# RnR: Reverse & Replace Decoding for Collision Recovery in Wireless Sensor Networks

Dali Ismail[†*], Mahbubur Rahman[†*], Abusayeed Saifullah[†*], Sanjay Madria[‡]
Department of Computer Science, Wayne State University, Detroit, MI, USA[†]
Missouri University of Science & Technology, Rolla, MO, USA[‡]
{dali.ismail, r.mahbub, saifullah}@wayne.edu[†], madrias@mst.edu[‡]
[*] Co-primary author

***Abstract*—Interference between concurrent transmissions causes severe performance degradation in a wireless network. This paper addresses interference cancellation to enable simultaneous packet receptions at a node with a single radio in Wireless Sensor Networks (WSN). Interference cancellation is particularly important for WSN as most of its applications rely on *convergecast* where all the traffic in the network is delivered to a base station leading to a lot of packet collisions. Existing solutions for collision recovery make simplified assumptions such as the availability of one of the collided packets, repeated collisions of the same packets, and the ability to identify the collided packets before recovering them which do not hold for WSNs and most wireless networks. In this paper, we propose a novel collision recovery method called *Reverse and Replace Decoding (RnR)* for WSNs. RnR entails a physical-link layer design to exploit the raw samples of the colliding signals. It does not rely on the assumptions made in existing work, and can recover all packets from a *single* collision. To demonstrate its feasibility, we have implemented RnR using GNU Radio on USRP devices based on IEEE 802.15.4 network. Our experiments on a 6-node testbed demonstrate that RnR can successfully decode packets in 95% cases of collisions, and improves the correctly packet decoding rate up to 97.5% compared to standard decoders in the case of collisions. Also, our simulation based on GNU Radio simulator using 25 nodes shows that RnR achieves 4x higher throughput compared to the state-of-the-art collision recovery mechanisms.**

## I. Introduction

In a wireless network, concurrent transmissions from different devices sharing the same channel collide at the receiver, causing no successful packet reception. Such *interference* between concurrent transmissions is a well-known problem that causes severe performance degradation in a wireless network. Many networks (e.g., IEEE 802.11 [1], IEEE 802.1.5.4 [2]) adopt Carrier Sense Multiple Access (CSMA) to avoid collisions [3], [4], [5]. CSMA senses the channel before transmitting, and backs-off for some amount of time if it detects any traffic on the channel, and attempts to retransmit thereafter. However, in many cases, CSMA cannot avoid collisions, for example, in the presence of hidden terminals [6] which is quite common in most wireless networks. Such collisions in CSMA protocols severely reduce the throughput. While networks such as IEEE 802.11 have the option of adopting RTS/CTS [7] to reduce collisions, it introduces significant overhead to the network and reduces the effective throughput. In most 802.11 nodes, RTS/CTS is disabled by default. In low power wireless networks such as IEEE 802.1.5.4 and WirelessHART [8], this method is impractical and is never adopted.

We address interference cancellation in wireless sensor networks (WSNs) such as those based on IEEE 802.15.4 and WirelessHART. Interference cancellation is particularly important for WSN as most of its applications rely on *convergecast* [9] where all the traffic in the network is delivered to a base station leading to a lot of packet collisions. Collision recovery has been studied in many early works. *Capture effect* [10] can recover at most one packet and only if its Received Signal Strength (RSS) is significantly higher (by 1-3dB) than that of the other colliding signal/s, and in cases (based on radio design) if it arrives before the others. A link layer solution cannot recover all collided packets as it requires the raw signal sampled at the physical layer, thus making collision recovery **challenging** for network engineers. Existing physical layer solutions for collision recovery in wireless networks make simplified assumptions such as the availability of one of the collided packets [11], [12], [13], [14], and repeated collisions of the same packets and the ability to identify the collided packets before recovering them [15] that do not hold in WSN as well as in most wireless networks. The broadcast scheme proposed in [18] adopted a collision recovery technique for identical broadcast transmission only, thus limiting its applicability for packet collisions in WSNs. A recently proposed recovery technique [16] from a single collision for ZigBee [17] radio only leverages on discerning an exponential (in number of packets that collide) number of amplitude levels, thus being subject to high bit error that makes it less effective in practice even when just two packets collide. It cannot be used if more than four packets collide. In WSN convergecast, a large number of packets may collide, thus requiring a new collision recovery mechanism.

In this paper, we propose a new collision recovery method called ***Reverse* & *Replace Decoding (RnR)*** for WSNs. RnR entails a physical-link layer design to exploit the raw samples of the colliding signals. It does not rely on the assumptions made in existing work, and can recover all packets from a ***single*** collision. It is bootstrapped only after a collision is detected, and starts by first extracting a collision-free chunk

from the collided signal. This chunk is then subtracted from the collision to retrieve the collided ones. This iterative process continues until the collided packets are recovered. To decode all packets from a single collision, the key idea in the proposed physical-link layer design is to replace the CRC (Cyclic Redundancy Check) of a packet by a longer error correction code, thus requiring packet augmentation. Since WSN packets, in practice, are much shorter than their maximum carriable size [18], augmenting a packet length is easily affordable in WSN without exceeding channel capacity. Hence, our RnR design is specifically focused on WSN. However, any network based on digital modulation that can afford such packet augmentation can adopt RnR for collision recovery.

To demonstrate the feasibility of recovering from collisions, we have implemented RnR in GNU Radio [19] on Universal Software Radio Peripheral (USRP) [20] devices for IEEE 802.15.4 networks. We have experimented on a 6-node testbed and also through GNU Radio simulator for larger scale tests. The experiments demonstrate that RnR can successfully decode packets in $\geq$ 95% cases. RnR also achieves 4x higher throughput over the state-of-the-art collision recovery mechanisms [16], [15]. Our extensive experiments also demonstrate that, in the case of collisions, RnR improves the correctly packet decoding rate up to 97.5% over standard decoders.

The rest of the paper is organized as follows. Section II reviews related work. Section III provides a detailed description of the proposed RnR decoder. Section IV describes our implementation of the RnR decoder. Section V describes the experimental results. Section VI describes the simulation results. Section VII concludes the paper.

## II. Related Work

Collision recovery was studied in various wireless domains under various simplified assumptions that do not hold for most wireless networks [11], [12], [13], [14], [15], [21], [22], [23]. Successive interference cancellation [21], [22], [23] works only if the radio's bit rate is significantly reduced from what its SNR (signal to noise ratio) allows and if the interfering senders have significantly different powers or precoded signatures. It also demands prior scheduling and known users, and is basically designed for cellular networks. Analog network coding [13], XORs [14], interference alignment [11], and full duplex [12] require a receiver to have one of the two colliding packets. Hence, they are not applicable for WSNs where packets from different senders are unknown a priori. SNOW [24] base station receives multiple packets using an OFDM based physical layer design that is different from the traditional WSN devices and not yet adopted in the commercially available WSN devices.

Obviating most of the above assumptions, ZigZag [15] was designed as a modulation independent decoding for 802.11 networks. As shown in Fig. 1 for 2 packets $P_a$ and $P_b$, ZigZag first decodes all interference-free samples using a standard decoder and then re-encodes those symbols and subtracts them from the collision that overlaps with those. It iteratively applies this technique to decode entire frames. However, to resolve one

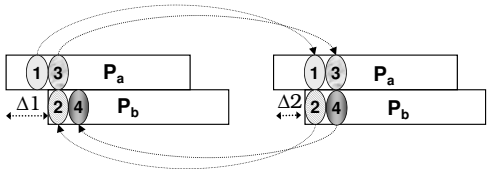



Fig. 1. **ZigZag decoding**: first decodes interference free chunk 1 in first collision. It subtracts chunk 1 from second collision to decode chunk 2, which it then subtracts from first collision to decode chunk 3, so on [15].

collision of $m$ packets, it needs at least $m$ repetitions of the collision, every time with different arrival time offsets between the packets (i.e. $\Delta_1 \neq \Delta_2$ must hold in Fig. 1). Depending on at least $m$ collisions reduces throughput, consumes huge energy, and is not worth adopting in WSN where nodes are resource constrained. Even if it is adopted at the cost of energy, to merge the chunks of a packet from different collisions, the receiver should correctly identify which packets the chunks across collisions belong to before recovering them. This assumption does not hold in WSN and most wireless networks. In fact, ZigZag would require to explore all possible combinations of the chunks which is exponential (in terms of the number of chunks as well as $m$) to recover a packet as its chunks cannot be identified from multiple collisions. In contrast, RnR is capable of recovering all packets from a single collision, and hence does not suffer from this problem.

While mZig [16] is designed to recover packets from a single collision in ZigBee networks, it depends on discerning $2^m$ amplitude levels if $m$ packets collide, thereby being subject to high bit errors. Since amplitude of a signal is highly susceptible to noise and obstacle, it is difficult to distinguish many amplitude levels in practice even if the signals are received correctly. Thus, mZig is less effective in recovering even when just two packets collide. Also, mZig will not work when the two bits to be separated from the different packets have an equal amplitude as explained below. It exploits the half-sine pulse shape of baseband signal in ZigBee devices. Thus, when 2 packets collide, it has to discern 4 amplitude levels of the bits. Considering $\alpha$ and $\beta$ as the amplitudes from the two packets, the 4 signal levels are $\alpha+\beta, \alpha-\beta, -\alpha+\beta, -\alpha-\beta$. Assuming $\alpha > \beta$, if the level $\alpha-\beta > 0$, then $\alpha$ is decoded as '1' and $\beta$ is decoded as '0', and if the level $-\alpha+\beta < 0$, then $\alpha$ is decoded as '0' and $\beta$ is decoded as '1'. Hence, when the bits from 2 packets have an equal amplitude, the signal levels $\alpha-\beta$ and $-\alpha+\beta$ will not allow us to determine which packet's bit is '0' and which packet's bit is '1'. Thus the decoding will fail. Most importantly, mZig does not work when $m > 4$. In WSN convergecast, a large number of packets may collide, thus requiring a new collision recovery mechanism. Finally, mZig is not applicable to any network other than ZigBee [17]. In contrast, RnR does not suffer from the above limitations as it does not depend on discerning amplitude levels. Specifically, RnR is capable of recovering all packets successfully from a collision of any number of packets (with no limitation on $m$). Additionally, RnR is applicable to any physical layer of WSN.

## III. Reverse & Replace Decoding

This section presents our proposed ***Reverse & Replace decoding (RnR)*** for recovering collided packets in WSNs. We first present the underlying challenges in collision recovery and the key principle in RnR design. This is followed by a detailed technical description of RnR and the design considerations.

### A. Key Design Principle

When two or more packets collide at a receiver's radio, the radio cannot recover any of those packets. Recovering the collided packets requires further decoding of the composite signal of the collision and is challenging. Also, for different modulation techniques, the decoding for collision resolution may be different. It is also impacted by the underlying radio design. As an example, we provide an overview of the off-the-shelf radios based on IEEE 802.15.4 [25] and WirelessHART [8]. During the synchronization header decoding mode while receiving a packet, its radio searches for preambles and start frame delimiter with the strongest RSS [26], [25], [8]. After this, the radio generates an interrupt and locks to payload reception mode, and no more searches for preambles. Therefore, ***capture effect*** [27], [10] can recover the stronger packet if it comes before the radio locks to a weaker packet's payload reception mode. If the stronger packet comes later, it may be possible to make the radio search for new preambles and resynchronize to it [28]. However, in either case, only the strongest packet can be recovered and only if its RSS is significantly higher (by 1–3dB based on modulation) than the other signal/s. A link layer solution cannot recover all collided packets, as it requires the raw signal sampled at the physical layer. Hence, the proposed RnR decoder involves a physical-link layer design to recover packets from collisions.

As discussed in Section II, existing physical layer solutions for collision recovery [13], [15], [16], [21], [22], [23] make simplified assumptions that do not hold for WSN and most wireless networks. As a quick recap, network coding [13], [14] requires a receiver to know one of the two colliding frames. mZig [16] depends on discerning $2^m$ amplitude levels if $m$ packets collide, thereby being subject to high bit errors. It does not work when the two bits from the different packets have the same amplitude. Also, it does not work when $m > 4$. Finally, mZig is not applicable to any network other than ZigBee [17]. ZigZag [15] decodes frames in 802.11 networks by iteratively subtracting collision free chunks from multiple collisions. However, to resolve one collision of $m$ packets, it needs at least $m$ repetitions of the collision, every time with different arrival time offsets between the packets. This is not worth adopting in WSNs. Even if it is adopted at the cost of energy, to merge the chunks of a packet from different collisions the receiver should correctly identify which packet each chunk belongs to before recovering them. This assumption does not hold in WSNs. The dependence on multiple collisions is the root of this problem. This motivates a collision recovery mechanism that should depend on a single collision. Our proposed RnR decoding hence depends on a single collision while following the principle of ZigZag in that it also first finds a collision free chunk. But, unlike ZigZag, it exploits that chunk to decode all packets from a *single* collision. RnR is invoked only upon a collision, and is applicable to any physical layer of WSNs.

Now we review the structure of the composite signal of collided packets which will be exploited by RnR. A wireless signal is represented as discrete complex values [29]. A received signal is represented as a sequence of samples spaced by sampling interval $T$. If $X[n]$ is the complex value of the $n$-th sample at the transmitter, the received sample is given by

$$Y[n] = HX[n] + W[n]$$

where $H = he^{\gamma}$ is the channel parameter with channel attenuation $h$ and phase shift $\gamma$, and $W[n]$ is the noise. If two senders $a$ and $b$ transmit concurrently, the received sample can be expressed as

$$Y[n] = H_a X_a[n] + H_b X_b[n] + W[n]$$

where $H_a$ and $H_b$ are the channel parameters; $X_a[n]$ and $X_b[n]$ are the transmitted samples of $a$ and $b$, respectively. In RnR design, we exploit the above structure of collided signals which helps recover a chunk of one packet from a collision by subtracting an already recovered chunk of the other.

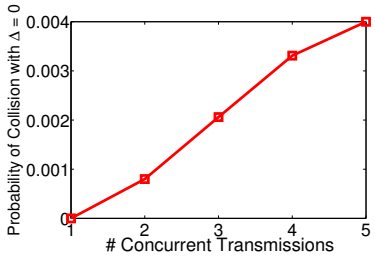



Fig. 2. **Fraction of collisions with $\Delta = 0$ in experiment.**

### B. Core Technique of RnR Decoding

We detail the RnR decoding technique first for 2 packets $P_a$ (transmitted by $a$) and $P_b$ (transmitted by $b$) with arrival time offset $\Delta$. Two senders' processing, clock drifts, back-offs, and distances from the receiver cause $\Delta \neq 0$, allowing an interference-free chunk to exist. Our 10-day long experiments show that the probability of collisions with $\Delta = 0$ tends to be 0 as shown in Fig. 2. We performed an experiment based on 802.15.4 networks in an indoor environment. We used our 6-node testbed (5 transmitters and 1 receiver). All the transmitting and receiving nodes were fixed in different locations. Every transmitter (Tx) is at a distance of approximately 25 ft from the receiver and all 5 transmitters try to send to one receiver at 0 dBm transmission power. Each transmitter sends a 129-byte packet (32 bits preamble and 1000 bits payload) every 4ms consecutively. We record the data for 5 hours a day for a period of 10 days considering 500,000 packets (100,000 packets for each Tx). Fig. 2 shows the fraction of total collisions that experience $\Delta = 0$ under varied number of concurrent transmitters. As the figure shows, the probability of collisions with $\Delta = 0$ is very low. Making the senders do a small random back-off before transmitting can further reduce

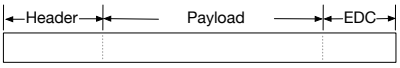



Fig. 3. **A WSN packet structure.**

this probability. Thus a collision-free chunk is almost certain to exist. RnR exploits this collision-free chunk.

To exploit a collision-free chunk to decode all packets from a single collision, a packet needs a little preprocessing before transmission. As shown in Fig. 3 for WSN, a MAC layer packet (frame) consists of three segments: header, payload, and EDC (error detection code). The EDC is usually 2-byte CRC. As preprocessing before transmission, we replace the CRC with a new and longer Error Correction Code (ECC) that is created by reversing the order of the bits of the entire packet and append it just after the original CRC. Thus, each packet $P_a$ is ***augmented*** as a new packet $P'_a$ whose first half is exactly $P_a$ and the second half is its ***clone*** created by reversing the order of the bits of $P_a$. We take the advantage that WSN packets, in practice, are much shorter than their maximum carriable size [18]. For example, IEEE 802.15.4, a widely used standard for WSN, has a maximum allowable packet size of 128 bytes of which 104 bytes is payload. In practice, their payload (data) is very short and of several bytes only [18]. The same is true for WirelessHART which is predominantly being used worldwide for wireless process monitoring and control purposes [8]. Thus, augmenting a packet length is easily affordable within channel capacity in WSN, and hence our RnR decoder design is specifically focused on WSN. However, any network based on digital modulation that can afford such packet augmentation can use our RnR decoder for collision recovery.

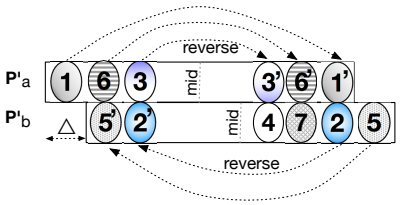



Fig. 4. **RnR**: first decodes chunk 1 and 5. Chunk 1 is reversed and re-encoded as chunk 1' and then subtract from collision which gives chunk 2, and so on.

Fig. 4 illustrates the RnR decoding for $P_a$ and $P_b$ when they collide with arrival time offset $\Delta$. As the figure shows, chunk 1 of $P'_a$ and chunk 5 of $P'_b$ are interference free and hence are decoded using standard decoder. Any chunk in one half of a packet has a ***clone*** in the other half which is its reversed version, and is located at the same position from the other end. For example, chunk 1' is the clone of chunk 1. To recover all chunks of $P_a$ and $P_b$, we iteratively do the following: An already decoded chunk is reversed and re-encoded as its clone, and then subtracted from the collision that overlaps with those symbols, giving a new chunk. That is, chunk 1 is reversed and re-encoded as chunk 1' and then subtracted from the collision that overlaps with those symbols, giving chunk 2. Chunk 2 is reversed and re-encoded as chunk 2' and then subtracted from the collision which gives chunk 3, and so on. We do the same for chunk 5 to get chunk 6, and so on. This re-encoding is done using a standard approach [29] as described in Section III-B1. Once all the chunks of a packet are decoded, they are merged and retrieved as the original packet. We can repeat the same for any number of packets to recover all packets involved in an $m$-packet collision, for any value of $m > 2$. In the general case when $m > 2$, we transform the decoding problem into system of linear equations problem. We consider each chunk in a packet as a variable, and each chunk-level collision yields an equation. We can decode all the packets only if number of equations $\geq$ number of unknown variable. Upon recovering, the link layer sends an acknowledgment (ACK) to all senders.

It can be noted that packet augmentation in our design may increase the chance of collision. Specifically, any collision that happens due to augmented packet length may be due to our design. As Fig. 5 illustrates, if a collision starts at the second half of a packet, there is no need to use our RnR decoder as each packet's non-colliding part contains the original packet or its clone and, hence, is decoded using standard decoder. Thus, the increased collision probability due to packet augmentation does not increase decoding overhead.

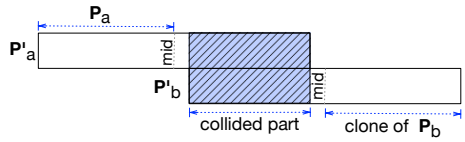



Fig. 5. **A collision that starts at the second half of $P'_a$:** both packets are recoverable from the collision-free parts using a standard decoder.

*1) Re-encoding of Chunks:* Ideally, when transmitting a signal, we expect to receive identical signal. In practice, we always receive slightly different signal from the transmitted one, which complicates the signal processing at the receiver. Frequency offset, $\delta f$, usually exists between two radios due to the difficulty of manufacturing radios with the same center frequency. Due to such offset, the received signal will always be shifted in frequency. Most receivers estimate the frequency offset, $\delta f$, by tracking the phase and then compensate for it. When building a communication system, we must compensate for frequency offset in a received sample as follows.

$$Y[n] = H_a X_a[n] e^{j2\pi n \delta f T} + W[n]$$

In our decoding, we first need to reverse the bits of an already recovered chunk and then re-encode it to be subtracted from the collision to recover a new chunk. This re-encoding is done using a standard approach [29]. Considering $\delta f_a$ as $a$'s frequency shifting, its symbol $X_a[n]$ is received as $Y_a[n] = H_a X_a[n] e^{j2\pi\delta f_a T}$ at the receiver if it is sampled exactly at the same locations as $a$. Taking into account a sampling offset of $\mu_a$ seconds between $a$ and the receiver, the sample at time $n+\mu_a$ can be re-encoded through interpolation based on Nyquist criterion as follows.

$$Y_a[n+\mu_a] = \sum_{i=-\infty}^{\infty} Y_a[i] \text{sinc}(\pi(n+\mu_a-i))$$

which, in practice, is approximated by taking the summation over few symbols near $n$. Standard wireless receivers can estimate the system's parameters such as $H_a$ and $\delta f_a$ using the preamble in $a$'s transmission. Note that preambles are detectable through correlation even during collision.

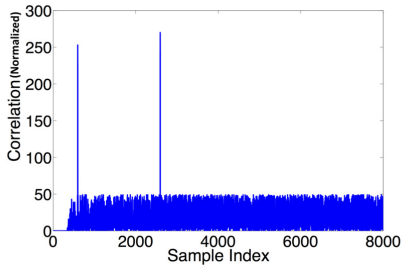



Fig. 6. **Time offset collision detection using correlation method.**

*2) Collision Detection:* Collision Detection is a critical part in the RnR decoding. We adopt the preamble correlation based technique which is a well-known and commonly adopted technique [15], [16]. In a wireless network, every packet is preceded by a known preamble. RnR detects the collision using the correlation between the known preamble and the received signal. Correlation is a widely used technique in signal processing to measure the similarity between two signals [30]. To understand how this technique works, let $L$ be the preamble length (in number of samples). When the receiver receives the first $R$ samples of the packet, it aligns those with $L$ preamble samples and calculates the correlation. It then shifts the alignment to the next $R$ samples and re-calculates the correlation. The receiver repeats this process until the end of the packet.

Fig. 6 shows an example of a 2-packet collision. The preamble is a pseudo-random sequence independent of transmitted data. Hence, the correlation value is always near zero, except when the preamble is perfectly aligned with the beginning of the packet. A value significantly greater than zero indicates a collision. In the figure, the spike at the beginning indicates the beginning of the first packet, while the position of the second spike indicates the beginning of the second packet. After collision detection, to detect the offset ($\Delta$) between the packet arrival times in the same collision, which yields the collision-free chunks, let $t_a$ be the arrival time of $a$'s packet ($P_a$) and $t_b$ be the arrival time of $b$'s packet ($P_b$). To find $\Delta$ between $P_a$ and $P_b$ in the same collision, the receiver finds the distance between the position of the beginning of the first packet and the position of the beginning of the second packet. Mathematically, $\Delta = |t_a - t_b|$. Thus, in Fig. 6, the distance between the two spikes indicates $\Delta$. Beyond two collisions, to detect $m$-packet collision, we detect if there are $m$ spikes in the correlation results. The receiver must compensate for the frequency offset before applying correlation. However, the frequency offset between two communication ends does not change significantly over a long period of time [15]. Hence, the receiver maintains a coarse estimation for each active transmitter at the beginning of the transmission by tracking the frequency offset on the collision-free samples.

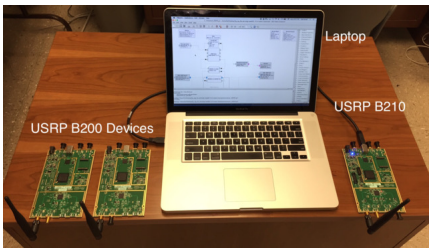



Fig. 7. **USRP connected to PC used in our experiment.**

## IV. Implementation

We implemented RnR in GNU Radio [19] on Universal Software Radio Peripheral (USRP) devices [20]. Each USRP device is connected to a PC running GNU Radio. GNU Radio is a free development toolkit that provides signal processing tools for implementing Software Defined Radios (SDR). USRP devices act as a transmitter/receiver front-end of the SDR platform that help implementing RF applications in a wide range of frequencies. We incorporated RnR in IEEE 802.15.4 GNU Radio implementation [31]. Fig. 7 shows a USRP device connected to a laptop, which was used in our experiment.

On the transmitter (Tx) side, we used B200 USRP device as an RF front-end, which can operate in 70 MHz - 6 GHz coverage range. We used a custom packet generator to generate augmented packets in the IEEE 802.15.4 packet format. A packet is represented using the default GNU Radio vector. RnR can use any standard decoder/encoder as a black-box since it is modulation-independent. Finally, we send the packet to the USRP RF front-end for transmission.

On the receiver side (Rx), a USRP B210 device is used as a base station and acts as the receiver. We include new block to extract the collision-free samples to bootstrap the decoding process. After receiving the signal we apply the correlation method explained in Subsection III-B2 to detect collisions. If there is a collision, we detect how many packets collided together using the correlation values outputted from the correlation block. Next, we extract the collision-free samples and start the iterative RnR decoding process until all collided packets are recovered. Fig. 8 shows the flow chart for the RnR decoding process.

## V. Experiment

To verify the feasibility of RnR, we perform experiments based on IEEE 802.15.4 networks. IEEE 802.15.4 is a widely adopted low power WSN technology. The experiments were performed using 6 USRP devices. Larger-scale evaluation is done in simulations (to be explained later).

### A. Experimental Setup

We incorporated the RnR decoder in the GNU Radio implementation of 802.15.4 [31]. Our experiments are limited to indoor and non-mobile environment, where all the transmitting and receiving nodes are fixed in different locations. Fig. 9 shows the positions of the nodes on the building floor plan where the experiments were conducted. We fixed the distance for each transmitter from the receiver to 25 ft. Five nodes

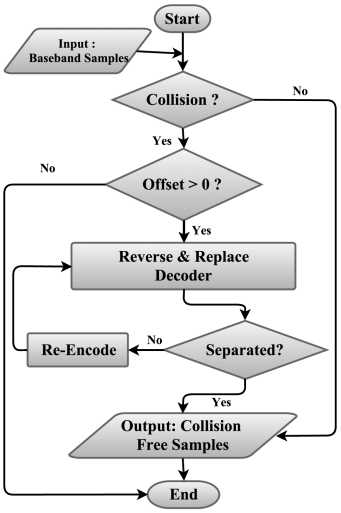



Fig. 8. **Reverse & Replace decoding flow chart.**


Fig. 9. **Node positions on building floor plan.**

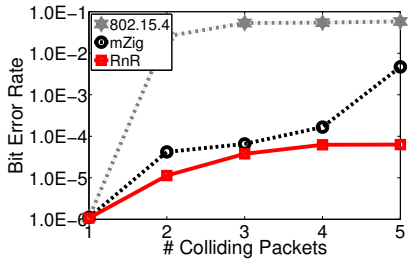
are transmitters and one node is the receiver. Unless stated otherwise, each transmitter sends a 1032-bit packet (32 bits (preamble+header) and 1000 bits payload) every 4ms. We experimented in the 2.4GHz band. To avoid interference from the existing WiFi networks, we choose channel 26 as the operating channel for 802.15.4, which is non-overlapping with WiFi. The bandwidth for this channel is 2MHz. To meet the IEEE 802.15.4 standard transmission and reception power requirements (maximum 0dBm), we set Tx gain to 70 dB in GNU Radio, which allows the USRP devices to perform transmission/reception near 0dBm. We fixed the antenna height approximately 5 ft above the ground.

### B. Evaluation Criteria and Baselines

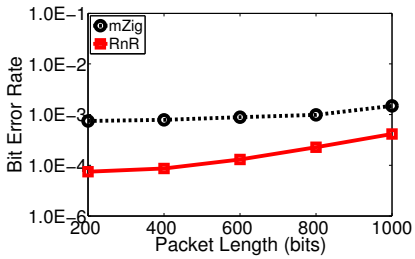
We compare the performance of RnR against that of mZig, which is the state-of-the-art collision decoder for ZigBee networks. mZig exploits the physical layer of ZigBee to resolve collision. We also compare against ZigZag, which is the state-of-the-art collision recovery mechanism for IEEE 802.11 networks, by adapting it to 802.15.4 networks. Finally, we compare RnR against the conventional IEEE 802.15.4 MAC protocol which handles collisions using CSMA/CA mechanism. For networks where few collision events occur, conventional IEEE 802.15.4 decoders would be sufficient.

We use the following metrics for performance evaluation.

- ***Bit Error Rate (BER)*** defined as the ratio of the number of incorrect bits to the total number of bits in the packet.




(a) **Under varying number of packets colliding**




(b) **Under varying packet sizes**

Fig. 10. **Bit Error Rate under different decoding**

We drop a packet if the BER $\geq 10^{-3}$. This setting complies with traditional wireless design [15].

- ***Correctly Decoding Rate (CDR)*** defined as the ratio of the number of packets that are correctly decoded at the receiver to the total number of transmitted packets. This is a key metric to evaluate RnR's performance.
- ***Throughput*** defined as the total number of bits received per unit time.
- ***Energy Consumption***.
- ***Latency***.

### C. Results

*1)* ***BER:*** First, we analyze the BER for different numbers of colliding packets. The node positions are shown in Fig. 9. We run the experiment for one hour. We vary the number of collided packets between 2 and 5 by turning off and on the Tx's. We compare the performance against mZig and the conventional 802.15.4 (we particularly considered ZigBee [2]). In this experiment, CSMA/CA is disabled for 802.15.4 to show the decoding capability from collisions. Fig. 10(a) shows that 802.15.4 has BER greater than $10^{-3}$ when 2 or more packets collide. Hence, it cannot decode packets from collisions. RnR has BER less than $10^{-3}$ when two or more packets collide. Analytical results from mZig [16] show that its BER exceeds $10^{-3}$ when more than 4 packets collide. RnR has much lower BER than mZig, and significantly lower than 802.15.4 as the number of colliding packets increases.

In addition to our chosen 1000-bit payload length, we vary the packet length and observe the BER. In this experiment, we vary the payload length from 200 to 1000 bits, and the number of colliding packets is limited to 2. Fig. 10(b) shows a negligible increase in BER under RnR as we increase the packet length. In contrast, the BER under mZig significantly increases as the packet length increases. This happens because mZig decodes by discerning amplitude levels of the colliding

packets (as we discussed in Section II). It has serious limitations in decoding as it experiences a BER greater than $10^{-3}$ when the payload length approaches 1000 bits. Our results (Fig. 10) as well as that in [16] show that mZig can decode packets with a maximum length of $\approx$ 1032 bits, with up to 4 concurrent transmissions, while RnR shows the capability of decoding any number of concurrent transmissions.

*2)* ***CDR:*** We measure the CDR, we use 2 transmitters. Every transmitter attempts to send a packet in a random time between 50-100ms of frame gap for 5 hours. Once the receiver receives the packets, we record the data. We recorded reception of nearly 100,000 packets and analyzed the CDR offline. In this experiment, we disabled CSMA/CA for 802.15.4 to show its decoding capability from collisions. Fig. 11 plots the Cumulative Distribution Function (CDF) of the CDR values of RnR and that of 802.15.4. It shows that RnR can correctly decode packets from almost all collisions. RnR has a CDR of nearly 97.5% in more than 95% cases of collisions. On the other hand, conventional 802.15.4 has CDR less than 0.05% for 85% cases of collisions. This is because 802.15.4 physical layer is inherently not capable of recovering packets from collision. The results demonstrate RnR as a highly effective approach for collision recovery.

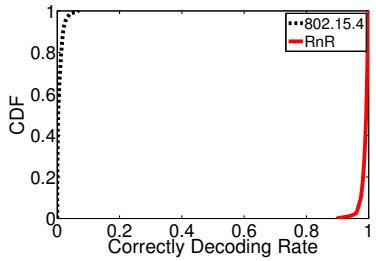



Fig. 11. **Distribution of CDR in RnR.**

*3)* ***Throughput:*** We now observe throughput under varying number of transmitters. In each 4ms time window, each transmitter sends one packet at some random time in the window (maintaining exactly one transmission per window). Fig. 12 compares the throughput of RnR against that of the default 802.15.4, mZig, and ZigZag. While all schemes achieve almost the same throughput under one transmitter (no collision), RnR outperforms others as the number ($m$) of concurrent transmissions increases. Although the throughput under mZig looks competitive against that under RnR as long as $m \leq 4$, it sharply decreases when $m > 4$ as mZig cannot decode packets in the latter case. It decodes some of those packets later if 4 or less of these packets re-collide. For larger values of $m$, we will show later in simulation that RnR significantly outperforms mZig. RnR also outperform ZigZag. Since ZigZag needs $m$ repetitions to decode packets from $m$-collisions, its throughput never exceeds that achieved with a single transmitter. Also, it requires to explore all possible (exponential number) combinations of the chunks to recover a packet as we cannot identify the packets across multiple collisions before they are recovered. On the other hand, RnR requires a single collision to decode all $m$ packets. Hence, RnR's throughput increases gradually with the increase in $m$, and remains nearly $m$ times that of ZigZag.

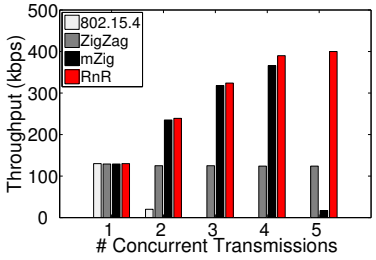



Fig. 12. **Throughput comparisons among RnR, mZig, ZigZag, and 802.15.4.**

| Device mode | Current Consumption (Supply voltage 3 v) |
|---|---|
| Tx | 17.5 mA |
| Rx | 18.8 mA |
| Idle | 0.5 mA |
| Sleep | 0.1 $\mu$A |

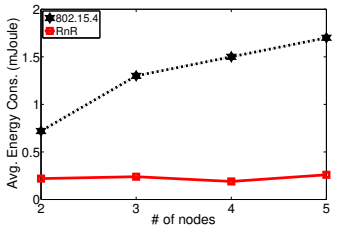
TABLE I
CURRENT CONSUMPTION IN CC2420

*4)* ***Energy Consumption and Latency:*** We measure the energy consumption in RnR and compare with that in the conventional 802.15.4. Here we enable CSMA/CA of 802.15.4 to measure the energy efficiency of the protocol. In 802.15.4 each transmitter uses CSMA/CA to sense the channel before transmitting. If the channel is busy, it performs a random back-off between 0.32ms and 4.8ms, and then attempts to retransmit. In this experiment, we collect packets from 5 transmitters all of which try to send to one receiver. Every transmitter is 25 ft apart from the receiver and sends a 1032-bit packet every 4ms. We calculate energy consumption to collect 100 packets from each node. We model the energy consumption based on CC2420 radio [25] which is based on IEEE 802.15.4 in 2.4 GHz. Table I show the energy model for CC2420. We assume the receiver is always connected to a power source, and do not consider its energy consumption. Fig. 13(a) shows the average energy in each node per packet. RnR has almost a fixed energy consumption of 0.013mJ. On the other hand, 802.15.4 consumes an average of 0.7mJ considering 2 transmitters. Its average energy consumption increases linearly with the number of transmitters due to retransmissions and the 802.15.4 MAC overhead (back-off, ACKs). This shows that RnR is highly energy efficient.

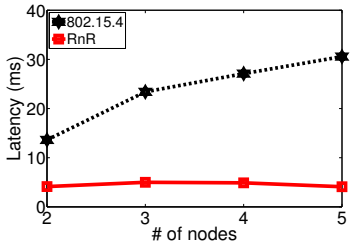
Fig. 13(b) demonstrates the latency improvement for using RnR under the same previous setting. We measured the latency to deliver each packet at the receiver under varying numbers of concurrent transmissions. RnR takes approximately 4ms on average to deliver a packet, while the 802.15.4 takes approximately 13ms for 2 transmitters. As expected, with the increase in the number of concurrent transmissions, the latency in 802.15.4 increases due to its MAC overhead while, in RnR, it remains almost constant.

## VI. SIMULATION

To evaluate RnR in larger scale, we have performed simulations using the GNU Radio simulation environment [19]. GNU




(a) **Average Energy Consumption**




(b) **Latency**

Fig. 13. **Energy consumption and latency under varying # of nodes.**

Radio includes a simulation environment that provides signal processing blocks to simulate signal processing systems on the host. For evaluation in simulations, we consider the same metrics and baselines used in our testbed experiment.

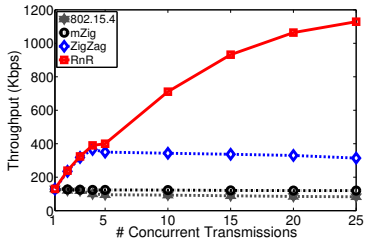



Fig. 14. **Throughput comparison in simuation.**

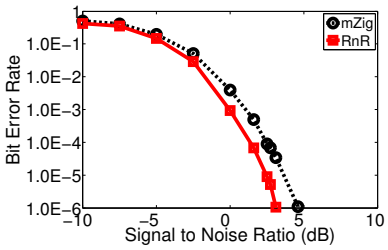



Fig. 15. **Bit error rate under different SNR conditions in simulation.**

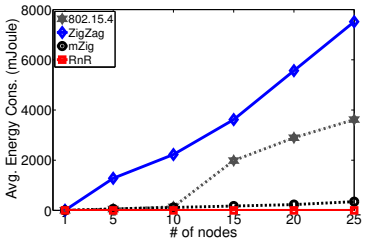
### A. Simulation Setup

In simulations, we consider the number transmitters up to 25 all of which try to send to one receiver. Once the receiver receives 100 packets for each value of $m$ between 2 and 25, we record the data. Every transmitter attempts to send a packet in a sliding window of 4ms, then remains idle for 100 ms. Every packet has a fixed length of 1032 bits. We record the data for 10 hours a day for 5 days, and analyze the results offline. During the simulation, every transmitter has a fixed gain of 70 dBi. A 3 dBm of Additive white Gaussian noise (AWGN) noise was added to the channel.

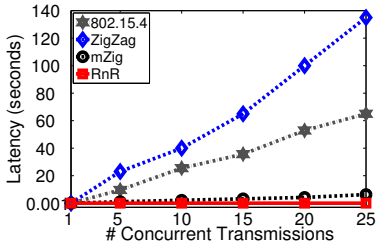
### B. Simulation Results

*1)* ***Throughput:*** We compare the RnR throughput against the conventional 802.15.4, mZig, and ZigZag. In this simulation, the MACs were enabled for both 802.15.4 and mZig. As Fig. 14 shows, when $m > 2$, RnR's throughput increases linearly while 802.15.4 maintains the same throughput of approximately 120 kbps with a small decrease as $m$ increases. As ZigZag requires $m$ retransmissions to recover $m$ collided packets, it maintains a constant throughput of around 121 kbps with the increase of $m$. RnR achieves much higher throughput compared to mZig, and continues to outperform mZig as $m$ increases. When $m > 4$, the throughput of mZig starts decreasing due to its decoding limitation. When $m = 25$, RnR has an average throughput of 1.2 mbps which is approximately 4x higher than mZig's average throughput.

*2)* ***BER:*** Fig. 15 shows the BER comparison among different schemes under varying SNR conditions when $m = 2$. The conventional 802.15.4 is not considered in this simulation since its BER exceeds the threshold for even 2-packet collision. The BER in RnR is much lower than that in mZig. Since mZig uses amplitude estimation to determine certain bits, the impact of SNR is higher in mZig. However, both schemes are still within the range of reference (3dB) line for $m = 2$.




(a) **Average energy consumption per node**




(b) **Latency**

Fig. 16. **Energy consumption and latency in simulation.**

*3)* ***Energy Consumption and Latency:*** To measure the energy consumption and latency, we use a sliding window of 1 second. Fig 16(a) shows the average energy consumption per packet at each node for all four schemes. For all the values of $m$, RnR maintains an average energy of 0.2 mJoules per packet. For 5 nodes, mZig consumes an average of 60 mJoules per packet to deliver 100 packets. For 25 nodes, it consumes approximately 349 mJoules. For mZig, the energy consumption increases due to the MAC protocol overhead as it allows only 4 concurrent transmissions. ZigZag consumes even higher energy. For 5 nodes it consumes approximately 1280 mJoules, and the average energy consumption increases

sharply with the increase in $m$. This is because ZigZag depends on $m$ retransmission to resolve an $m$-packet collision. Thus, an exponential time is needed in calculating different chunk combinations to resolve the collision. For 802.14.5, the average energy consumed for 5 nodes is 53 mJoules. Also in 802.15.4, the increase in $m$ leads to an increase in the average energy consumed due to the MAC overhead (back-off, ACKs, and retransmissions).

Fig. 16(b) shows the per packet latency for delivering 100 packets for each value of $m$. RnR takes approximately 5ms on average for the different values of $m$. For 5 nodes, mZig takes 1 second on average to deliver the packets, and for 25 nodes it takes approximately 6 seconds. In mZig, the BER constraints the number of concurrent transmissions, and mZig experimental result shows that up to 4 concurrent transmission must be decoded before transmitting again. Hence, mZig suffers from an increased latency. ZigZag on the other hand takes around 23 seconds to deliver the packets for 5 nodes. Its latency increases with the increase in the value of $m$ due to the exponential waiting time and the retransmission. While the latency in RnR is 8 ms for 25 concurrent packets, traditional 802.15.4 takes around 65 seconds. This time increases with the increase of $m$ in 802.15.4 because it suffers from collisions, random back-off waiting time, and retransmissions.

## VII. Conclusions

In this paper, we have presented *Reverse and Replace decoding (RnR)*, a novel interference cancellation method for wireless sensor network (WSN). Because interference poses a serious problem in wireless network, the mechanisms for recovering from collisions have been studied in many early works [11], [12], [13], [14], [15], [21]. However, these existing solutions for collision recovery make simplified assumptions such as the availability of one of the collided packets, repeated collisions of the same packets, and the ability to identify the collided packets before recovering them which do not hold for WSNs and most wireless networks. RnR does not rely on the assumptions made in the existing works, and can recover all packets from a single collision. It entails a physical-link layer design that exploits the raw signal samples from a collision to recover the collided packets. RnR is bootstrapped only when a collision is detected, and involves no decoding overhead in the absence of collision. We have also implemented RnR in GNU Radio for USRP devices considering IEEE 802.15.4 based networks. Our experiments using 6 USRP devices and also simulation results using GNU Radio simulator demonstrate that RnR can successfully decode packet in 95% cases of collisions, and improves the correctly packet decoding rate of up to 97.5% compared to standard decoders in case of collisions, providing 4x higher throughput compared to the state-of-the-art collision recovery mechanisms. The results demonstrate RnR as a practical choice for collision recovery in WSNs.

## Acknowledgments

This work was supported by NSF through grants CRII-1565751 (NeTS), and was partially supported by DOE grant P200A120110 and NIST grant award 60NANB12D156.

## References

[1] IEEE 802.11 standards. [Online]. Available: http://standards.ieee.org/about/get/802/802.11.html
[2] IEEE 802.15 standards. [Online]. Available: http://standards.ieee.org/about/get/802/802.15.html
[3] G. Bianchi, L. Fratta, and M. Oliveri, "Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs," in *PIMRC*, 1996.
[4] M. Garetto, T. Salonidis, and E. W. Knightly, "Modeling per-flow throughput and capturing starvation in csma multi-hop wireless networks," *IEEE/ACM TON*, vol. 16, pp. 864–877, 2008.
[5] S. Sen, R. R. Choudhury, and S. Nelakuditi, "CSMA/CN: carrier sense multiple access with collision notification," *IEEE/ACM TON*, vol. 20, pp. 544–556, 2012.
[6] S. Khurana, A. Kahol, and A. P. Jayasumana, "Effect of hidden terminals on the performance of IEEE 802.11 MAC protocol," in *LCN*, 1998.
[7] K. Xu, M. Gerla, and S. Bae, "Effectiveness of RTS/CTS handshake in IEEE 802.11 based ad hoc networks," *Ad Hoc Networks*, vol. 1, pp. 107–123, 2003.
[8] WirelessHart. [Online]. Available: http://www.hartcomm2.org
[9] A. Saifullah, S. Sankar, J. Liu, C. Lu, R. Chandra, and B. Priyantha, "Capnet: A real-time wireless management network for data center power capping," in *RTSS*, 2014.
[10] J. Lu and K. Whitehouse, "Exploiting the capture effect for low-latency flooding in wireless sensor networks," in *SenSys*, 2008.
[11] S. Gollakota, S. D. Perli, and D. Katabi, "Interference alignment and cancellation," in *SIGCOMM*, 2009.
[12] M. Jain, J. I. Choi, T. Kim, D. Bharadia, S. Seth, K. Srinivasan, P. Levis, S. Katti, and P. Sinha, "Practical, real-time, full duplex wireless," in *MobiCom*, 2011.
[13] S. Katti, S. Gollakota, and D. Katabi, "Embracing wireless interference: Analog network coding," in *SIGCOMM*, 2007.
[14] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," in *SIGCOMM*, 2006.
[15] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," in *SIGCOMM*, 2008.
[16] L. Kong and X. Liu, "mZig: Enabling multi-packet reception in ZigBee," in *MobiCom*, 2015.
[17] Zigbee alliance. [Online]. Available: http://www.zigbee.org
[18] E. H. Callaway Jr, *Wireless sensor networks: architectures and protocols*. CRC press, 2003.
[19] GNURadio. [Online]. Available: http://gnuradio.org
[20] USRP B200-B210 data sheet. [Online]. Available: http://www.ettus.com/content/files/b200-b210_spec_sheet.pdf
[21] D. Halperin, T. Anderson, and D. Wetherall, "Taking the sting out of carrier sense: interference cancellation for wireless LANs," in *MobiCom*, 2008.
[22] P. Patel and J. Holtzman, "Analysis of a simple successive interference cancellation scheme in a DS/CDMA system," *Selected Areas in Communications, IEEE Journal on*, vol. 12, pp. 796–807, 1994.
[23] S. Sen, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi, "Successive interference cancellation: Carving out mac layer opportunities," *IEEE Trans. Mobile Computing*, vol. 12, pp. 346–357, 2013.
[24] A. Saifullah, M. Rahman, D. Ismail, C. Lu, R. Chandra, and J. Liu, "Snow: Sensor network over white spaces," in *SenSys*, 2016.
[25] Texas instruments. [Online]. Available: http://www.ti.com/product/cc2420
[26] http://www.atmel.com/images/doc8111.pdf.
[27] J. H. Kim and J. K. Lee, "Capture effects of wireless CSMA/CA protocols in rayleigh and shadow fading channels," *IEEE Trans. Vehicular Technology*, vol. 48, no. 4, pp. 1277–1286, 1999.
[28] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler, "Exploiting the capture effect for collision detection and recovery," in *EmNets-II*, 2005.
[29] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.
[30] P. Castoldi, *Multiuser detection in CDMA mobile terminals*. Artech House, 2002.
[31] GNU Radio IEEE 802.15.4 implementation. [Online]. Available: https://github.com/bastibl/gr-ieee802-15-4