Moving Away from Collision Avoidance: Towards Collision Detection in Wireless Networks

Souvik Sen
Duke University

Naveen Santhapuri
Duke University

Romit Roy Choudhury
Duke University

Srihari Nelakuditi
University of South Carolina

ABSTRACT

Wireless networks are founded on the principles of collision avoidance. This paper makes an attempt to detect and abort collisions in wireless networks. Briefly, the receiver uses physical layer information to detect collisions, and immediately notifies the transmitter to abort transmission. The collision notification consists of a unique signature, sent on the same frequency channel as the data. The transmitter uses a second listener antenna to discern this notification through *signature correlation*. The transmitter aborts, freeing the channel for other productive transmissions. We call this Carrier Sense Multiple Access with Collision Notification (CSMA/CN). Early results from a small USRP/GNURadio testbed confirm the feasibility and performance gains with CSMA/CN.

1. INTRODUCTION

Wired networks implement Carrier Sense Multiple Access with Collision Detection (CSMA/CD). The transmitter senses the wired channel while transmitting, and upon detecting a collision, aborts its own transmission immediately. CSMA/CD is feasible on wired links because signal attenuation is negligible, hence, a collision detected at the transmitter implies a collision at the receiver. CSMA/CD is beneficial because once a collision is detected, the remainder of the packet is not transmitted unnecessarily. Instead, the channel is released for other transmissions.

Wireless networks are unable to implement CSMA/CD because channel conditions can be considerably different at the transmitter and the receiver. Therefore, MAC protocols are founded on Collision Avoidance (CSMA/CA), an approach that aims to separate concurrent transmissions in space and time. Packets still get corrupted due to collision or channel fading. The transmitter remains unaware of the corruption and continues to transmit the entire packet unnecessarily. Eventually, based on the absence of an acknowledgment from the receiver, the transmitter infers packet loss, and prepares for retransmission. Channel utilization degrades, leading to poor system performance with CSMA/CA. This paper proposes Carrier Sense Multiple Access with Collision Notification (CSMA/CN) – an attempt to obtain part of CSMA/CD's benefits in wireless environments.

CSMA/CN takes a cross-layer approach between PHY and MAC (Figure 1). The transmitter is equipped with two wireless interfaces tuned to the same channel, one for transmission and another for "listening". The receiver has a single interface. Once communication begins, the receiver exploits preamble correlation to detect the presence of an interference. Realizing that the packet is likely to fail, the receiver checks the confidence of incoming bits via SoftPHY [1, 2]. When the receiver is reasonably confident of an error, it initiates a *collision notification* to the transmitter. The notification is a short signature unique to the receiver, also known to the transmitter. The transmitter's listening antenna continuously "searches" for this signature using *signature correlation*. We show that even in the presence of a strong signal from the transmit antenna, signature correlation at the listening antenna can reliably discern the notification. The transmitter aborts, releasing the channel for nearby transmitters.

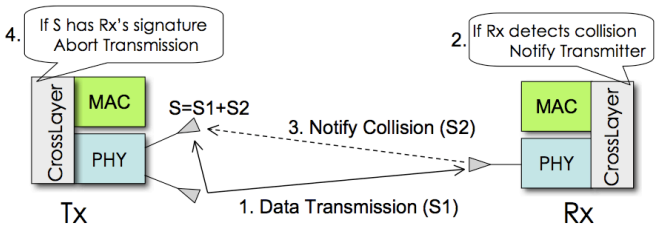

Figure 1: Basic structure of CSMA/CN.

CSMA/CN may not be necessary when a packet experiences a few bit errors from fading. Partial Packet Recovery (PPR) [1] guesses which parts of a packet are in error, and requests the transmitter to retransmit only these erroneous parts. With few bit errors, PPR offers good gains because it avoids an entire packet retransmission. However, when a packet undergoes a collision, many bit errors are likely, and retransmitting all of them can be wasteful. Collision Notification aborts transmission of a colliding packet. *The intuition is that aborting (or prevention) is better than recovery (or cure).* The remainder of the packet is resumed later after appropriate (backoff/rate/carrier-sensing) adjustments.

With CSMA/CN, one may argue that the additional interface at the transmitter may be better utilized for a parallel com-

munication, perhaps on a different channel. We clarify that the "listener" may not be viewed as an additional interface, only a simple correlation logic with an antenna. Decoding capabilities are not necessary, hence, this logic can be part of the same interface. Besides, collision notification to the listener is in-band, requiring no additional bandwidth.

This paper makes three contributions:
(1) *Identifying a middle ground between CSMA/CD and CA.* CSMA/CN is an early attempt to rethink medium access control in wireless networks. This paper explores the first steps in this direction, demonstrating that further developments are feasible and worth pursuing.

(2) *Develop the Collision Notification architecture with practical constraints in mind.* Our designs are validated with supporting measurements on the USRP/GNURadio platform. We show the feasibility of detecting collisions at the receiver, as well as reliable identification of the collision notification at the transmitter's listening antenna.

(3) *Implement and evaluate CSMA/CN on a prototype of seven USRP nodes.* Experimental results show consistent throughput improvements over 802.11 and PPR. We identify several avenues of further research.

2. ARCHITECTURE AND DESIGN

We believe that any attempt to abort collisions in wireless links will need to conform to the following functional requirements. (1) A wireless transmitter T cannot detect the collision while transmitting; the receiver R must get involved. (2) Receiver R will need to detect collision and convey it back before the packet is fully transmitted. (3) T needs at least an additional antenna for listening while transmitting. This section proposes CSMA/CN as a practical system that conforms to these requirements. Design decisions are discussed next followed by feasibility tests. Later, Section 3 presents the overall testbed results.

(1) Design: Transmission and Collision Detection
In CSMA/CN, the transmitter T uses one interface for transmitting and the other (correlator) for listening. The receiver R uses its single interface for multiplexing between transmission and reception. Transmission is initiated as in IEEE 802.11, except one difference: for every packet, the PHY layer preamble is concatenated with an additional bit sequence, a *signature*, uniquely computed from the intended receiver's identifier (Figure 2(a)). The transmitter T ensures the channel is idle and transmits this packet using the transmit antenna. The listening antenna, by virtue of being very close to the transmitting antenna, receives this signal with a high signal strength – we call this the *self-signal*. The packet's intended receiver R also receives the transmitted signal and starts decoding the arriving bits. Simultaneously, R triggers collision detection.

Collision happens when a nearby transmitter $T1$, interferes

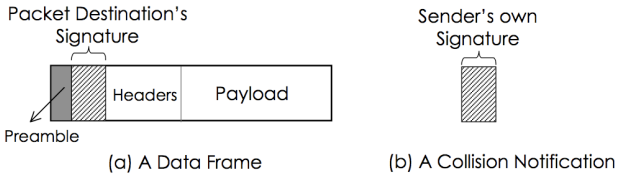

Figure 2: CSMA/CN frame formats.

with R's reception, causing packet corruption (Figure 3). To detect such collisions, receiver R "searches" for a PHY layer preamble in its incoming signal. Searching occurs through correlation of the preamble with the signal arriving at R's antenna. This happens in parallel, and does not affect the normal packet decoding procedure. Once $T1$'s preamble impinges on R's antenna, the correlation exhibits a spike, raising an alert that the packet may be in "trouble". Of course, arrival of a new preamble may not necessarily cause a collision; the reception may sometimes succeed even in presence of the interference. To verify the impact of interference, R consults SoftPHY [1] to obtain confidence values of the bits arriving from T. The confidence value is an indicator of how likely a bit is in error. Based on a window of confidence observations, R is able to infer whether the packet is expected to get corrupted. If so, R halts reception, and prepares to send a collision notification to transmitter T.

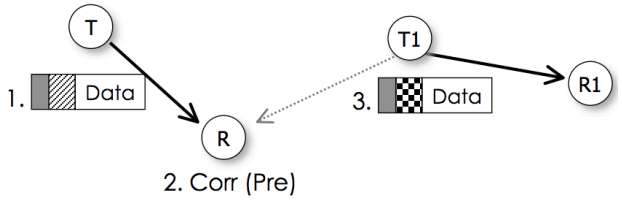

Figure 3: R correlates to preamble, denoted Corr(Pre).

If the interferer $T1$ starts first, and the transmission from T starts later, R may need to abort T (Figure 4). However, R must first ensure that the later-arriving signal is actually meant for itself. Preamble correlation is not sufficient because T may use the same preamble for transmitting to some other receiver; R should not send an abort then. Because of this, R "searches" for its own signature in the signal. If T intends its transmission to R, it would embed R's signature in the packet. R will detect this through signature correlation, and send out an abort. *To summarize, R searches for a preamble while receiving its signal of interest, but searches for its own signature while receiving an interference.*

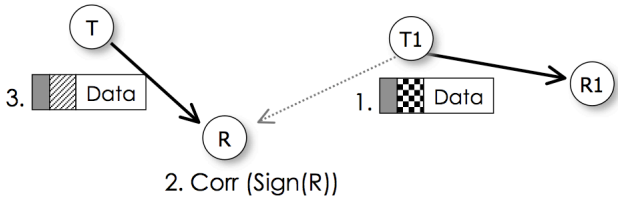

Figure 4: Under interference, R correlates its own signature, denoted as Corr(Sign(R)).

(2) Design: Collision Notification and Abort

The Collision Notification (CN) is composed of only the receiver R's own signature. This is the same bit sequence that T included in its packet to R (Figure 2(b)). The receiver transmits the CN packet like a regular 802.11 ACK – there is no carrier sensing, hence, the CN is transmitted even though the transmitter is still transmitting. The listening interface of the transmitter continuously correlates for the receiver's signature in the incoming signal (Figure 5). This correlation is more challenging because the self-signal is much stronger than the notification. We show that even then the listener can discern the notification with consistent accuracy.

Upon discerning the notification, the listener immediately alerts the transmitting interface, which then suspends transmission (other transmitters like $T1$ do not abort because they are not correlating with R's signature). The correctly-aborted transmitter backs off as prescribed in 802.11. Other backlogged nodes in the vicinity takes up this opportunity to transmit; if no other node transmits, the same transmitter may resume transmission.

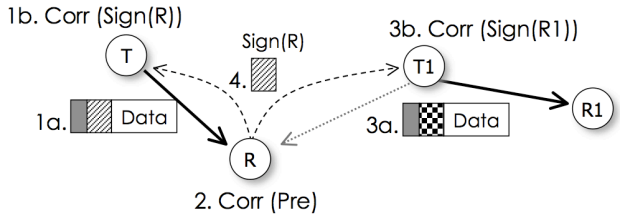

Figure 5: T aborts due to signature correlation. $T1$ continues because it is searching for $R1$'s signature.

A pertinent question is *whether the collision notification will interfere with nearby active transmissions.* This will certainly be true when the interferer's receiver is close to the notification sender. Nevertheless, the small size of the notification permits various possibilities for efficient recovery. When it interferes, the short window of bit errors can be repaired by a scheme like PPR, as if its a small burst of fading loss. Such a scheme employed to cope with fading can handle errors due to notification as well. Alternatively, the packet may be augmented with just enough error correcting codes to recover from the notification-sized errors [3]. Finally, observe that 802.11 ACKs can also induce errors at a nearby receiver, much like CSMA/CN's collision notifications. They only differ in the kind of topologies they impact.

We believe CSMA/CN is a simple but novel approach to wireless medium access control. The following pseudo code captures the core flow of operations.

Algorithm 1 : T.transmit(R, Data)

```
1: Begin transmitting frame <Preamble:Sign(R):Data>
2: Keep listening and correlating with Sign(R)
3: if Corr(Sign(R)) high then
4:     Abort transmission
```

Algorithm 2 : R.receive()

```
1: if Frame of interest is already being received then
2:     if Corr(Preamble) high and many bits suspect then
3:         Transmit <Sign(R)>
4: if Interfering frame is being received then
5:     if Corr(Sign(R)) high then
6:         Transmit <Sign(R)>
```

(3) Supporting Measurements: Signature Correlation

We emphasize that signature correlation does not require decoding the signature bits – it suffices to only detect the signature's presence in the channel. Systematic experiments under varying network conditions show that *signature correlation* can be robust. Figure 6(a) shows the noticeable jumps in correlation when a known signature arrived amid continuous background transmissions.

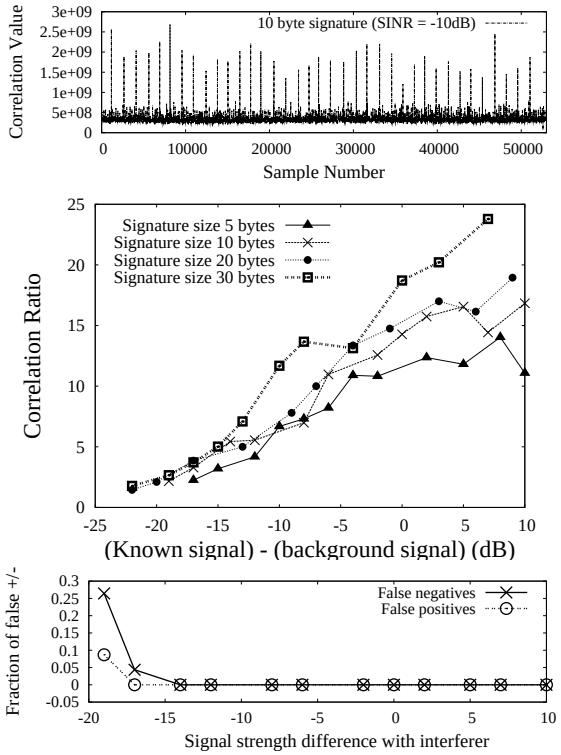

Figure 6: (a) The correlation value spikes every time a known signature arrives amid a background transmission. (b) Correlation Ratio with varying transmitter-receiver separation and signature size. (c) False positive and false negative rates of discerning collision notification against varying transmitter-receiver separation.

To quantify the jumps, we define *Correlation Ratio* as the correlation observed once the signature arrives, divided by the average correlation before its arrival. Figure 6(b) shows the variation of Correlation Ratio with increasing RSSI difference between the signature and background transmissions. Put differently, in the lower end of the x-axis, the signature is much weaker than the background signal, as

can be expected when the collision notification arrives in presence of the self-signal. Evidently, performance is satisfactory so long as the signature is no weaker than 18dB of the self-signal. Figure 6(c) shows the false positive and false negative rates of signature correlation. Results are certainly promising. In our section on ongoing work, we will discuss methods of improved correlation for weaker links.

We also studied *whether different notification signatures may induce similar correlations.* Figure 5 shows a scenario where R's signature arrives when $T1$ is searching for $R1$'s signature. To ensure that $T1$ does not abort, CSMA/CN needs to ensure that the two signatures do not exhibit a high correlation (no false positives). We investigate this by cross-correlating randomly chosen 10-byte signatures. Table 1 shows high differences between auto-correlation (table diagonal) and cross-correlation. The values in the braces denote the minimum hamming distance between signatures i and j. Although promising, we plan to better understand how this scales to more signatures.

Table 1: Correlation with different signatures (minimum Hamming Distance shown in parenthesis)

Sign.	1	2	3	4	5
1	**5.8 (0)**	1.2 (37)	0.9 (42)	1.1 (34)	1.3 (37)
2	1.1 (37)	**4.1 (0)**	1.1 (39)	1.7 (32)	1.1 (39)
3	1.3 (35)	1.3 (34)	**5.1 (0)**	1.3 (34)	1.4 (34)
4	1.2 (34)	1.6 (32)	0.9 (42)	**7.3 (0)**	1.2 (37)
5	1.1 (37)	1.2 (39)	.9 (39)	1.1 (37)	**5.0 (0)**

3. IMPLEMENTATION AND EVALUATION

We have implemented CSMA/CN on a USRP testbed of 7 nodes. We used the GNURadio framework with Zigbee CC2420 as the physical layer. CC2420 operates at 2.4 GHz with a symbol rate of up to 2M symbols/s, translating to a maximum of 250 Kbps. The Zigbee nodes perform carrier sense and backoff. The SoftPHY, signature correlation, and collision detection logic were ported to Zigbee. The Zigbee implementation uses a hard decision decoder. On the transmitter side, every 4 data bits (nibble) is mapped to a 32 bit codeword. Hence there are 16 valid codewords. For every 32 bits received from the demodulator output, the receiver maps it to the closest valid codeword. The Hamming distance between the received codeword and the closest valid codeword provides SoftPHY's confidence of the nibble.

We implement *signature correlation* using a matched filter. We use a threshold on the *Correlation Ratio* – when the signature correlation is above the ratio, the node identifies the arrival of a new transmission. The threshold is chosen based on the RSSI difference between the self-signal and the receiver's signal (details omitted in the interest of space). Signature sizes are chosen to be 5 bytes.

3.1 Performance Evaluation

Our experiments are designed to answer 3 performance questions: What is the (1) accuracy and delay of detecting a collision at the receiver? (2) turn around time from collision detection to transmission abortion? (3) CSMA/CN's throughput gain over PPR and 802.11?

To evaluate the accuracy and delay of collision detection at the receiver, we set up a transmitter-receiver (T-R) pair, and a moving interferer with backlogged traffic. All packets are 1500 bytes. The T-R link delivers almost 100% of the packets in absence of the interferer. In presence of interference, the packet is either received correctly, received with errors, or not received at all (preamble corrupted). Figure 7(a) shows the break-up of these three cases with increasing SINR on the X-axis. CSMA/CN's collision detection accuracy curve is also shown. Evidently, when the SINR is low, CSMA/CN can accurately identify a collision at the receiver through preamble-correlation and SoftPHY hints (recall scenario in Figure 3). The accuracy decreases when the interference becomes much weaker than the signal – in such cases, SoftPHY is sometimes unable to recognize the collision (false negatives). However, due to the weak interference, the packet is correctly decoded 93% of times, and hence, the impact of false negatives is marginal. We also observed negligible false positives (1%) in the experiments.

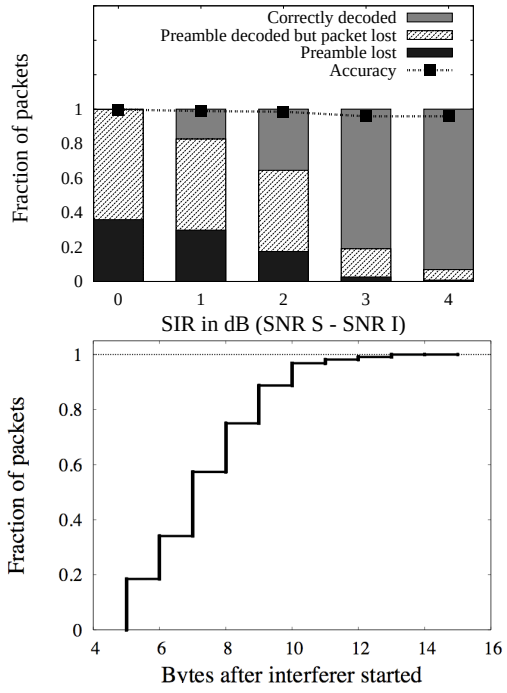

Figure 7: (a) Collision detection accuracy with increasing interference strength (b) CDF of bytes when the collision was detected after it started

CSMA/CN initiates the abort notification only after an interfering preamble is detected and the SoftPHY hints have confirmed the collision. This operation introduces a delay. Figure 7(b) shows the CDF of this delay (expressed in number of bytes). Evidently, all the collisions were detected within 16 bytes from the time of preamble detection. The

median is around 7 bytes, a fairly small value in comparison to a typical packet size. We conclude that CSMA/CN can detect collision reasonably quickly and accurately.

The next experiment compares CSMA/CN's throughput against PPR [1] and the conventional scheme (we call it *802.11-like*). We set up 3 USRPs, two collocated as the transmitter and one at the receiver, to form the CSMA/CN link. Two other links are set up around the CSMA/CN link – one acts as a hidden terminal link and the other an exposed terminal link. These secondary links comprise of 2 USRPs each, hence, do not perform collision notification themselves. Instead, they act as interferers to the CSMA/CN link, and contend and transmit on the "free" channel when the CSMA/CN transmission is aborted. In other words, their throughput reflects the gains from CSMA/CN. Figure 8 reports the aggregate throughput over all three 3 links, against increasing failure rate at the CSMA/CN link. The failure rate is increased by moving the hidden/exposed links closer to the CSMA/CN link. The results are averaged across 10 different topologies, and normalized over the *802.11-like* scheme. As one may expect, CSMA/CN's gain increases with higher packet failures. In this regime, large portions of the packet are unnecessarily transmitted (and later retransmitted) with PPR. Aborting this redundant transmission upfront offers benefits.

The turn around time (i.e., the time from the first bit error to the time when transmission is halted) is of interest. A shorter turn around time (TAT) implies potential for larger gains. Figure 8(b) shows the CDF of TAT in the units of USRP byte-transmission time (we do not report absolute time because USRP units incur artificial delays at the transmit and receive end, hence, the absolute time is misleading). Evident from the figure, the median turn around time is around 150 bytes. This is somewhat large due to the artificial delay in obtaining samples from the USRP front-end to the user space. We are exploring the possibilities of implementation in an FPGA/hardware.

Trace based evaluation

We use trace based evaluation to understand CSMA/CN's performance in a multi-link topology. The traces were collected from 5 different topologies, each with 5 USRP transmitters scattered around 2 USRP receivers. Each receiver was positioned to be in the range of 2 to 4 senders. The links were made reasonably strong, hence, fading losses were negligible. However, with multiple senders contending for the channel, hidden and exposed terminals were naturally present. Two types of traffic patterns were used – one continuous and another bursty, with a constant packet size of 1000 bytes. We emulated the three protocols offline (CSMA/CN, PPR, and 802.11-like) on these collected traces. Briefly, when a collision is observed at a link (based on the CSMA/CN criteria), our emulator stops that link, and starts the next transmission earlier. Carrier sensing, backoff, and SIFS time slots are carefully accounted for between transmissions. This emulates (although with some approximation) what would have happened if CSMA/CN was running on the same network.

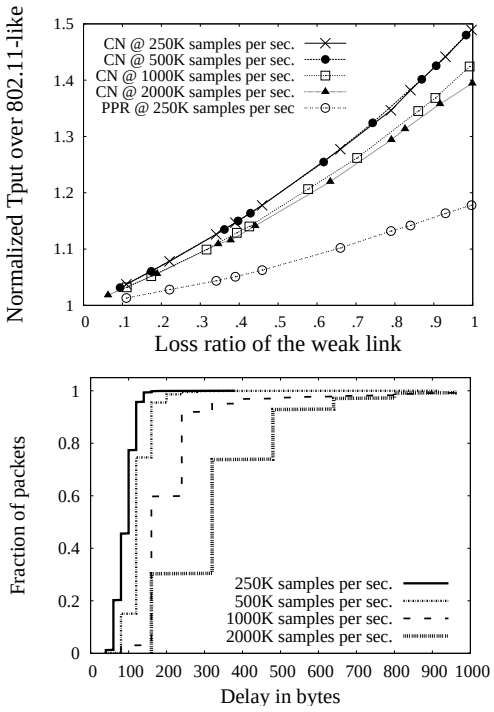

Figure 8: (a) Performance gains with CSMA/CN and PPR over packet receivers (b) CDF of notification overhead for CSMA/CN in bytes

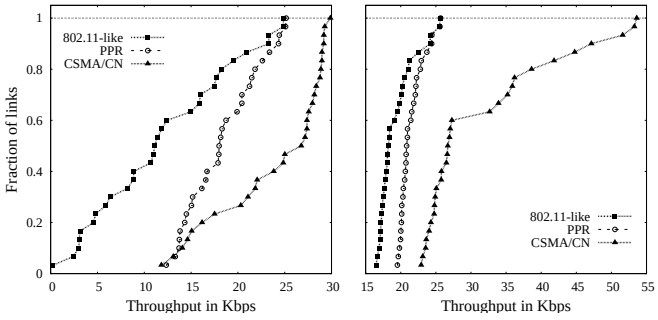

Figure 9: (a) CDF of throughput in Kbps where all nodes are continuously active. (b) CDF of throughput in Kbps with fixed size traffic (like FTP).

Figure 9(a) plots the CDF of the throughput achieved by each link across all our experiments. This experiment was performed with continuous traffic from all transmitters. While PPR outperforms the 802.11-like scheme, CSMA/CN performs even better. This is expected because the network is collision dominated, hence, aborting a failed transmission (prevention) is better than recovering from a failure (cure). Although overall throughput improves, we find that the weaker links in CSMA/CN do not benefit (the lower end of the CSMA/CN CDF). This is because the weaker links are continuously interrupted by interference, and hence, can never progress sufficiently. We verify this through Fig-

ure 9(b), in which transmitters have bursts of traffic. The stronger links finish their chunks quicker, allowing for the weaker links to advance without interruption.

4. LIMITATIONS AND ON-GOING WORK

Improved Correlation: We observed that signature correlation is reliable when the collision notification is no weaker than 18dB compared to the self-signal. This could be a limitation of CSMA/CN since collision may be more likely (hence detection/abortion more vital) when the receiver is far away from the transmitter. We are exploring approaches that can subtract the known self-signal, and apply the correlation on the residue. While this may appear to require complex interference cancellation [4], a simple and approximate modeling of self-signal may suffice as we do not need the actual bits of the notification signature.

Multiple Interferers: Our accuracy of collision detection through preamble correlation is done in a setting with only one interferer. We believe that preamble correlation works well even in the presence of multiple interferers as it has shown to be feasible in [5]. We plan to verify this further.

Decodable Feedbacks: In the current version of CSMA/CN, the collision notification is unable to convey more than a yes/no information. Additional information from the receiver (suitable bitrate, cause of failure, etc.) could certainly help the transmitter in deciding the appropriate response to the collision. Preamble correlation is inadequate to convey such feedback. One option for the receiver is to time the CN frame such that the transmitter can abort its transmission, and then start receiving the feedback information in the clear. We are investigating the merits of this possibility.

5. RELATED WORK

Avoiding Collisions: There have been numerous MAC protocols proposed for wireless networks [6]. A common feature of most of these schemes is that they avoid collisions by utilizing control frames or out-of-band busy tones. These schemes tend to be either quite conservative by reserving a large space around the communicating nodes or do not completely eliminate the collisions. Some studies have shown that enabling RTS-CTS reduces the overall throughput [7] and hence disabled by default in many deployments [8].

Recovering from Collisions: Apart from PPR mentioned earlier, ZipTx [3] makes use of known pilot bits to detect errors and recover the partial packets. We do not insert any known bits for detecting collisions. A receiver could apply interference cancellation [4] to recover the frame of interest by decoding the interfering transmission first and then canceling it out. However, this approach works only when the relative strengths of the signals at the receiver satisfy certain thresholds. ZigZag decoding [5] is a form of interference cancellation that recovers frames from repeated collisions. While this is a creative approach, it requires that the same set of frames be involved in multiple collisions.

Detecting Collisions: Authors in [9] enables a transmitter to distinguish between a fading and collision by having the receiver return the received bits. SoftRate [2] utilizes SoftPHY information to distinguish between collision and fading for rate adaptation. In contrast, CSMA/CN detects and aborts collisions on the fly.

Aborting Collisions: The only scheme that bears some similarity with CSMA/CN is [10]. Authors use an out of band control channel to transmit pulses for the purpose of indicating active transmissions. Transmitters sense the control channel to detect potential collisions, however, such decisions at the transmitter are not an accurate indicator of collision at the receiver. We use an *in-band* collision detection scheme at the receiver with explicit feed back to the transmitter to abort. To the best of our knowledge, this is the first prototype implementation of an in-band scheme for detecting and aborting collisions.

6. CONCLUSIONS

This paper proposes CSMA/CN to show that it is feasible and beneficial to abort an unsuccessful transmission with the aid of a collision notification from the receiver. The architecture is simple, while the additional hardware requirements are minimal. Several facets of CSMA/CN still need to be thoroughly explored through extensive implementation and experimentation before it can be considered a viable alternative to CSMA/CA. Nevertheless, we believe this work presents a preliminary yet promising first step in what could evolve into a new technology for future wireless networks.

7. REFERENCES

[1] K. Jamieson and H. Balakrishnan, "PPR: Partial Packet Recovery for Wireless Networks," in *ACM SIGCOMM*, August 2007.

[2] M. Vutukuru et al, "Cross-Layer Wireless Bit Rate Adaptation," in *SIGCOMM*, 2009.

[3] K. Lin, N. Kushman, and D. Katabi, "ZipTx: Harnessing Partial Packets in 802.11 Networks," in *Proc. ACM Mobicom*, 2008.

[4] D. Halperin, T. Anderson, and D. Wetherall, "Taking the sting out of carrier sense: Interference cancellation for wireless lans," in *MOBICOM*, 2008.

[5] S. Gollakota and D. Katabi, "Zig-Zag Decoding: Combating Hidden Terminals in Wireless Networks," in *Proc. ACM Sigcomm*, 2008.

[6] S. Kumar et al, "Medium Access Control protocols for ad hoc wireless networks: A survey," *Elsevier Ad Hoc Networks*, vol. 4(3), May 2006.

[7] Atheros, "802.11 WLAN Performance," .

[8] Broadcom, "Wireless LAN Adapter User Guide.," .

[9] S. Rayanchu et al, "Diagnosing Wireless Packet Losses in 802.11: Separating Collision from Weak Signal," in *INFOCOM*, 2008.

[10] Jun Peng et al, "A Wireless MAC Protocol with Collision Detection," *IEEE Transactions on Mobile Computing*, vol. 6(12), Dec 2007.