Vulnerability of WiFi's Noise Floor Calibration

Seongmin Kim and Jeongyeup Paek *Senior Member, IEEE*

***Abstract*—Most wireless technology, including WiFi, rely heavily on clear channel assessment (CCA) to avoid collisions, not only among the devices within the same technology, but also against cross-technology interference. If the CCA threshold is not configured properly, both the transmission and reception performance will be seriously affected with behaviors unexpected from the protocol's perspective. On the other hand, WiFi uses an adaptive CCA threshold based on a noise floor calibration algorithm to account for ambient noise floor changes and slight differences in hardware. However, this turns out to be a serious vulnerability of WiFi. In this work, we show that one can easily generate a wireless signal that can take advantage of the noise floor calibration algorithm of WiFi to inflate the CCA threshold and degrade performance significantly. The signal can be generated from a COTS Zigbee device, and if well designed, need not be too long nor strong. We show that WiFi is vulnerable to such attack, and more surprisingly, the network performance does not recover long after the signal disappears. We exemplify and verify our findings through extensive real-world experiments using 5 types of commercial WiFi NICs and 3 different WiFi APs to show that this is a critical problem that exists in reality and must be addressed.**

***Index Terms*—IEEE 802.11, Clear Channel Assessment (CCA), Cross Technology Interference, Internet of Things (IoT),**

I. Introduction

Wireless technology is prevalent in today's everyday life. In the upcoming Internet of Things (IoT) era, we will be surrounded by a vast number of wireless networks and devices, not only WiFi, but also Bluetooth, Zigbee, LoRa, WiSUN and many more [1]. These different wireless technologies have their own unique protocols to communicate among themselves, and cannot communicate with other technology[1], but can still interfere with others due to overlap in the frequency (e.g. 2.4GHz ISM band) [8]–[13]. Nevertheless, many technologies adopt a common technique (or at least the concept) of carrier sense multiple access (CSMA) to avoid collisions, not only among the devices within the same technology, but also against cross-technology interference by measuring the signal strength on the channel. There are many different ways of implementing this, but the concept of CSMA is widely adopted for its simplicity and effectiveness.

To enable CSMA, first and the most important thing we need is the clear channel assessment (CCA) mechanism and its threshold, also known as the energy detection (ED) threshold[2]. The baseline reference point of this CCA/ED threshold is the estimated noise floor, which could be a hard-coded value at design/manufacturing time, or a *continuously estimated and calibrated value at run-time*, the focus of our work. Suppose that this estimated noise floor, either hard-coded or calibrated at run-time, was set incorrectly or not set as desired. Then, the wireless interface will effectively become deaf; transmitters will be unable to assess the existence of other transmitters (clear channel), nor the receivers will be able to detect whether other devices are sending signals to it (energy detection). Thus, multiple transmitters may transmit simultaneously (CCA failure) and receivers will be unable to receive packets (ED failure), resulting in a malfunctioning wireless network.

Motivated by our real-world experiences, in this work, we show that this phenomenon exists in reality and can be reproduced. We show that a potential malicious attacker can easily neutralize a WiFi network by generating a wireless signal that can take advantage of the noise floor calibration algorithm of WiFi to inflate[3] the CCA threshold and degrade performance significantly. The signal can be generated from a commercial off-the-shelf (COTS) Zigbee device, and if well designed, need not be too long nor strong (well below jamming level). This phenomenon is due to the *noise floor estimation and adaptation mechanism of WiFi*, whose purpose is to account for ambient noise floor changes and slight differences in hardware (e.g. HW imperfections). Details of how this noise floor calibration algorithm should be implemented is left to the manufacturer in the IEEE 802.11 standard [14] "section 17.3, CCA requirements", and this turns out to be a critical vulnerability of WiFi.

More surprisingly, although the calibrated noise floor recovers back to the *normal* value after the noise-inflating signal disappears (to the value before the noise-inflating signal was introduced), the network performance (i.e. throughput) does not recover back to its original value for a long duration of time. This is due to an unanticipated behavior in the auto rate control algorithm of WiFi which adapts its modulation and coding scheme (MCS) to the packet losses experienced during the noise-inflated and CCA-malfunctioning time period. This is something that the software implementations of many popular WiFi chipsets did not expect to happen. Unfortunately, unless the firmware and device driver of WiFi NICs are fully released as open source by the manufacturer[4], it is challenging to know for sure exactly what interaction is occurring between the auto rate control and noise floor calibration of WiFi, and what is happening on the devices internally regarding noise

S. Kim and J. Paek are both with the School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea. (email: *shieldnet@cau.ac.kr, jpaek@cau.ac.kr*). J. Paek is the corresponding author.

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2020R1F1A1051282).

[1]There are some recent research efforts on cross-technology communication (CTC) [2]–[7], but only in a limited setting. More on this in SectionV-C.

[2]Some technologies treat CCA and ED thresholds separately, while others regard them as identical. WiFi (IEEE 802.11) uses one value for both.

[3]We use the verb 'inflate' to concisely express and mean 'increase or raise excessively beyond true value or normal operating range'.

[4]There is one open source firmware we found for AR9271, *open-ath9k-htc-firmware* [15], but it is experimental and does not provide noise floor information nor implement CCA calibration.

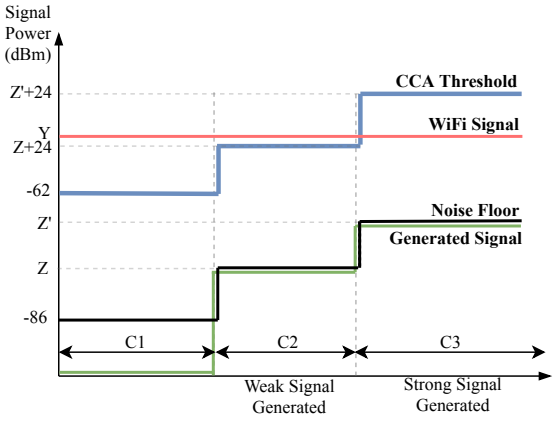


Fig. 1. Various situations with CCA threshold calibration.

floor estimation and CCA mechanism. Instead, we take an *indirect approach* to understand this problem. We show how aforementioned phenomenon can happen universally through extensive experiments using 5 different WiFi NICs and 3 different WiFi APs with different chipsets and device drivers.

The contributions of this paper are three fold:

- Identify the problem that one can easily manipulate the noise floor calibration mechanism of WiFi using a signal generated by a Zigbee device.
- Demonstrate how inappropriately calibrated noise floor impact WiFi's performance.
- Show that this problem is universal, the performance may not recover even after the interfering signal disappears, and thus the noise floor calibration is a critical vulnerability of WiFi that must be addressed. Once the root cause of the problem is identified correctly, solution is straightforward.

The remainder of this paper is organized as follows: Section II provides a brief background on WiFi's CCA mechanism. Then in Section III, we discuss the vulnerability of WiFi's noise floor calibration in detail through extensive experiments using various scenarios and devices to show the performance problem when a noise-inflating Zigbee signal is introduced into a WiFi network. We characterize the noise-inflating signal in Section IV, and suggest potential countermeasures in Section V. We conclude the paper in Section VII.

II. Background on WiFi's CCA

This section provides a brief background on WiFi's CCA and noise floor calibration algorithm.

A. Terminology

Noise floor is the measure of signal created from the sum of all noise sources and unwanted signals within a measurement system, where 'noise' is defined as any signal other than the one being monitored [14]. For WiFi, noise floor is the ambient/thermal noise of specific channel that the device senses, which may include artificial signal from other devices.

CCA threshold is the clear channel assessment threshold used for carrier sensing and energy detection. Carrier sensing is provided by the PHY layer[5], and is a measuring of the signal strength on the channel. If it is above a certain level, the medium is considered 'busy' [17].

MCS is the set of modulation and coding schemes used by WiFi, among which WiFi will select one to be used adaptively at run-time based on the link quality it experiences. It has direct impact on the physical data rate and error resilience of the WiFi link. Higher the modulation, better the throughput but more fragile to noise or interference. Higher coding rate is more robust to noise or interference, but lower the throughput. If a WiFi device experiences several packet losses or CRC errors, it will lower the MCS for reliability. If there are no losses for several consecutive packets, MCS will be increased for higher throughput.

IEEE Std 802.11-2016, section 17.3.10.6 mandates *CCA requirements*. According to the standard, CS/CCA mechanism shall detect a medium busy condition within 4 microseconds of any signal with a received energy that is 20∼24 dBm above the minimum MCS sensitivity. The exact value is *implementation dependent*. So, the CCA threshold (which is identical to energy detection threshold, ED) can be represented by,

$$\text{CCA threshold} = ED = NF + 24 \tag{1}$$

where the CCA threshold (ED) depends directly on the noise floor (NF) with a minimum NF value of -95 dBm mandated by the standard [14]. For example, minimum calibrated noise floor of the Atheros ath9k driver is -86 dBm [18], and thus the lowest CCA threshold of WiFi chips using ath9k driver is -62 dBm.

B. Possible cases with inflated noise floor

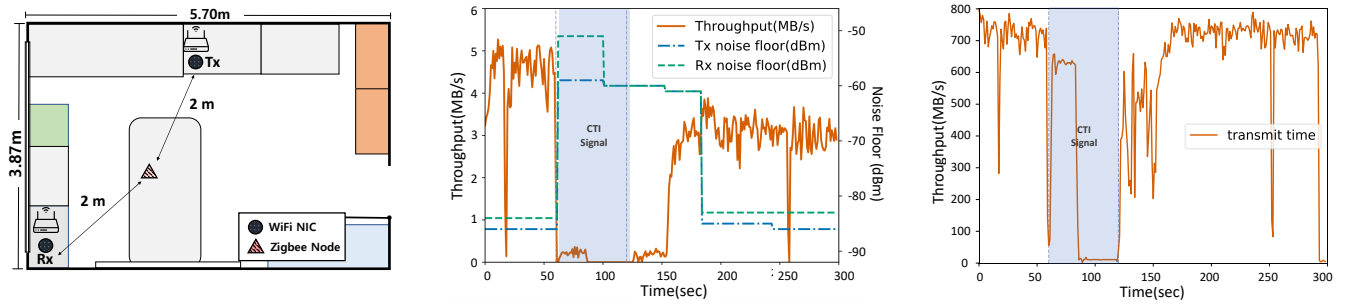
With noise floor calibration algorithm in place, let's enumerate the possible cases that may occur when interference signal is introduced to WiFi devices. For the purpose of illustration, consider Fig. 1 where the baseline noise floor is -86 dBm, CCA threshold is calculated as NF+24 (as in Eq.(1)), Y represents the received signal strength (RSS) of legitimate WiFi signal, and Z (and Z') represent the RSS of interfering signal when introduced. Then, we have three cases;

- **C1** : there is no interfering signal (*'normal state'*),
- **C2** : interfering signal with RSS Z such that $Z + 24 < Y$
- **C3** : interfering signal with RSS Z' such that $Z' + 24 > Y$

In case C1 where there is no interfering signal, noise floor is set to the default minimum value (set by the implementation), and legitimate WiFi signals have sufficiently larger strength than the CCA threshold. This is the *'normal state'* in which we expect (and want) the protocol to operate in. In case C2, an interfering signal with RSS of $Z >$ -86 dbm has increased the estimated noise floor of WiFi device to Z dBm. Nevertheless, the calibrated CCA threshold $(Z + 24)$ is still below the strength of legitimate WiFi signal (Y), and thus CCA will operate as intended.

However in case C3, stronger interference signal (of strength Z') has increased the CCA threshold $(Z' + 24)$ beyond the

[5]In fact, Carrier Sense (CS) mechanism in WiFi consists of both physical CS and virtual CS. However, Virtual CS [16] is out of the scope of this work.



(a) Experiment setup

(b) Throughput of WiFi flow, and the noise floor estimated by WiFi Tx and Rx.

(c) Transmission time per second (at Tx device) represent channel utilization.

Fig. 2. Experiment setup and results with two AR9380 WiFi devices in ad-hoc mode and one interfering Zigbee node in the middle. Shaded area in the plots represent the duration of interfering signal.

strength of legitimate WiFi signal (Y). In this case, WiFi Tx will always assess the channel as 'clear' since the strength of any other signal will always be less than the CCA threshold. For the same reason, WiFi Rx will also be unable to detect the existence of any WiFi signals, resulting in repeated collisions and failure of receptions. For example, if we can generate an interfering signal that is perceived as RSS of -60 dbm at the WiFi devices, then legitimate WiFi signals would need to be stronger than -36 dBm for the WiFi network to operate correctly. Otherwise, significant compromise in performance is inevitable. We will show this in the next section.

III. Vulnerability Analysis of WiFi's CCA

This section investigates the vulnerability of WiFi's CCA and its noise floor calibration algorithm, and presents our findings through extensive experiments using various scenarios and devices.

A. The Problem – Main Result

It is well known that the IEEE 802.15.4 (a.k.a. Zigbee) overlaps in 2.4 GHz frequency band with the IEEE 802.11 (a.k.a. WiFi), and cross-technology interference (CTI) is a challenging problem between the two despite both adopt CCA to avoid collisions [19]. It is also well known that due to lower transmit power (-21∼5 dBm for Zigbee, 20 dBm for WiFi) and data rate (which results in longer RX-TX/TX-RX switching times), Zigbee is usually the victim of the challenge [8]–[10]; i.e. it is usually the Zigbee network that suffers more when competing with WiFi. However, even with this relatively inferior David versus Goliath like contention, it is possible for a Zigbee device to disrupt WiFi's communication by generating a CTI signal that inflates the noise floor experienced by WiFi devices. A single 127 byte packet may not be long enough, but a Zigbee signal with adequate length (∼1 sec) and power can raise the CCA threshold of WiFi beyond the appropriate operating range and degrade its performance significantly in terms of both loss and throughput. This lengthened Zigbee signal, also known as the 'continuous wave (CW)' feature in many IEEE 802.15.4 chipsets, can be generated by several popular COTS Zigbee devices such as TI CC2650-LaunchPad [20], NXP FRDM-KW36 [21], and Nordic nRF52840 [22], just to name a few.

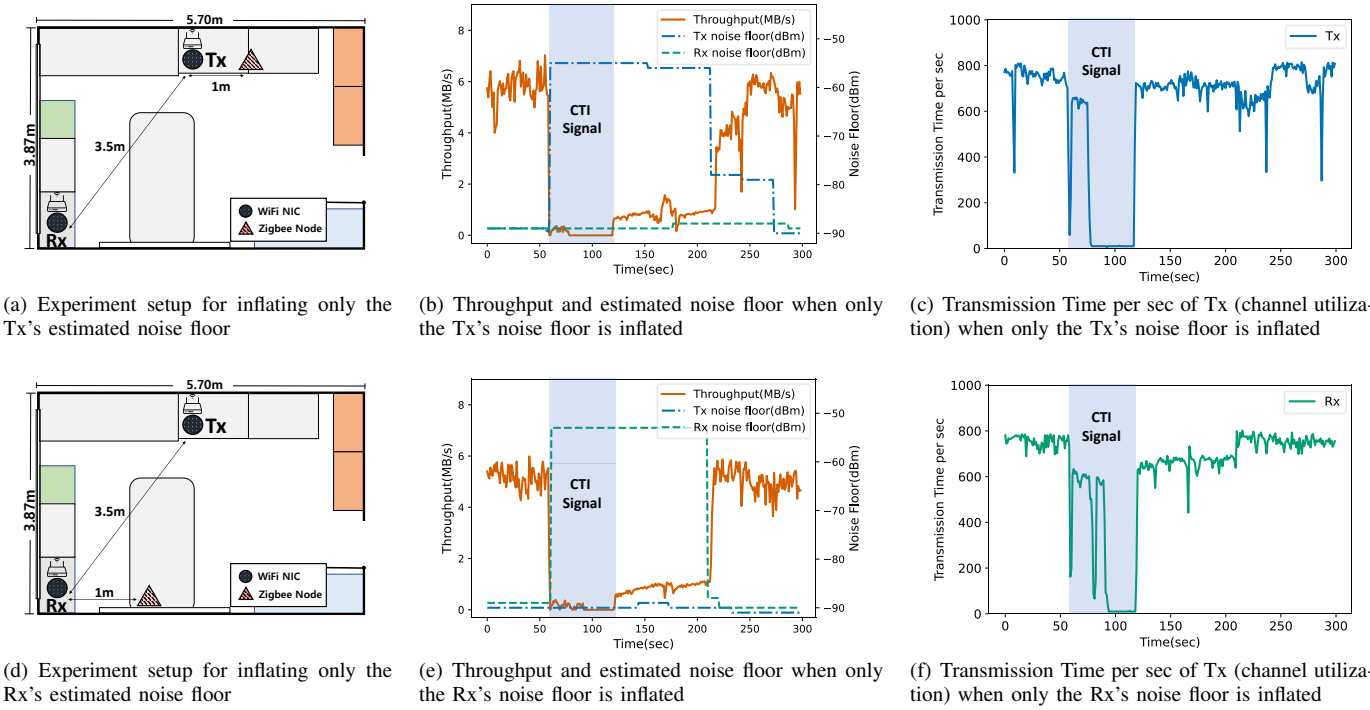
To exemplify this in a real-world setting, we have conducted an experiment using two WiFi devices and one Zigbee device in an office environment. Fig. 2(a) illustrates the setup, where all devices are placed on tables (∼1 m above ground) and the distance between WiFi Tx and WiFi Rx is ∼4 meters. Atheros AR9380 chipset [23] with ath9k driver[6] is used for WiFi in ad-hoc mode, and TI CC2650-LaunchPad [20] is used as the Zigbee device[7]. Using this setup, WiFi Tx will transmit packets to WiFi Rx continuously using *iperf*, during which we measure the throughput of WiFi flow, noise floor estimated by each WiFi device, and the channel utilization of WiFi Tx in terms of transmit time per second. While doing so, starting from t = 60 seconds into the experiment, the Zigbee node will generate a continuous wave signal at Tx power of 5 dBm and stop at t = 120 sec. The purpose of this experiment is to see what happens on the WiFi devices when there is an interfering signal from a Zigbee device.

Fig. 2(b) plots the throughput achieved by the WiFi flow, as well as the noise floor experienced by the two WiFi devices, Tx and Rx. Furthermore, Fig. 2(c) plots the channel utilization in terms of time spent for transmission by the WiFi Tx per second. Before the start of the Zigbee signal (t = [0:60] sec, *'normal state'*), the noise floor estimated by WiFi devices are around -83 ∼ -86 dBm, the throughput of WiFi flow is around 4.5MB/s, and the transmission time per second is around 750ms (∼75% channel utilization).

At t = 60 sec, the Zigbee node starts generating its continuous wave signal. Immediately, the transmission time per second (Fig. 2(c)) drops sharply for a very short time period (between t = 60∼61 sec). This is when the WiFi Tx notices the external signal via CCA and backs off its transmissions (seizes to transmit). Shortly after that (t > 61 sec), WiFi Tx starts transmitting again (Fig. 2(c)) despite the fact that Zigbee signal still persists. The reason for this can be seen in Fig. 2(b) where the noise floor estimated by the WiFi devices has risen

[6] *ath9k* is an open source device driver for Qualcom Atheros WiFi NICs supporting IEEE 802.11 abg/n.

[7] We used channel 11 for WiFi (2462 MHz) and channel 22 for Zigbee (2460 MHz).



(a) Experiment setup for inflating only the Tx's estimated noise floor

(b) Throughput and estimated noise floor when only the Tx's noise floor is inflated

(c) Transmission Time per sec of Tx (channel utilization) when only the Tx's noise floor is inflated

(d) Experiment setup for inflating only the Rx's estimated noise floor

(e) Throughput and estimated noise floor when only the Rx's noise floor is inflated

(f) Transmission Time per sec of Tx (channel utilization) when only the Rx's noise floor is inflated

Fig. 3. Two scenarios: noise floor of Tx and Rx inflated in isolation. Subfigures in the upper and lower row present each cases respectively.

to -60 ∼ -51 dBm; i.e. *signal from the Zigbee node has raised the noise floor of WiFi devices*. Since the noise floor ascended, both the CCA and ED thresholds must have increased as well to -36 ∼ -27 dBm (Section II), beyond the signal strength of legitimate WiFi signal. Thus the WiFi Tx becomes deaf against external signal (CCA failure), and the WiFi Rx is unaware of the transmission from the Tx (ED failure). Because the WiFi Tx is incorrectly assessing the channel as clear, it proceeds with its transmissions (t = [61:75] sec in Fig. 2(c)) only to fail (t = [61:75] sec in Fig. 2(b)).

Soon after (t = [75:120] sec), WiFi Tx realizes that it is experiencing a lot of packet losses (or ACK losses), and reduces its transmissions to a very low rate (Fig. 2(c)). Although ath9k driver does not provide MCS information in ad-hoc mode, we conjecture that the MCS has gone down to the lowest level at this point. Note, however, that the transmissions have not stopped completely; Tx is still attempting sporadically (t = [90:120] sec in Fig. 2(c)) only to fail at the receiver with zero throughput (t = [90:120] sec in Fig. 2(b)), which will keep the Tx at the lowest MCS and backoff-and-retry rate.

At t = 120 sec, the Zigbee node stops generating the noise-inflating signal. At this point, you would expect the noise floor to go down immediately, just like it increased immediately at around t = 60∼61 sec. Unlike our expectation, however, it does not. The noise floor of both the WiFi Tx and Rx remain at around -60 dBm for another 60 seconds till t = 180 sec, after which the noise floor returns back to where it was before the noise-inflating Zigbee signal was introduced (-85 ∼ -83 dBm).

During the first half of this period (t = [120:150] sec), WiFi transmissions start to succeed slowly. This is because, although the noise floor has not returned back to its original value yet, external interference at the Rx side has actually been removed, and this allows some packets to be detected and decoded correctly at the Rx side. This process is slow because Tx was in the lowest MCS and backoff-and-retry rate during t = [90:120] sec. Nevertheless, these packet delivery successes will help the Tx to gradually increase its transmission rate (t = [120:150] sec in Fig. 2(c)). Only after the second half of this period (t = [150:180] sec), the channel utilization (Fig. 2(c)) of the WiFi Tx returns back to where it was before the noise-inflating signal was introduced (∼75%).

Nevertheless, the throughput achieved by the WiFi flow after it has gone through the noise-inflating signal (∼3 MB/s) still does not fully recover to where it was before the signal was introduced (∼4.5 MB/s). It achieves only 2/3 of what it should, and this trend continues even after t = 180 sec when the noise floor of both the Tx and Rx returns back to the normal value. More surprisingly, *it continues indefinitely (for some WiFi NICs) until unless we disconnect and reconnect the WiFi association*. This is a serious problem, and it is an unexpected consequence of the noise floor calibration and MCS adaptation algorithms of WiFi.

Since there was no competing WiFi traffic in this experiment, the reasons for the throughput reduction are as follows; When the noise floor is inflated, Tx transmits regardless of interfering signal because its CCA is malfunctioning. When there is interfering signal, Rx cannot decode the packets correctly. When the noise floor is high, Rx cannot detect the transmissions correctly. If the Rx is not returning ACKs, Tx will backoff and reduce its MCS level. Finally, it takes some time for the noise floor and MCS level to return back to the normal state. Then the remaining questions that need to be

answered are,

- Can the problems at Tx and Rx be reproduced in isolation?
- Does this phenomenon occur for other WiFi devices and configurations (ad-hoc mode vs. AP mode)?
- How long does it take for an interfering signal to inflate the noise floor experienced by WiFi, and how long does it take to recover back to the original value?
- What are the characteristics of such noise-inflating signal?
- What happens when there are competing WiFi traffic?

We discuss these question in the following subsections.

B. Tx and Rx, noise-inflated in isolation

In Section III-A, the problem was identified for the case where the noise floor of both the WiFi Tx and Rx was inflated simultaneously. In this subsection, we inflate the noise floor of Tx and Rx separately to investigate its impact in isolation since they will have different influence on performance.

For this purpose, we use setups as in Figs. 3(a) and 3(d). We use two WiFi devices (circle) for Tx and Rx at 3.5 m distance, and an interfering Zigbee node (triangle) is placed at 1 m distance from the target WiFi device (either Tx or Rx) with Tx power of -6 dBm. The setup was intended to inflate the noise floor of just one WiFi device, either Tx or Rx, but not both. Subfigures in the top and bottom row of Fig. 3 each represent either Tx or Rx noise-inflated scenario, respectively.

Figs. 3(b) and 3(e) plot the throughput and estimated noise floors of Tx and Rx, and Figs. 3(c) and 3(f) plot the transmission time per second for each scenario. It can be seen that when the interfering Zigbee signal is introduced at t = 60 sec, only the estimated noise floor of either Tx or Rx is inflated to around -60 dbm (from -90∼-84 dbm), but not both.

Regardless, the overall trend of the throughput and channel utilization are very similar to Fig. 2 in Section III-A. Throughput is decreased significantly for both scenarios immediately after the interfering signal has started, and recovers back ∼2 minutes after the noise-inflating signal disappears. Transmission time per second drops sharply for a very short time period at the moment when the interfering signal is first detected (between t = 60∼61 sec), but blind transmission attempts (CCA failure due to inflated noise floor) are made even when Zigbee signal persists (t = [61:75] sec) only to fail. Then follows a period of lowest transmission rate during t = [90:120] sec, after which transmissions resume and channel utilization returns back to where it was before (∼800 ms/s).

However, the two cases have slightly different (and interesting) reasons for the drop in performance. When the interfering signal is introduced to the Tx, Tx will first assess the channel as busy and back off. Once the noise floor of Tx is inflated, Tx will assess the channel as clear despite interference, and resume its transmissions (t = [61:75] sec in Fig. 3(c)) only to fail due to interference. After a period of frequent packet losses, Tx will reduce its MCS level, and will continue its transmissions at a lower PHY rate (t = [75:120] sec in Fig. 3(c)). On the other hand, when the interfering signal is introduced to the Rx, Rx will have difficulties in decoding the packets that may be corrupt due to interference. Once the noise floor of Rx is inflated, Rx will also have difficulties in detecting the packets sent to itself, although it was partially receivable in our 3.5m setup. Either way, Rx will not return ACKs properly, which will result in timeouts, backoffs, and lowered MCS at the Tx (t = [75:120] sec in Fig. 3(f)).

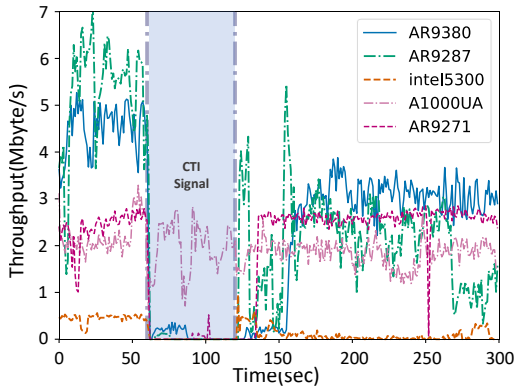


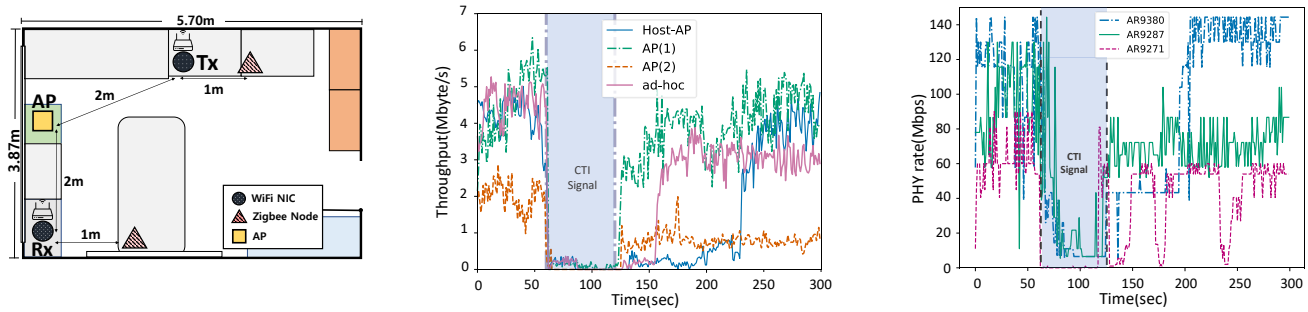
Fig. 4. Throughput comparison of five different types of WiFi NICs when noise-inflating signal is introduce for 60 seconds.

C. Impact of WiFi NIC type

Next, we conduct experiments using five different WiFi devices to show the impact of NIC type and whether the problem is universal in other WiFi devices. We use AR9271 (with ath9k-htc device driver), AR9380 (ath9k), Intel wireless 5300 (iwlwifi), iptime AU1000UA (proprietary), and AR9287 (ath9k) for the five WiFi NIC types. All other setups including the topology (Fig. 2(a)) are identical to our first experiment in Section III-A, and same type of devices are used in pairs for Tx and Rx. As before, an experiment consisted of 3 periods: First period (t = [0:60] sec) is the 'normal state' before the interfering signal is introduced, second period (t = [60:120] sec) is when the interfering signal is present, and the last period (t = [120:300] sec) is after the interfering signal has stopped.

Fig. 4 plots the change of throughput over time for five different kinds of NICs in ad-hoc mode. In all cases, we see significant impact on throughput when interfering signal is being emitted. In fact, except for A1000UA, other four types show throughput of close to zero. When the interfering signal stopped, it took around 30∼40 sec on AR9380, AR9287, and AR9271 for the throughput to increase back up. For three out of five device types (AR9380, AR9287, and Intel5300), however, the throughput is not fully recovered indefinitely[8] even after the interfering signal disappeared and noise floor has returned back to the values before the interfering signal was first introduced. These results are consistent with the those in Section III-A, which indicates that this is not a problem of just one device type. Only one device (A1000UA) out of five maintained similar throughput before and after experiencing the interfering signal. Although we could not confirm what is happening in the A1000UI internally due to lack of API and proprietary device driver, our observations indicate that A1000UI is using a fixed threshold for CCA.

[8]Although the plot shows only up to 300 sec, we have monitored it for much longer durations, sometimes even overnight, to confirm our claim.



(a) Experiment setup for AP mode

(b) Throughput comparison of three different types of AP, and AP mode vs. ad-hoc mode

(c) MCS level dynamics when interfering signal is introduced, for 3 NIC types in AP mode with AP(1).

Fig. 5. Experiment setup and results for the AP mode evaluation using three different APs. Shaded area represent the period of interfering signal.

D. Impact of WiFi mode – Ad-hoc vs. AP

So far, all experiments were done in *ad-hoc mode* of WiFi to see the behavior at Tx and Rx node independent from any influence of an access point (AP). In this subsection, we conduct experiments in *AP mode* using three different APs and compare their results. For this purpose, we use two commercial APs, an ASUS 'RT-AC58U' WiFi router (AP(1)) and a Cisco 'Linksys E1200' WiFi router (AP(2)), both supporting IEEE 802.11n in 2.4GHz band. We also use an AR9271 NIC as an AP running *'Host-AP'* driver [24]. For the end devices (Tx and Rx of data traffic), we use the same AR9380 NIC as the first experiment in Section III-A. Fig. 5(a) depicts the experiment setup.

Fig. 5(b) plots the throughput over time for three different APs as well as the result from the ad-hoc mode experiment. In all cases, throughput plunges to almost zero during t = [60:120] sec when interference signal is introduced. However, their behavior of recovery after the interfering signal disappears is slightly different for each case. AP(1) and AP(2) cases partially regain their throughput immediately, but it takes around 110 sec for Host-AP to regain its throughput (t = 225 sec) whereas it took around 40 sec for ad-hoc mode. Furthermore, AP(2)'s throughput is never fully recovered indefinitely, as is for the ad-hoc mode case. The difference comes from whether and how the AP is calibrating its noise floor, and how it adapts its MCS in response to that. These results show that the problem exist not only in the ad-hoc mode of WiFi but also in the AP mode for popular commercial APs.

E. PHY rate (MCS) adaptation in AP mode

To understand how PHY rate changes over time in response to noise-inflating interference signal, we conducted additional experiments using three different types of NICs in *AP mode* with AP(1)[9] on the same setup as Fig. 5(a). It can be seen from Fig. 5(c) that PHY rates drop sharply to the lowest value during the period of interfering signal. Thereafter, once the interfering signal disappears, it takes some time for the PHY rates to recover back, sometimes only partially. Specifically, it took over 100 sec for AR9380 to return back to its original PHY rate. For AR9271 and AR9287, although their PHY rates did increase immediately after the interfering signal disappeared, they never fully returned back to their original values before experiencing the noise-inflating signal.

TABLE I
PERFORMANCE REDUCTION RATIO (BEFORE AND AFTER THE NOISE-INFLATING INTERFERENCE SIGNAL) FOR 18 DIFFERENT SCENARIOS.

Device \ Setup	AP-1	AP-2	Host-AP	Ad-Hoc
AR9380	12.7%	42.8%	59.0%	41.3%
AR9287	13.0%	52.2%	13.8%	61.5%
intel5300	39.6%	43.5%	-	83.2%
AR9271	33.4%	43.3%	48.7%	1.6%
A1000UA	4.2%	5.4%	-	3.2%

In fact, to see the overall throughput reduction ratio after experiencing an noise-inflating interference signal, we have experimented on all 18 possible setups that we can create using five different WiFi NICs and two different connection modes with three different APs[10]. Table I summarizes the result. For example, if we use AR9380 as the WiFi Tx and Rx going through AP(1), then the throughput is degraded 12.7% compared to the normal state after going through the interference signal. It can be seen that 14 out of 18 cases had throughput reduction of more than 10%.

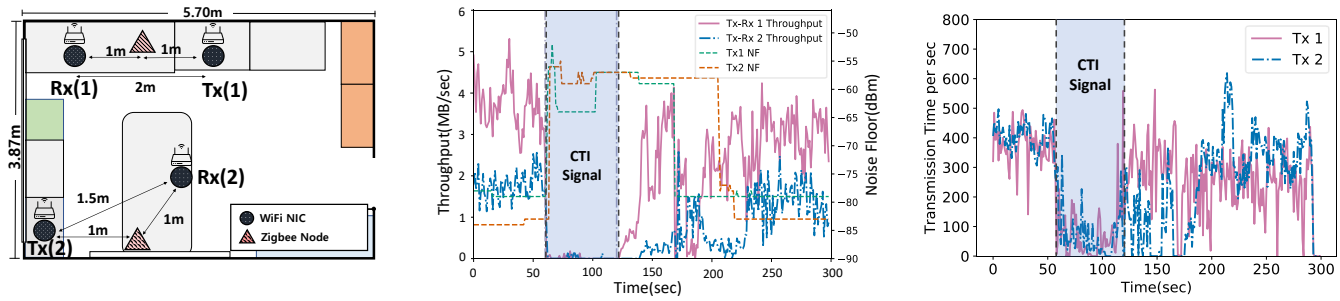
F. Multiple concurrent WiFi flows

So far, we have investigated the noise-inflation problem in various scenarios but all with a single WiFi flow only. In this subsection, we consider a scenario where two concurrent WiFi flows are sharing the channel. Fig. 6(a) depicts the experiment setup. Two pairs of WiFi Tx and Rx nodes (AR9380 and AR9271, respectively) communicate in ad-hoc mode, and two Zigbee devices (TI CC2650) are placed close to each WiFi Tx-Rx pair.

Fig. 6(b) plots the throughput of two WiFi flows as well as the noise floor estimated by each transmitter node Tx1 and Tx2. Fig. 6(c) plots the transmission time per second which represent the channel utilization for each transmitter. If we

[9]MCS information was not available in ad-hoc mode on ath9k driver.

[10]Host-AP [24] did not support intel5300 and A1000UA.



(a) Setup for 2-flows experiment

(b) Throughput of 2 WiFi flows, and the estimated noise floor at the two transmitters Tx1 and Tx2.

(c) Transmission time per second (channel utilization) of the two transmitters Tx1 and Tx2.

Fig. 6. Experiment setup and results for 2 concurrent WiFi flow case. Shaded area represent the period of interfering signal.

compare this with the single flow case in Fig. 2, other than the fact that the channel (throughput and utilization) has been shared by the two contending flows, the overall trend looks very similar. During the interfered period (t = [60:120] sec), the estimated noise floor jumps from -85∼-80 dbm to around -60 dbm (which means CCA threshold of -36 dbm), and the throughput of both flows drop close to zero. Even when the interfering Zigbee signal stops at t = 120 sec, it takes another 30∼40 sec for the throughput to ramp up again. Furthermore, it takes around 50∼100 sec for the estimated noise floor to drop back to the original value. Most importantly, the throughput (both individual and aggregate) never fully recovers back to where it was before the noise-inflating signal despite the fully recovered aggregate channel utilization. Thus, it can be said that the findings from the single flow experiments still hold for the multi flow scenarios.

IV. Characterization of Interfering Signal

In the previous section, we have shown that an interfering Zigbee signal can raise the noise floor perceived by WiFi and seriously impact its performance. However, it does not mean that *any* Zigbee signal will cause such problem. Thus, in order to understand when such problem occurs in a practical setting, we characterize the problem-causing interference signal in this section.

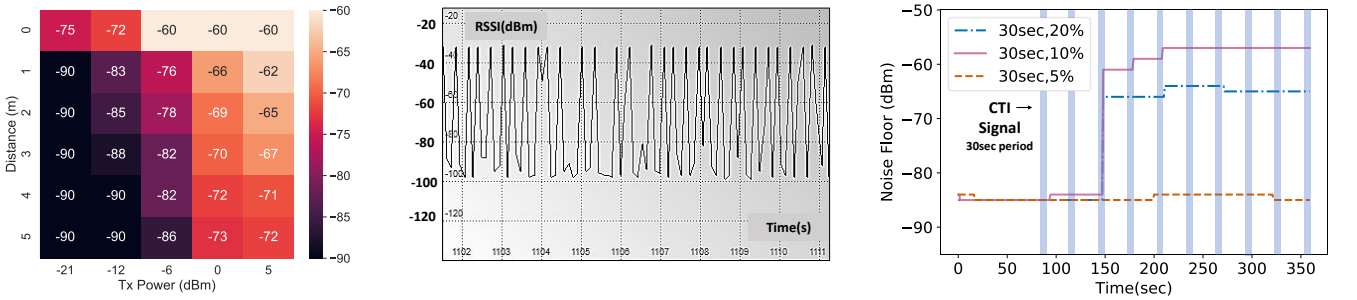
Consider an office environment (e.g. Fig. 5(a)) where there is a WiFi AP and users connect to the Internet via wireless LAN through this AP. This is a realistic and popular setting which most of us are familiar with today. Assuming that WiFi devices are using the default TX power of 20 dBm and most users are within ∼10 m radius from the AP, the perceived RSS for legitimate WiFi device will be roughly in the range of -20∼ -50 dBm. That means, based on Eq.1 and Fig. 1, the noise floor estimated by the WiFi devices should be kept sufficiently below -74 dbm for this WiFi network to operate correctly. Otherwise, the interfering signal will have impact on WiFi's performance.

From the opposite point of view, a malicious attacker who wishes to launch a denial-of-service (DoS) attack to this WiFi network would need to generate a 2.4GHz signal that inflates the noise floor estimated by WiFi above this point. Then, in order to find and characterize such an interfering signal, we have conducted a series of experiments while varying the distance, transmission power, and duty-cycle of the interfering signal.

In the first experiment, a Zigbee device (CC2650) is placed at [0, 5]m distances with 1 m interval in line-of-sight from a WiFi device (AR9380), and five different Tx power setting was chosen from [-21, -12, -6, 0, 5] dBm. Then for each distance-power configuration, interference signal was generated continuously, i.e. with 100% duty cycle, while the value of WiFi's estimated noise floor was measured at one second interval. Fig. 7(a) plots the average noise floor estimated at the WiFi device. Intuitively, it increases with Tx power and decreases with distance. This is obvious. However, there are two things to note here. First is that the maximum value of calibrated noise floor is limited to -60 dBm even if the RSS perceived by the device has a higher value. This is an implementation dependent decision made by the device driver of the WiFi NIC [18]. Secondly, interfering signal with TX power of 0 dbm or higher can impact the performance of WiFi even at 5 m distance or further. This means that an attacker need not be very close to the victim device when launching a DoS attack.

Next, we consider a more realistic scenario where instead of transmitting a continuous signal for a long duration, an attacker uses a short bursts of duty-cycled signal to avoid detection. This is similar to what is known as stealthy jamming [25]. Fig. 7(b) shows an example of a duty-cycled Zigbee signal generated from a CC2650 and measured at 100 ms polling interval using RF studio [26]. This example has 250 ms cycle-time and 50% duty-cycle (i.e. signal on for 125 ms, and off for 125 ms). Using this method, we generated an interfering signal with [5, 10, 20]% duty-cycles and 30 second cycle-time, starting from t = 90 sec into the experiment, and measured the estimated noise floor on the WiFi devices. For example, 5% means the signal was on for 1.5 sec, and off for 28.5 sec, every 30 seconds. Fig. 7(c) plots the calibrated noise floor at the WiFi devices over time. It shows that as low as 10% duty-cycled (i.e. 3 seconds in every 30 seconds) interfering signal is sufficient to inflate the noise floor of WiFi devices and compromise communication performance.

Of course, to successfully accomplish this attack with such a low duty-cycle, the attacker needs two hints. First hint is the noise floor calibration algorithm of the target NIC; e.g. Alg. 1 for *ath9k* device driver. Although this is implementation de-



(a) By distance and Tx power

(b) Interfering signal example at 50% duty-cycle with 250ms period.

(c) Calibrated noise floor over time when interfering signal with various duty-cycle is introduced.

Fig. 7. The relationship between noise floor vs. distance, Tx power, and duty-cycle.

Algorithm 1: Noise floor estimation and calibration algorithm of *ath9k*

```
SIZE ← 5;
DEFAULT_NOISE ← −95;
nf_history[SIZE];
noise_floor ← DEFAULT_NOISE;
now_index ← 0;
while not NF calibration error occurred do
    nf_history[now_index] ← noise_value;
    if not nf_history is full then
        noise_floor ← nf_history[now_index];
    else
        nf_buffer ← nf_history;
        sort( nf_buffer );
        noise_floor ← nf_buffer[ SIZE / 2 ];
    end
    now_index ← (now_index+1) %5;
    Wait 30 seconds;
end
```

pendent, *ath9k* uses *historical* noise floor (NF) measurements in a circular array of size 5, measured every 30 seconds, and adopts the median as the calibrated noise floor. This is the reason why noise floor jumped at t = 150 sec (third 30 sec cycle) instead of t = 90 sec (first cycle) in Fig. 7(c); it was waiting for a third high value as the median in a size-5 array. Second hint is the starting time boundary of a NF measurement cycle, as shown as shaded rectangles in Fig. 7(c). It turns out that the NF measurement cycle period starts from the time when WiFi association is made, and it is possible for an attacker to figure this out by sniffing the association frames on promiscuous mode with packet capture tools [27]. Thus we can conclude that an adversarial can easily neutralized a WiFi network at a distance by generating a Zigbee signal with a reasonably low transmission power and as low as 10% duty-cycle. This is a serious threat to WiFi.

V. Potential Countermeasures

The main contribution of this work is in identifying the problem of WiFi's noise floor calibration mechanism (conforming to IEEE 802.11 standard), and to show how vulnerable it is against unintended interference signal or attack. We have shown that it is a critical problem which seriously affects network performance (for duration longer than the interfering signal), and also a universal problem that must be addressed. Once and if the root cause of the problem is identified precisely, practical solution is straightforward if not trivial. We discuss a few of those potential countermeasures in this section. However, please note that the countermeasures are simple because the problem has been identified, and our contributions are in identifying the problem, not in proposing these straightforward solutions.

A. Bounds on NF and CCA threshold

A naive method to address the aforementioned problems caused by dynamic adaptation of CCA threshold is to not do dynamic adaptation at all: that is, use a fixed constant value as the noise floor and CCA threshold. This is a quick-and-simple fix that can be adopted easily and will work in most cases since ambient noise is usually measured around -105∼-90 dBm in most environments. However, this method nullifies the original motivation and intent of dynamic noise floor calibration, which is to adjust for the actual differences in measurement values of ambient noise and also the slight differences in radio hardware.

Then, a better alternative would be to allow the dynamic adjustment but put an upper bound on NF that is close to -90 dbm since the problems that we have observed are caused by the noise floor being significantly inflated (e.g. -60 dbm). This way, we have the advantage of dynamic calibration within a plausible (-95 ∼ -85 dbm) region while blocking the chances of significant inflation. This solution would be sufficient for most cases at the cost of not being able to detect rare high ambient noise situations in which WiFi would not work well anyway even if noise is real and legitimate.

B. Longer and randomized NF measurement period

To exploit the vulnerability discussed in this work efficiently and effectively, an attacker would need two hints as described in Section IV; (1) the noise floor calibration algorithm of the target NIC, and (2) the starting time/boundary of a NF measurement cycle. Then the natural defence would be to

obscure these hints. If we take Alg. 1 as an example, we can first randomize the NF measurement times instead of doing it at periodic 30 second intervals. We can also increase the size of historical NF buffer, and take the median of the lower half of the values instead of the whole set. This way, with high probability, an attacker would have no way other than to jam the channel completely. There is no defence against well-funded powerful wideband jamming attacks [28], but hopefully detection mechanism can take actions [29], [30].

C. Cross-technology communication (CTC)

Direct communication between heterogeneous IoT protocols can also be a potential solution to solve the cross technology interference (CTI) problem. For example, if WiFi and ZigBee can decode the packets of each other, this will enable coordination of channel access among those devices. Many recent studies have shown potential to support direct communication between WiFi, Bluetooth, and ZigBee devices [2]–[7] These pioneering designs enable cross-technology communications by embedding bits into various side-channels such as packet (or symbol) transmission time-frequency shifts and patterns. However, these techniques have only been explored under limited lab settings, many do not run on commercial devices, and will help in resolving the CTI issue *if and only if the two devices are willing to coordinate*. Thus, with the current noise floor calibration algorithm of WiFi, CTC cannot resolve the problem if the CCA threshold is raised excessively due to an unknown signal.

VI. Related Work

To the best of our knowledge, there is no research paper that discusses the vulnerability of WiFi's noise floor calibration and CCA mechanism. There is one work by Ock *et al.* [10] which briefly mentions such observation while designing a busytone scheme to protect Zigbee network from WiFi by exploiting CTI to combat CTI, but it does not delve into the problem and simply mentions it as a peculiarity of the device they used. However, our work provides in-depth measurement-based analysis to investigate the reasons behind such behavior, and show that the problem exists in many popular WiFi devices and scenarios.

CTI and co-existence issue between WiFi, Zigbee, and Bluetooth has been discussed in numerous prior work. For example, Han *et al.* [31] considered a smart home scenario with WiFi and Zigbee devices, and modeled the interference and CSMA algorithm to find and adapt proper CCA mode to get better performance. Zhu *et al.* [32] discussed the interference issue in heterogeneous multi-radio network, and proposed a coexistence-aware TXOP adaption which improves transmission efficiency in a WiFi/Bluetooth dual-radio device. Bluetooth and WiFi integration [33] and . Wojtiuk *et al.* [33] analyzed interference between Bluetooth and WiFi, and suggested collaborative MAC-level switching between the two technology to eliminate the interference. BlueCoDE [34] also investigated the interference issue between WiFi and Bluetooth, and proposed bonding multiple Bluetooth channels for improving throughput of both technologies.

Related to the perceived noise floor, Gokturk *et al.* [35] suggested a channel selection algorithm that measures SINR and searches for the channel with lowest noise floor to achieve maximum physical data rates. Lee *et al.* [36] modelled environmental noise level using noise signatures from real world measurements in order to accurately simulate wireless packet delivery in simulators such as TOSSIM, ns2, EmStar, etc.

Several prior work (including aforementioned) discussed the interference and co-existence issue among different wireless technologies to suggest improvements [8]–[13] or propose CTC [2]–[7], but none of them have discussed the vulnerability of WiFi's noise floor calibration algorithm and its implication on CCA mechanism.

VII. Conclusion

WiFi implements a noise floor calibration mechanism as suggested and permitted in the IEEE 802.11 standard. The original intention was to adapt to ambient noise floor changes and account for slight differences in hardware. Unfortunately, however, this turns out to be a critical vulnerability of WiFi. Estimated noise floor can be manipulated easily using a signal generated by a COTS Zigbee device with a duty-cycle as low as 10%, and this inflated noise floor deafens either the Tx or Rx or both to have significant impact on WiFi's performance, sometimes down to zero throughput. More surprisingly, the performance may not recover fully even long after the interfering signal disappears (recovers only 1/2∼2/3), sometimes never before re-association. The problem is universal in the sense that it has been identified in several popular WiFi devices and APs, for both ad-hoc and AP mode. Thus the noise floor calibration is a critical vulnerability of WiFi that must be addressed. Fortunately, once the root cause of the problem is identified precisely, solution is a straightforward implementation. To the best of our knowledge, this work is the first to present these findings.

References

[1] "The Growing Trend of IoT Devices," accessed: 2020-11-10. [Online]. Available: https://cultureofgaming.com/the-growing-trend-of-iot-devices
[2] S. M. Kim, S. Ishida, S. Wang, and T. He, "Free side-channel cross-technology communication in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2974–2987, 2017.
[3] W. Jiang, S. M. Kim, Z. Li, and T. He, "Achieving receiver-side cross-technology communication with cross-decoding," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 639–652.
[4] W. Wang, S. He, L. Sun, T. Jiang, and Q. Zhang, "Cross-technology communications for heterogeneous iot devices through artificial doppler shifts," *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 796–806, 2019.
[5] Z. Li and T. He, "WEBee: Physical-Layer Cross-Technology Communication via Emulation," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom'17)*, 2017, p. 2–14.
[6] Z. Yin, Z. Li, S. M. Kim, and T. He, "Explicit Channel Coordination via Cross-Technology Communication," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'18)*, 2018, p. 178–190.
[7] X. Guo, Y. He, X. Zheng, Z. Yu, and Y. Liu, "LEGO-Fi: Transmitter-Transparent CTC with Cross-Demapping," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 2125–2133.
[8] X. Zhang and K. G. Shin, "Enabling Coexistence of Heterogeneous Wireless Systems: Case for ZigBee and WiFi," in *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'11)*, 2011.

[9] C. A. Boano, T. Voigt, C. Noda, K. Römer, and M. Zúñiga, "JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2011, pp. 175–186.
[10] J. Ock, J. Paek, and S. Bahk, "QBT: Queue-size based Busy Tones for Protecting Multihop Low-power Networks," in *IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2019, pp. 389–397.
[11] "IEEE 802.19 Wireless Coexistence Working Group," accessed: 2020-11-10. [Online]. Available: http://www.ieee802.org/19/
[12] X. Zhang and K. G. Shin, "Enabling Coexistence of Heterogeneous Wireless Systems: Case for ZigBee and WiFi," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2011.
[13] J. Ock, H. Kim, H.-S. Kim, J. Paek, , and S. Bahk, "Low-power Wireless with Denseness: The Case of an Electronic Shelf Labeling System - Design and Experience," *IEEE Access*, vol. 7, no. 1, pp. 163 887–163 897, Dec 2019.
[14] "IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012) IEEE Standard for Information technology - Telecommunications and information exchange between systems Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," pp. 1–3534, Dec 2016.
[15] "open-ath9k-htc-firmware for Qualcomm Atheros AR7010 and AR9271 USB 802.11n NICs," accessed: 2020-11-10. [Online]. Available: https://github.com/qca/open-ath9k-htc-firmware
[16] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: a media access protocol for wireless LAN's," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 212–225, 1994.
[17] P. R. Paul Vrancken, CoSiNe, "IEEE 802.11 Medium Access Control (MAC) - Clear Channel Assessment."
[18] "Atheros ath9k Device Driver," accessed: 2020-11-10. [Online]. Available: https://wireless.wiki.kernel.org/en/users/drivers/ath9k
[19] A. Hithnawi, H. Shafagh, and S. Duquennoy, "Understanding the impact of cross technology interference on IEEE 802.15.4," in *Proceedings of the 9th ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, 2014, pp. 49–56.
[20] "Texas Instruments, CC2650 LAUNCH PAD," accessed: 2020-11-10. [Online]. Available: http://www.ti.com/tool/LAUNCHXL-CC2650
[21] "NXP, FRDM-KW36, Freedom Development Kit for Kinetis," accessed: 2020-11-10. [Online]. Available: https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/frdm-kw36-freedom-development-kit-for-kinetis-kw36-35-34-mcus:FRDM-KW36
[22] "Nordic nRF52840," accessed: 2020-11-10. [Online]. Available: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840
[23] "Atheros AR9380," accessed: 2020-11-10. [Online]. Available: http://static6.arrow.com/aropdfconversion/bb617b8c7c79a7311d45c8a20f4672e90256ea96/ar9380-al1a.pdf
[24] "Jouni Malinen's hostapd," accessed: 2020-11-10. [Online]. Available: http://w1.fi/hostapd/
[25] J. Heo, J. Kim, J. Paek, and S. Bahk, "Mitigating stealthy jamming attacks in low-power and lossy wireless networks," *Journal of Communications and Networks*, vol. 20, no. 2, pp. 219–230, 2018.
[26] "TI Smart RF Studio," accessed: 2020-11-10. [Online]. Available: https://www.ti.com/tool/SMARTRFTM-STUDIO
[27] "WireShark," accessed: 2020-11-10. [Online]. Available: https://www.wireshark.org/
[28] A. D. Wood, J. A. Stankovic, and G. Zhou, "DEEJAM: Defeating Energy-Efficient Jamming in IEEE 802.15.4-based Wireless Networks," in *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2007, pp. 60–69.
[29] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05)*, 2005, p. 46–57.
[30] J. Heo, Y. Yoo, J. Suh, W. Park, J. Paek, and S. Bahk, "FMS-AMS: Secure Proximity-based Authentication for Wireless Access in Internet of Things," *Journal of Communications and Networks*, vol. 22, no. 4, pp. 338–347, April 2020.
[31] T. Han, B. Han, L. Zhang, X. Zhang, and D. Yang, "Coexistence study for WiFi and ZigBee under smart home scenarios," in *IEEE International Conference on Network Infrastructure and Digital Content*, 2012, pp. 669–674.
[32] J. Zhu, A. Waltho, X. Yang, and X. Guo, "Multi-radio coexistence: Challenges and opportunities," in *IEEE International Conference on Computer Communications and Networks*, 2007, pp. 358–364.
[33] J. Wojtiuk, "Bluetooth and WiFi integration: Solving co-existence challenges," *R.F.Design*, vol. 27, no. 10, 10 2004.
[34] W. Sun, J. Koo, S. Byeon, W. Park, S. Lim, D. Ban, and S. Choi, "BlueCoDE: Bluetooth coordination in dense environment for better coexistence," in *IEEE International Conference on Network Protocols (ICNP)*, 2017, pp. 1–10.
[35] M. S. Gokturk and G. Ferazoglu, "Adjacent channel interference aware channel selection for wireless local area networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2014, pp. 2922–2927.
[36] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *Proceedings of the 6th ACM International conference on Information Processing in Sensor Networks (IPSN)*, 2007, pp. 21–30.

Seongmin Kim is currently an undergraduate student at the School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea. He is also a research assistant at the Networked Systems Laboratory (NSL) led by Dr. Jeongyeup Paek.

Jeongyeup Paek received his B.S. degree from Seoul National University in 2003 and his M.S. degree from University of Southern California in 2005, both in Electrical Engineering. He then received his Ph.D. degree in Computer Science from the University of Southern California (USC) in 2010. He worked at Deutsche Telekom Inc. R&D Labs USA as a research intern in 2010, and then joined Cisco Systems Inc. in 2011 where he was a Technical Leader in the Internet of Things Group (IoTG), Connected Energy Networks Business Unit (CENBU, formerly the Smart Grid BU). In 2014, he was with the Hongik University, Department of Computer Information Communication as an assistant professor. Jeongyeup Paek is currently an associate professor at Chung-Ang University, School of Computer Science and Engineering, Seoul, Republic of Korea since 2015. He is an IEEE senior member and an ACM member.