (i) the number of on-going transmissions is within $M - margin$, where M is the total number of channels, (ii) if not, it uses consistent hashing to choose M out of the current N transmitters involved in the collision to continue transmitting. All the receivers who see the same set of transmitters involved in collisions will choose the same set of M transmitters to continue. The unchosen sender will remain silent until it detects the receiver's notification, which will be sent when the number of on-going transmissions seen by the receiver is within $M - margin$.

The receiver-side approach offers several unique advantages: (i) it captures the receiver's channel condition, which is the one that matters; (ii) it knows the real reason for losses (*e.g.*, whether due to too many transmissions or due to weak signal from the transmitter) and sends notification accordingly; (iii) even under undecodable collisions, the chosen senders can immediately send without waiting, thereby fully utilizing the resources.

3.5 Enhancing Spectrum Utilization Using Virtual Flows

When the total number of flows N_f is below the total number of channels M, the spectrum might be under-utilized. There are two ways to address the issue: (i) increase the channel width and reduce M to close to N_f, or (ii) increase N_f by creating more flows. Modifying channel width complicates decoding since different flows may not see the same number of contending flows, and it is difficult to support heterogeneous channel widths in a network. Therefore we adopt (ii) by letting each physical flow create virtual flows so that the total number of virtual flows N_f is close to M. For example, when only 2 physical flows use 10 channels, we let each physical flow create 5 virtual flows and stripe the data across these 5 virtual flows. In this way, essentially we have 10 flows in the system using 10 channels, achieving high utilization.

3.6 Handling Decoding Failures

A decoding failure arises when a receiver fails to correctly construct the linear system (*e.g.*, due to an inaccurate channel estimate) or the channel coefficients of different flows are not linearly independent. We use ACKs and retransmissions to enhance reliability of data traffic. Right after transmitting the data frame, the sender waits for an ACK. The ACK is sent in the same way as data frames. That is, a receiver sends an ACK on the same set of selected channels and the sender decodes the ACK by solving a linear system. To further improve efficiency of retransmissions, we can further apply partial frame recovery to extract correct symbols from a partially corrupted frame.

3.7 Handling Misalignment between Colliding OFDM Symbols

In real networks, frames rarely collide at exactly the same time (*i.e.*, the OFDM symbols from different transmissions that collide may be misaligned). A natural question is how to decode the collisions when misalignment arises. The following will detail how the Cyclic Prefix (CP) of each OFDM symbol will permit us to properly decode collisions with misalignment and how a phase correction for each subcarrier of the delayed colliding symbol is necessary for proper symbol decoding.

Cyclic Prefix and phase correction: The CP is widely used to handle misalignment when decoding clean signals [6, 20, 40] in

[2] A node can estimate the number of on-going transmissions using preambles and postambles. The presence of a preamble indicates a new transmission and the presence of a postamble indicates a departing transmission. Alternatively, it can estimate the total number of on-going transmissions based on the instantaneous signal strength measured across all channels, the selected channels, and their coefficients.

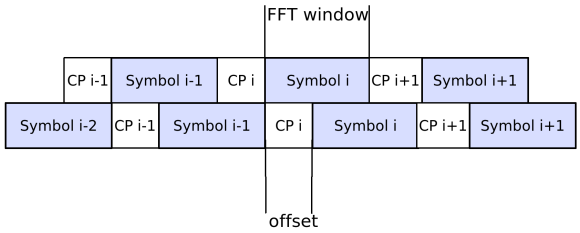


Figure 3: A example shows that the collision offset between two senders is the same as the Cyclic Prefix (CP).

existing wireless systems (*e.g.*, IEEE 802.11a/g/n). The CP works by appending the last N_{CP} samples of an OFDM symbol to the start, and is generally designed to deal with multipath arrivals [6]. For example, if an OFDM symbol contains 512 samples and the CP contains 12 samples. OFDM will append samples 501 through 512 to the start of the OFDM symbol so that the final OFDM symbol contains 524 samples. Even when the FFT window misaligns with the incoming signal by up to 12 samples, it will still contain the entire 512 samples for decoding.

Now the question is how to decode collisions caused by misaligned transmissions. To simplify the description, we consider two symbols involved in the collision. The same technique extends to a collision involving more than two symbols. Figure 3 shows an example. Since OFDM symbols rely on an IFFT to modulate subcarriers, any timing offset that occurs in the channel results in a phase shift after the receiver FFT [40]. First, we begin with the PN sequence correlator to detect the exact time at which each transmission involved in a collision starts. This gives us the value of the misalignment, denoted as τ. We begin the synchronization process with the FFT window that aligns with one of the transmissions but is misaligned with the other transmission by τ.

The IFFT at the OFDM transmitter is expressed as:

$$x(k) = \frac{1}{N}\sum_{n=0}^{N-1} X_n e^{\left(j2\pi k\frac{n}{N}\right)},$$

where N is the number of subcarriers, which is also the size of the IFFT, $x(k)$ is the symbol at sample time k, and X_n is the modulated data symbol on the n^{th} subcarrier.

At the receiver, assuming perfect frequency synchronization, the FFT will be:

$$\begin{aligned} X_{n_0} &= \sum_{k=0}^{N-1} x(k) e^{\left(-j2\pi k\frac{n_0}{N}\right)} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} X_n \left\{\sum_{k=0}^{N-1} e^{\left(j2\pi k\frac{n-n_0}{N}\right)}\right\}, \end{aligned}$$

where the term in the braces denotes the Fourier transform of an exponential, with the result choosing the frequency component at $n = n_0$.

If there is an offset between the received signal and the perceived start of the symbol (*i.e.*, in our case, an offset between the two colliding symbols), we can rewrite our time samples as $x(k-\tau)$, where τ is the timing offset in sample intervals. Then the FFT at the receiver becomes

$$\begin{aligned} \hat{X}_{n_0} &= \sum_{k=0}^{N-1} x(k-\tau) e^{\left(-j2\pi k\frac{n_0}{N}\right)} \\ &= [\sum_{k=0}^{N-1} x(k-\tau) e^{-j2\pi(k-\tau)\frac{n_0}{N}}] e^{(-j2\pi\tau\frac{n_0}{N})} \qquad (4) \end{aligned}$$

$$= X_{n_0} e^{\left(-j2\pi\tau\frac{n_0}{N}\right)}. \qquad (5)$$

Therefore, pre-multiplying the received signal after the FFT with a factor of $e^{\left(-j2\pi\tau\frac{n_0}{N}\right)}$ will synchronize the signals. In Figure 9, we plot the amplitude and angle with and without using this factor to compensate for the offset. As it shows, this compensation restores the misaligned signal.

Therefore, when signals perfectly align, the receiver solves $\forall j$: $R(j,f) = \sum_i h(i,j,f)c(i,f)x(i)$. Now with misalignment by τ_i, the receiver solves

$$\forall j : R(j,f) = \sum_i h(i,j,f)c(i,f)e^{\left(j2\pi\tau_i\frac{n_0}{N}\right)}x(i).$$

Clock synchronization: Without clock synchronization, the offset between two signals can differ by more than the CP duration. In such a case, one signal's FFT window does not contain a complete OFDM symbol from the other signal and will result in interference from loss of orthogonality. As derived in [16], such interference tends to be weak and it is possible to use iterative interference cancellation to decode the signal.

To further enhance the decoding success rate, we can use synchronization to keep the offset between signals within the CP duration. In particular, the sender divides time into equal sized slots and starts transmitting a new frame only at the beginning of a new slot. A clock synchronization protocol is used to synchronize the slot boundaries to be within the CP duration. Note that the default OFDM CP duration is 0.8 microsecond in IEEE 802.11a/g/n. Such level of synchronization can be achieved by existing synchronization protocols for wireless LANs [30, 37]. In particular, [30] develops a source synchronization protocol that can synchronize neighboring senders to within 20 nanoseconds across the operational range of IEEE 802.11 SNRs. As a result, for two senders that are H-hops away from each other, the synchronization error is bounded by $20 \cdot H$ nanoseconds. Since most wireless networks have relatively few hops (*i.e.*, H is small), all senders' clocks can be properly synchronized within the CP. The duration of each slot is chosen to be equal to the time it takes to transmit an OFDM symbol including its CP. Such a choice ensures that if the beginning of a frame is closely aligned with a slot boundary, then all the OFDM symbols in the frame are also aligned with slot boundaries. By starting frame transmissions only at or near slot boundaries, CRMA guarantees that the maximum offset for any colliding OFDM symbols is bounded by the CP duration. Therefore, we can directly apply the above phase correction to effectively compensate for misalignment.

4. QUALNET SIMULATION

4.1 Simulation Methodology

We implement our approach in Qualnet 4.5.1 [28], and compare its performance with the following algorithms:

- CRMA: This is CRMA without virtual flows.
- CRMA-VF: It is CRMA with virtual flows enabled to utilize all available channels. If the number of physical flows is smaller than the number of channels, each flow activates multiple virtual flows. We assume the number of virtual flows for each sender is given a priori. If the number of physical flows are greater than or equal to the number of channels, each node uses just one virtual flow, which is the same as CRMA without virtual flows.

- CSMA/CA: A transmitter carrier senses all channels and transmits on the channels whose total energy is below its clear channel assessment (CCA) threshold. If multiple channels have energy below CCA, the transmitter stripes data across all of the selected channels, which are picked either randomly or based on least-used. If any of them has energy above CCA, the transmitter performs binary backoff. The receiver sends a MAC-layer ACK when it successfully receives the frame. Upon ACK timeout, the transmitter retransmits and doubles the contention window until the maximum contention window is reached. Upon a successful transmission, the contention window is reset to the minimum value.

- Wi-Fi: Different from CSMA/CA, it only transmits on one of the channels whose noise is below the CCA threshold. If multiple such channels exist, one of them is chosen either randomly or based on least-used channel selection.

- Random Access: A sender transmits its signal on a selected channel without performing carrier sense (no DIFS overhead). The contention window is doubled after a failure and reset to the minimum after a successful transmission.

In all schemes, senders and receivers exchange the set of channels to use on a separate control channel, and the receivers know the channel coefficients on all the channels. In addition, all schemes use slot time of 9 μs and SIFS of 16 μs. They all use MAC-layer ACKs (with the same format) and retransmissions to provide resilience. In addition, all schemes except CRMA and CRMA-VF can decode the symbols whose SNR is above the required threshold. We assume perfect synchronization when simulating CRMA.

The parameters used in simulation are: 1000-byte frames (including MAC and PHY header sizes), 16-QAM modulation, 20 MHz total spectrum divided into 10 channels each having 2 MHz, 6.4 Mbps physical data rate on each channel, CCA threshold of -85 dBm (as used in IEEE 802.11a) for 5 GHz short distance networks, and CCA threshold of -91 dBm (as used in IEEE 802.11b) for 700 MHz spectrum long distance networks. We use short distance networks as the default setting. We implement the modulation and demodulation of 16-QAM to map between bits and signals. Qualnet models the propagation delay of all signals.

For each scenario, we conduct 10 random trials. In each trial, flow sources and destinations are picked randomly and the simulation time is 5 seconds. Longer simulations yield similar results. For fair comparison, all the schemes use either random or least-used channel selection (as described in Section 3.1). All simulations use 3 channels per flow. We generate channel coefficients of 10 subcarriers within each channel between a sender and receiver pair using Rayleigh fading (*i.e.*, the real/imaginary parts of the channel responses are modeled by an independent and identically distributed zero-mean Gaussian process). The variance of the fading across different subcarriers is 6 dB according to our indoor measurements of 20 MHz WiFi channels [2]. We compare our scheme with the other schemes using long-distance networks, short-distance networks, varying the number of flows, data rate, and payload size.

4.2 Simulation Results

Long-distance networks: Next, we evaluate the performance of various schemes under long-distance networks using 700 MHz spectrum, which corresponds to white spaces. We use a transmission power of 29 dBm, which gives the transmission range of 19 km and carrier sense range of 42 km. We randomly place nodes in a 40 $km \times 40$ km terrain and pick flows whose senders and receivers are at least 75% of the transmission range (but within the transmission range) from each other to yield long propagation delay. The total data rate over 10 channels is 64 Mbps.

In long-distance networks, the carrier sensing becomes more challenging and more inefficient since the longer propagation delay increases the chance of collisions [24, 26, 32]. Using 802.11 default parameters for DIFS and ACK timeouts is not suitable for long-distance networks, thus we adapt the modified DIFS and ACK timeouts according to [18]. Due to the longer propagation delay, one sender does not receive another sender's signal in time so that CRMA cannot accurately identify the exact number of active senders in the network. Similarly, CSMA/CA and Wi-Fi also suffer from inaccurate carrier sensing.

Figure 4 compares the performance under long-distance networks when all the schemes use random channel selection. We make the following observations. First, we observe that CRMA-VF consistently receives high throughput of 47-48 Mbps across all numbers of physical flows. It utilizes the available 10 channels by creating 10 virtual flows in all cases. As we would expect, when the number of physical flows are smaller than the number of channels, the virtual flow approach is more efficient than CRMA: its throughput is 150-400% higher than CRMA under 2-4 physical flows and 24-66% higher than CRMA under 6-8 physical flows. Its performance is the same as CRMA when the number of physical flows is 10 or higher.

Second, CRMA without virtual flows significantly out-performs CSMA/CA, Wi-Fi, and Random Access. In particular, for a small number of flows (2-4 flows), CRMA out-performs CSMA/CA by 33%-50%, Wi-Fi by 86%-93%, and Random Access by 64%-68%. For a larger number of flows (6-16 flows), CRMA out-performs CSMA/CA by 48%-61%, Wi-Fi by 47%-111%, and Random Access by 16%-73%. CRMA without virtual flows has a lower performance benefit under smaller numbers of flows because it duplicates the same signal over multiple channels, which is wasteful in the low load case.

Third, comparing the existing schemes, we observe that Random Access performs similarly to Wi-Fi and CSMA/CA in low load, but out-performs Wi-Fi and CSMA/CA in high load. Since Random Access uses only one channel per flow whereas CSMA/CA uses multiple channels per flow, its collision probability is lower and performs better than CSMA/CA. Furthermore, Random Access out-performs Wi-Fi by 12%-27%, because Wi-Fi pays for the carrier sense overhead but does not get accurate information in return. In comparison, Random Access does not pay for carrier sense and performs better.

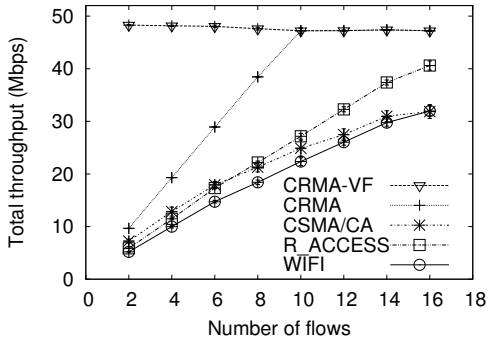


Figure 4: Performance comparison in long-distance networks, using random channel selection.

Short-distance networks: Next, we evaluate the performance of 5 GHz networks by varying the number of flows from 2 to 16. In short-distance networks, the communication and carrier sense

ranges are both 93.3 m, and nodes are randomly placed in a 100 m $\times$ 100 m area. Each flow is between a sender and a receiver that are within communication range from each other. Figure 5 shows the total throughput of each scheme using random channel selection.

First, CRMA-VF consistently achieves the highest throughput across all numbers of flows by effectively utilizing all available channels. It out-performs CRMA by 144%-399% under 2-4 flows, and 12%-57% under 6-8 flows. Its performance is the same as CRMA under 10-16 flows, as we would expect. In addition, CRMA-VF out-performs CSMA/CA by 52-246%, Wi-Fi by 52-670%, and Random Access by 95-685%.

Second, CRMA (without virtual flows) out-performs all the existing schemes under 6-16 flows. In particular, CRMA has 41%-79% higher throughput than CSMA/CA, 52%-113% higher than Wi-Fi, and 94%-137% higher than Random Access. Under 2-4 flows, CRMA performs 56%-82% better than Wi-Fi, and 57%-87% better than Random Access, but comparably or 30% worse than CSMA/CA. The latter is because CRMA does not utilize all the available channels. CRMA-VF effectively addresses this limitation.

Third, comparing the existing schemes, we observe that CSMA/CA performs better than Wi-Fi by 0%-125%, and Wi-Fi performs 0%-28% better than Random Access. CSMA/CA performance degrades under high load case because it uses more channels per flow than Wi-Fi and Random Access and experiences more collisions.

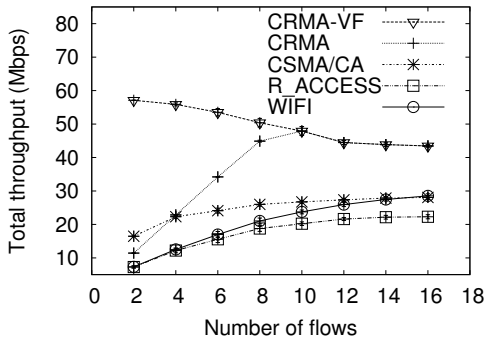


Figure 5: Performance comparison in short-distance networks, using random channel selection.

Effects of payload sizes: Next, we evaluate the impact of payload size on the performance of all schemes. We omit the curve of CRMA-VF, since CRMA-VF has similar performance as CRMA when the number of physical flows is close to the number of channels. We use 8 flows, the 5 GHz network, and vary the payload size from 100 bytes to 1500 bytes. Short packets are common in networks, *e.g.*, TCP/ACK and VoIP. As shown in Figure 6, the performance benefit of CRMA increases as the packet size decreases. For 1000-1500 byte packets, CRMA performs 57-73% better than CSMA/CA, 103-107% better than Wi-Fi, and 139-140% better than Random Access. The improvement increases to 99-130% over CSMA/CA, 134-166% over Wi-Fi, and 150-166% over Random Access for 400-600 bytes. As the packet sizes decrease to 100-200 bytes, corresponding to VoIP and TCP/ACK, the benefit of CRMA increases to 208-308% over CSMA/CA, 248-361% over Wi-Fi, and 208-266% over Random Access.

Effects of data rates: Figure 7 shows the normalized throughput as we vary the total data rate of all channels from 1 Mbps to 64 Mbps. The normalized throughput is defined as the ratio between the achieved application throughput versus the total data rate (*e.g.*, throughput of 1 Mbps under 2 Mbps data rate corresponds to 0.5 normalized throughput). We show normalized throughput because the y-axis spans a significant range due to the large difference in

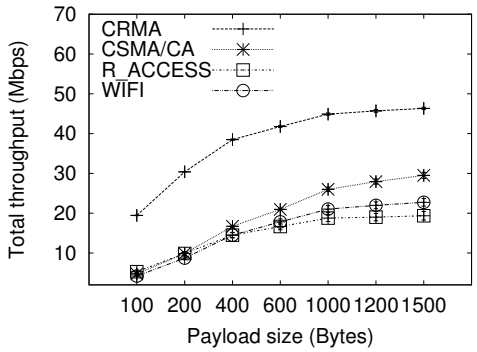


Figure 6: Performance comparison under varying payload sizes, using random channel selection.

data rates, which makes it hard to see the relative performance of various schemes if plotted in absolute numbers.

We make two observations. First, across all data rates, CRMA has higher efficiency than other protocols: its efficiency is 70%-72%, whereas the efficiency of CSMA/CA, Wi-Fi, and Random Access is 44%-54%, 38%-47%, and 34%-38%, respectively. The efficiency of all schemes is considerably lower than 100% due to significant header overhead, since we only count the application payload towards the throughput. Second, the performance gain of CRMA increases with the data rate. For example, it is 30% better than CSMA/CA under 1 Mbps, 33-36% better than CSMA/CA under 2-8 Mbps, 38-50% better than CSMA/CA under 16-32 Mbps, and 58% better than CSMA/CA under 64 Mbps. As the wireless link capacity constantly increases, the carrier sense overhead is becoming increasingly significant. CRMA is an effective approach to harnessing ever-increasing data rates.

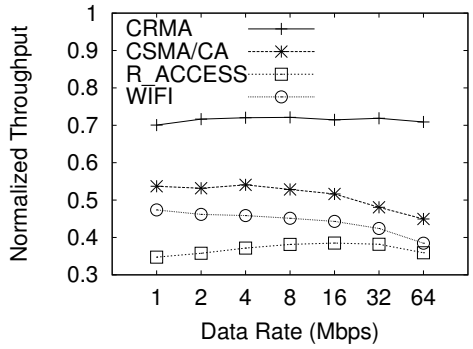


Figure 7: Performance comparison under varying data rate, using least-used channel selection.

5. TESTBED EXPERIMENTS

We implement CRMA on top of the default OFDM implementation in the GNU Radio/USRP platform [39]. Below we introduce our experiment methodology, and present performance results.

5.1 Experiment Methodology

We conduct our experiments in the 5 GHz spectrum space to avoid interference with our campus WLAN network. Unless otherwise mentioned, we use a total of 0.39 MHz spectrum, which consists of 200 subcarriers, each with 1.95 KHz. We partition the spectrum into two channels when performing experiments of two flows. We use BPSK as the modulation scheme. To evaluate the performance, it is necessary to have collisions. We generate collisions in the following way. We first separately log the signals in

the time domain without collisions. Then we add them up to mimic collisions and give the resulting signals to our decoder. We feed the decoded signals output by our decoder to the GNU Radio receiver to measure the packet delivery rate (PDR) and bit error rate (BER). PDR is defined as the fraction of frames that our algorithm can successfully decode.

The decoding ratio is significantly affected by signal strength. GNU Radio does not have a unit for the received signal strength, so we use a spectrum analyzer to calibrate the received signal and compute SINR.

5.2 Performance Results

Accuracy of preamble detection: We first evaluate the accuracy of preamble detection using signal correlation described in Section 3. We have a pair of senders transmit frames back-to-back without any carrier sensing. The receiver runs the correlation algorithm to detect the preambles. The preamble has one OFDM symbol. We quantify the accuracy of preamble detection using *false positive* and *false negative* under different values of SINR. A false positive indicates that our algorithm detects a preamble that is actually not present and a false negative indicates that our algorithm fails to detect a preamble that is actually present.

In order to identify a false positive and a false negative, we obtain the ground truth as follows. We start the first sender A and then record the clean preamble from it and measure the inter-packet arrival time, which remains constant throughout the experiment due to the absence of carrier sense. We then start the sender B, which collides with A for 500-600 times at each value of SINR. We then stop the sender A and measure the inter packet arrival time from B. We can now backtrack into the collisions of A and B, and figure out all the positions where A and B's packets would have arrived. This is then compared against the preambles detected based on correlation, where we automatically detect a spike in correlation whenever the correlation value exceeds a threshold, which is set to be proportional to the amplitude of the clean preamble received from the sender.

As shown in Figure 8, the false positives and false negatives are 0 when the SINR is as low as -2. Further reducing SINR slightly increases the false positives while the false negatives remain 0. We can adjust the threshold used to detect correlation spike to tradeoff between false positives and false negatives.

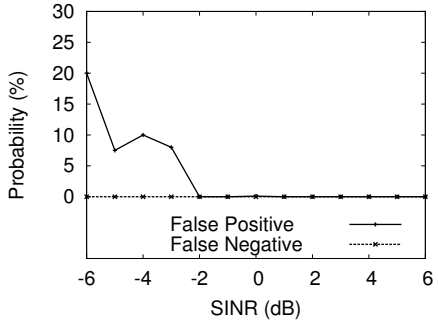


Figure 8: Accuracy of preamble detection.

Accuracy of sender and receiver ID detection: In this experiment, we quantify the accuracy of identifying the source using the correlation technique described in Section 3.2. The detection accuracy of identifying the receiver should be the same. We assign each sender ID with a unique PN sequence and add it right after the frame preamble. The receiver first records the PN sequence received from each of the senders during a clean transmission. Let s_i denote the received PN sequence from sender i. Whenever the receiver detects a preamble, it runs correlation against all s_i's and determines the source as the one that gives the highest spike in the correlation. We vary SINR from -6 to 6 dB. At each SINR value, we use around 400 packets to quantify the accuracy. In all cases, the correlation-based detection correctly identifies the source ID.

Effects of updating channel estimates: The wireless channel varies over time. We find that without updating channel estimates the packet delivery rate is low and the BER rises quickly with increasing bit positions in the frame because the channel estimation of the later bits becomes inaccurate.

Channel update interval (# OFDM symbols)	High SNR	Low SNR
20	60%	29%
10	71%	57%
5	95%	80%
3	98%	95%
1	98%	96%

Table 1: Packet delivery rate with varying update intervals. 1000-byte packets, SINR=3 at the high SNR node and -3 at the low SNR node.

Table 1 summarizes the packet delivery rates with varying intervals of updating the channel estimates. Reducing the update period from 20 OFDM symbols to 10 OFDM symbols improves packet delivery rate by 10-30%; further increasing it to 3 OFDM symbols improves packet delivery rate to 95-98%.

Effect of SINR on decoding accuracy: Table 2 and Table 3 show the packet delivery rates for 500-byte and 1000-byte packets when we update the channels every 3 OFDM symbols. We observe that CRMA can decode packets from both high and low SNR senders. The decoding probabilities are lower when the SNR is close to zero due to possible signal cancellation. When the receiving signal strength is larger than 7 dB, the capture effect dominates. In such cases, the stronger signals have high decoding rates, but the weaker signals have low decoding rates. Other collision decoding schemes, such as [16], have similarly low decoding rates in this case (*e.g.*, its BER is 0.016 when the SNR is -8.7 dB, which is close to 0 frame delivery rate for 500-byte or 1000-byte frames). We can use partial packet recovery to extract correct symbols from partially correct frames to improve performance.

SINR (dB)	High SNR	Low SNR
0	42%	42%
1	95 %	90%
3	98%	95%
5	99%	90%
7	97%	0%

Table 2: Packet delivery rate with varying SINR. 500-byte packets, channel estimates updated every 3 OFDM symbols, where SINR in the table is reported by the high SINR node and the low SINR node sees $-$SINR.

Effects of misalignment: In CRMA, we can decode collisions with offsets up to the length of the Cyclic Prefix (CP). In practice, FFT itself is robust against a few misalignments, so the actual misalignment that can be tolerated is slightly higher than the CP length. We use a CP of length 128 in the following experiments.

First, we show our approach in Section 3.7 can correctly decode misaligned transmissions. Specifically, a receiver receives two colliding signals A and B, and A arrives earlier than B by 10 sam-

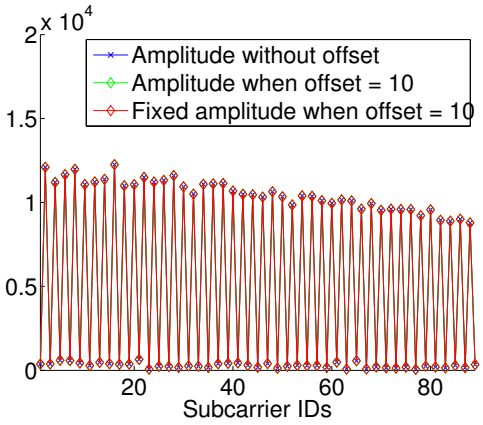
SINR (dB)	High SNR	Low SNR
0	25%	25%
1	98%	95%
3	98%	95%
5	100%	98%
7	99%	0.09%

Table 3: Packet delivery rate with varying SINR. 1000-byte packets, channel estimate updated every 3 OFDM symbols, where SINR in the table is reported by the high SINR node and the low SINR node sees −SINR.

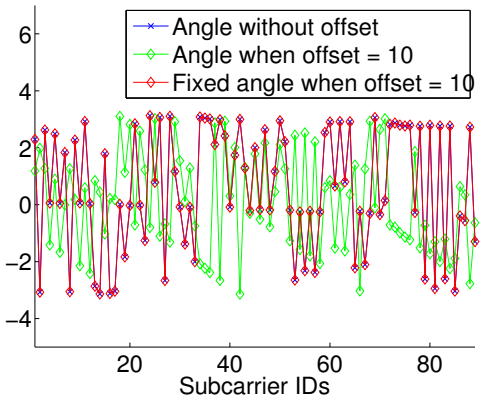
ples (there are 512 samples in our FFT windows) in the time domain. Let $FFT(B)$ and $FFT(A+B)$ denote the signals after FFT for B and the colliding signals, respectively. If the offset is zero, one should expect that $FFT(A+B) = FFT(A) + FFT(B)$ or $FFT(B) = FFT(A+B) - FFT(A)$. When the offset is not zero, $FFT(B) \neq FFT(A+B) - FFT(A)$ as shown by the curves marked as 'amplitude/angle without offset' and 'amplitude/angle when offset = 10' in Figure 9, respectively. However, by using Equation 5 the offset signal can be readily fixed, and so the curves marked as 'fixed amplitude/angle when offset = 10' completely overlap with 'amplitude/angle without offset'.

Note that the amplitudes in all cases are almost the same but the angle differs by a significant amount. After applying our approach, the new signal has almost the same amplitude and angle as the original signal without any offset (*i.e.*, the first and third curves overlap in both plots). This indicates our approach can correctly decode misaligned signals.

We further quantify the accuracy of decoding two colliding signals with misalignment ranging from 0 to 140. As shown in Figure 10, CRMA can accurately decode collisions up to 140 sample offsets. Beyond that, the FFT window does not contain complete OFDM symbols and has difficulty in decoding the symbols, as we would expect.



(a) Amplitude



(b) Angle

Figure 9: Amplitude and angle before and after using Equation 5 to fix the impact of collision offset. The offset is 10. In the first subfigure, all three curves almost entirely overlap since the amplitude of the signals before and after having an offset is quite similar, and so the difference is not visible in this small plot. Similarly, in the second subfigure, the angle without offset and the angle that is fixed using our technique almost entirely overlap which means that Equation 5 is working well. Note however, the angle with offset is not the same.

6. RELATED WORK

Our work is related to the research on (i) decoding collisions, (ii) CDMA, and (iii) channel assignment and channel hopping, which we briefly describe below.

Decoding collisions: The ability to simultaneously receive multiple frames at a wireless node has been an active research topic. For example, [7] and [38] provide theoretical analysis of potential gain of multiple frame reception using advanced signal processing and antenna array techniques. [41] proposes successive interference cancellation to decode collisions. The authors in [10] build a prototype of SIC on ZigBee and experimentally demonstrate its effectiveness. A major limitation of SIC is that it is applicable only when the information rate is lower than what the current SNR can support. To overcome this limitation, ZigZag [8] develops a novel approach to decode using multiple collisions. It exploits varied offsets in different collisions. It first decodes all interference-free symbols using a standard decoder and then re-encodes those symbols and subtracts them from the collision that overlaps with those symbols. It iteratively applies this technique to decode entire frames. It has a known issue of error propagation since decoding future symbols relies on the correct decoding of previous symbols. Different from ZigZag, our decoding algorithm avoids such error propagation since we decode individual symbols independent of previous symbols. Analog network coding proposed in [13] can also decode collisions, but it requires a receiver to have one of the two colliding frames, which does not hold in general. Moreover, SIC, ZigZag, and analog network decoding are different ways to decode collisions whereas CRMA leverages the ability of decoding collisions to build a MAC protocol.

[9] applies the combination of interference alignment and interference cancellation to decode colliding signals in MIMO networks. [37] exploits MIMO to provide spatial multiple access in wireless LANs and decode colliding signals. Both [9] and [37] require the wireline network connecting APs for coordination and work only under MIMO.

CDMA: Our coding-based spectrum allocation is related to CDMA. However, the existing CDMA schemes have several limitations that make them either impractical or inefficient. Synchronous CDMA assigns users with orthogonal codes and other users' transmissions do not cause any interference. Its major limitation is that the orthogonality of codes is only guaranteed when all the transmissions are synchronous, since it is mathematically impossible to generate orthogonal codes with arbitrarily random offsets [4, 42]. Asynchronous CDMA addresses this issue by encoding the signals using "pseudo-random" or "pseudo-noise" (PN) sequences. They do not require synchronization. However, different PN sequences are statistically uncorrelated and signals coded with other PN sequences

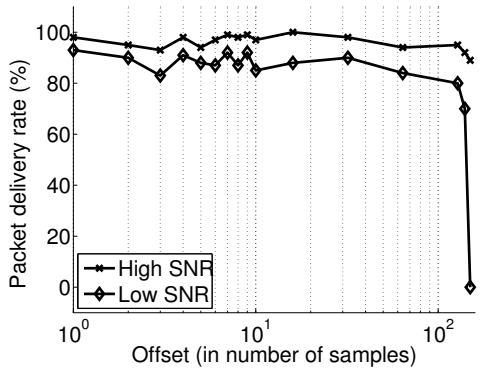


Figure 10: Packet delivery rate for two senders. The SINR between the two received signals is 3, and the packet size is 1000 bytes.

produce Multiple Access Interference (MAI) [4, 42]. Therefore, unlike synchronous CDMA, other users' signals interfere with the desired signal and the amount of interference is proportional to the number of users and the bit-error-rate (BER) deteriorates quickly with the number of active transmitters. The reason that asynchronous CDMA suffers from interference is that the receiver has no knowledge about the other signals and simply treats them as noise. However, CRMA explicitly takes advantage of information about other signals and can cancel them out to achieve high reception quality.

Channel assignment and channel hopping: There are significant works on channel assignment (*e.g.*, [15, 21, 33]) and channel hopping (*e.g.*, SSCH [1] and MAXchop [22]). All these approaches aim to ensure no more than one transmission can be active on any channel. CRMA provides a new perspective on spectrum sharing by using random coding and can more effectively support spectrum sharing in a distributed network.

Other capacity enhancement techniques: FICA [37] also recognizes the significant contention overhead, especially as the data rate increases. It improves capacity by dividing a channel into smaller subchannels and letting users contend and use subchannels. Meanwhile it reduces contention overhead by replacing time-domain backoff with frequency-domain backoff. The efficiency of FICA depends on how often users choose unique subchannels. FICA lets each flow use one random subchannel, which may yield high collisions on some subchannels while leaving other subchannels under-utilized. In comparison, CRMA can more effectively utilize channels using random coding.

7. CONCLUSION

In this paper, we propose CRMA, which views the OFDM physical layer as multiple orthogonal but sharable sub-carriers, and leverages random linear codes to share the available spectrum. Compared with CSMA, Random Access, and Wi-Fi, CRMA can achieve high efficiency without fine-grained coordination and presents an interesting design point in spectrum sharing. Our simulation results show that CRMA significantly out-performs the alternative schemes in a wide range of scenarios. Our testbed implementation and experiments using USRP further show the feasibility of our approach.

This paper is just a first step towards a new direction of spectrum sharing. There are many open questions that remain to be answered. First, CRMA requires fairly accurate channel estimation. This is feasible in static environments. In mobile networks, the channel condition can change rapidly and it is important to quickly adapt the channel estimate and enhance the robustness of the decoding algorithm against estimation error. Second, an increasing data rate places more stringent requirements on the accuracy of the channel estimation and decoding algorithm as well as decoding efficiency. We are interested in evaluating the performance under higher data rates in the future. Third, CRMA currently incurs overhead in order to ensure the number of active transmissions is no more than the number of channels being used. We plan to explore decoding algorithms that can gracefully degrade as the number of active transmissions increases beyond the number of channels. Fourth, we are interested in developing a full-fledged prototype implementation and more extensively evaluating its performance in a wide range of scenarios.

Acknowledgments

This research is supported in part by NSF grants CNS-1017549, CNS-0916106, CNS-0916309, and CNS-0546755. Tianji Li is supported by Irish Research Council for Science, Engineering and Technology. We thank Kun Tan for answering our questions about the physical layer, and thank Romit Roy Choudhury and the anonymous reviewers for their valuable comments.

8. REFERENCES

[1] P. Bahl, R. Chandra, and J. Dunagan. SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proc. of ACM MobiCom*, 2004.

[2] A. Bhartia, Y.-C. Chen, S. Rallapalli, and L. Qiu. Harnessing frequency diversity in wi-fi networks. In *Proc. of ACM MobiCom*, 2011.

[3] R. E. D. Borth and B. Oberlie. Considerations for successful cognitive radio systems in US TV white space. In *Proc. of DySpan*, 2008.

[4] Code division multiple access. `http://en.wikipedia.org/wiki/Code_division_multiple_access`.

[5] Y. C. Cheng, J. Bellardo, P. Benk, A. C. Snoeren, and G. M. Voelker. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *Proc. of ACM SIGCOMM*, 2006.

[6] Cylic prefix. `http://en.wikipedia.org/wiki/Cyclic_prefix`.

[7] S. Ghez, S. Verdu, and S. C. Schwartz. Stability properties of slotted Aloha with multipacket reception capability. *IEEE Transcactions on Automatic Control*, July 1988.

[8] S. Gollakota and D. Katabi. ZigZag decoding: Combating hidden terminals in wireless networks. In *Proc. of ACM SIGCOMM*, 2008.

[9] S. Gollakota, S. D. Perli, and D. Katabi. Interference alignment and cancellation. In *Proc. of ACM SIGCOMM*, Aug. 2009.

[10] D. Halperin, T. Anderson, and D.Wetherall. Practical interference cancellation for wireless LANs. In *Proc. of ACM MobiCom*, 2008.

[11] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *Proc. of ACM SIGCOMM*, 2010.

[12] K. Jamieson, B. Hull, A. K. Miu, and H. Balakrishnan. Understanding the real-world performance of carrier sense. In *Proc. of ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND)*, Aug. 2005.

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

[14] S. Khurana, A. Kahol, and A. P. Jayasumana. Effect of hidden terminals on the performance of IEEE 802.11 MAC protocol. In *Proc. of LCN*, 1998.

[15] Y. Lee, K. Kim, and Y. Choi. Optimization of AP placement and channel assignment in wireless LANs. In *Proc. of IEEE LCN*, Nov. 2002.

[16] L. E. Li, K. Tan, Y. Xu, H. Viswanathan, and Y. R. Yang. Remap decoding: Simple retransmission permutation can resolve overlapping channel collisions. In *Proc. of ACM MobiCom*, 2010.

[17] Z. Li, W. Xu, R. Miller, and W. Trappe. Securing wireless systems via lower layer enforcements. In *Proc. of 5th ACM Workshop on Wireless Security (WiSe'06)*, Sept. 2006.

[18] Air stream technical guide: ACK timeouts and the effects on distance links. http://www.air-stream.org.au/ACK_Timeouts.

[19] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik. Radio-telepathy: Extracting a secret key from an unauthenticated wireless channel. In *Proc. of ACM MobiCom*, Sept. 2006.

[20] H. Minn, M. Zeng, and V. Bhargava. On timing offset estimation for OFDM systems. *Communications Letters, IEEE*, 4(7):242 –244, July 2000.

[21] A. Mishra, V. Brik, S. Banerjee, S. Aravind, and W. Arbaugh. A client-driven approach for channel management in wireless LANs. In *Proc. of IEEE INFOCOM*, Apr. 2006.

[22] A. Mishra, V. Shrivastava, D. Agarwal, S. Banerjee, and S. Ganguly. Distributed channel management in uncoordinated wireless environments. In *Proc. of ACM MobiCom*, Sept. 2006.

[23] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan. Load-aware spectrum distribution in wireless LANs. In *Proc. of ICNP*, Oct. 2007.

[24] S. Nedevschi, R. Patra, S. Surana, S. Ratnasamy, L. Subramanian, and E. Brewer. An adaptive high-performance MAC for long-distance multihop wireless networks. In *Proc. of ACM MobiCom*, Sept. 2008.

[25] L. M. S. C. of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.

[26] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer. WiLDNet: design and implementation of high-performance wifi-based long distance networks. In *Proc. of ACM/USENIX NSDI*, 2007.

[27] N. Patwari and S. K. Kasera. Robust location distinction using temporal link signatures. In *Proc. of ACM MobiCom*, Sept. 2007.

[28] The Qualnet simulator from Scalable Networks Inc. `http://www.scalable-networks.com/`.

[29] H. Rahul, F. Edalat, D. Katabi, and C. Sodini. FARA: Frequency-aware rate adaptation and MAC. In *Proc. of ACM MobiCom*, Sept. 2009.

[30] H. Rahul, H. Hassanieh, and D. Katabi. SourceSync: A cooperative wireless architecture for exploiting sender diversity. In *Proc. of ACM SIGCOMM*, 2010.

[31] H. Rahul, N. Kushman, D. Katabi, C. Sodini, and F. Edalat. Learning to share: Narrowband-friendly wideband wireless networks. In *Proc. of ACM SIGCOMM*, Aug. 2009.

[32] B. Raman and K. Chebrolu. Design and evaluation of a new MAC protocol for long-distance 802.11 mesh networks. In *Proc. of ACM MobiCom*, 2005.

[33] E. Rozner, Y. Mehta, A. Akella, and L. Qiu. Traffic-aware channel assignment in enterprise wireless LANs. In *Proc. of ICNP*, Oct. 2007.

[34] J. Schiller. *Mobile Communications*. Addison Wesley, 2003.

[35] S. Sen, R. R. Choudhury, and S. Nelakuditi. CSMA/CN: Carrier sense multiple access with collision notification. In *Proc. of ACM MobiCom*, Sept. 2010.

[36] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, D. Papagiannaki, S. Keshav, and A. Mishra. CENTAUR: Realizing the full potential of centralized WLANs using a hybrid data path. In *Proc. of ACM MobiCom*, Sept. 2009.

[37] K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, J. Zhang, and Y. Zhang. Fine grained channel access in wireless LAN. In *Proc. of ACM SIGCOMM*, 2010.

[38] L. Tong, Q. Zhao, and G. Mergen. Multipacket reception in random access wireless networks: From signal processing to optimal medium access control. *IEEE Commnications Magazine*, Nov 2001.

[39] USRP. `http://www.ettus.com/products`.

[40] J. van de Beek, M. Sandell, and P. Borjesson. ML estimation of time and frequency offset in OFDM systems. *Signal Processing, IEEE Transactions on*, 45(7):1800 –1805, July 1997.

[41] S. Verdu. *Multiuser Detection*. Cambridge University Press, 1998.

[42] A. J. Viterbi. *CDMA: Principles of Spread Spectrum Communication*. Prentice Hall, Apr. 1995.

[43] L. Yang, W. Hou, L. Cao, B. Y. Zhao, and H. Zheng. Supporting demanding wireless applications with frequency-agile radios. In *Proc. of NSDI*, 2010.

[44] Y. Yuan, P. Bahl, R. Chandra, T. Moscibroda, and Y. Wu. Allocating dynamic time-spectrum blocks in cognitive radio networks. In *Proc. of ACM MobiHoc*, 2007.

[45] J. Zhang, M. H. Firooz, N. Patwari, and S. K. Kasera. Advancing wireless link signatures for location distinction. In *Proc. of ACM MobiCom*, Sept. 2008.