Transparent Cross-technology Communication over Data Traffic

Wenchao Jiang†, Zhimeng Yin†, Song Mim Kim‡, Tian He†
†Department of Computer Science and Engineering, University of Minnesota;
‡ Department of Computer Science, George Mason University
{jiang832, yinxx283}@umn.edu, song@gmu.edu, tianhe@cs.umn.edu

***Abstract*—Cross-technology communication (CTC) techniques are introduced in recent literatures to explore the opportunities of collaboration between heterogeneous wireless technologies, such as WiFi and ZigBee. Their applications include context-aware services and global channel coordination. However, state-of-the-art CTC schemes either suffer from channel inefficiency, low throughput, or disruption to existing networks. This paper presents the CTC via data packets (DCTC), which takes advantage of abundant existing data packets to construct recognizable energy patterns. DCTC features (i) a significant enhancement in CTC throughput while (ii) keeping transparent to upper layer protocols and applications. Our design also features advanced functions including multiplexing to support concurrent transmissions of multiple DCTC senders and adaptive rate control according to the traffic volume. Testbed implementations across WiFi and ZigBee platforms demonstrate reliable bidirectional communication of over 95% in accuracy while achieving throughput $2.3\times$ of the state of the art. Meanwhile, experiment results show that DCTC has little and bounded impact on the delay and throughput of original data traffic.**

I. Introduction

Wireless communication and mobile computing have brought great convenience to daily life over the last decades. They have also inspired the development of many wireless technologies, offering benefits in different aspects. Many of them, including WiFi and ZigBee, coexist at the 2.4GHz ISM band, where they compete for the channel and interfere with each other [11], [15], [20]. The competition for channel access causes inefficiency and fairness problems, especially between different wireless technologies. That is because many network coordination protocols, such as TDMA and RTS/CTS, can not be applied to heterogeneous wireless technologies.

To tackle this issue, several cross-technology communication (CTC) techniques are introduced in recent literatures [6], [21], [12], to provide direct communication across technologies. As a result, network coordination protocols, such as TDMA and RTS/CTS, can be extended to be globally applied across wireless technologies. It is notable that CTC not only alleviates the issue of interference, but also serves as a fundamental building block for collaborative applications via cross-technology cooperation. For example, CTC brings cost-efficient smart home by enabling WiFi-equipped devices (e.g., smart phones) to directly interact with ZigBee-embedded smart appliances. Thus we save the expense for a dedicated controller or a costly gateway hardware. In addition, CTC also provides energy-efficient mobile computing by embedding a smart phone with a low-power ZigBee radio. The ZigBee radio wakes up power-hungry WiFi interface only when it detects a WiFi AP of interest.

The principal behind CTC is that wireless devices, despite of the technologies they use, are available and mandatory to sense the energy of the channel (i.e., RSSI) in order to access the channel and communicate with each other. Although devices using different technologies can not decode each other's information due to incompatible physical layer, they are all able to identify the existence of each other through channel energy sensing.

CTC is achieved by generating interference patterns universally interpretable regardless of underlying PHY layers. Up-to-date CTC techniques either generate such patterns by injecting dummy packets of certain size [6], [21], constructing customized preambles [19], or shifting mandatory beacon packets [12]. Despite of the motivating results, they commonly suffer from channel inefficiency, low throughput, or disruption to existing networks, leaving significant room for further improvement. In this paper we introduce CTC via data packets (DCTC), where the key concept is utilizing existing data packets which are the dominating portion of wireless network traffic. By slightly perturbing the transmission timings of enqueued data packets, we construct specific energy patterns that can be demodulated by heterogeneous wireless technologies. DCTC overcomes the limitations of state-of-the-art techniques. More specifically, DCTC (i) significantly enhances CTC throughput, while (ii) keeping the technology transparent to upper layer protocols and applications.

Utilizing existing network data packets for CTC is not trivial. First, the dynamic nature of the volume, both bursty and sparse, affect the opportunity for CTC modulation. Moreover, delay sensitivity of the applications, especially real-time applications, such as audio/video streaming applications and online games, require the perturbation delay to be small enough to comply with applications' delay requirements.

To the best of our knowledge, this is the first work that achieves CTC with existing data packets, naturally flowing through any network entity. Its transparency is achieved by (i) specific CTC modulation independent of the arrival of future data packets, (ii) packet perturbation with bounded delay even without the knowledge of incoming packet distribution, and (iii) dynamic configuration accommodating CTC throughput to the amount of available data packets. We have imple-

978-1-5090-5336-0/17/$31.00 ©2017 IEEE

mented our design on a wireless open-access research platform (WARP[5]) and MICAz[2], compliant to the WiFi (i.e., 802.11) and ZigBee (i.e., 802.15.4) standards respectively. Our experiment results demonstrate the average perturbation delay of only $0.5ms$ while achieving 95% accuracy in transmitting DCTC symbols from a WiFi AP to a ZigBee node, showcasing the reliability of DCTC as well as its transparency to upper layer applications. We have also implemented our design from a ZigBee node to a WiFi device to achieve bidirectional communication.

Our contributions in this paper are as follows.

- We propose DCTC, a novel CTC framework that explores existing data traffic. DCTC is designed to take advantage of the large volume of data traffic in the wild, and achieves significantly higher throughput compared with the state-of-the-art works, without incurring traffic overhead.
- Modulation in DCTC is done by perturbing the transmission timings of data packets, where the amount of perturbation delay incurred is shown to be small and bounded. This ensures transparency of our design to upper layer protocols and applications.
- DCTC requires no hardware modification and is compatible with off-the-shelf devices. Implementation of DCTC on WiFi and ZigBee platforms shows that DCTC offers reliable symbol delivery with an average delay of less than $0.5ms$. In addition, DCTC has little impact on the performance of popular network applications, such as FTP, video streaming, and audio streaming.

The rest of this paper is organized as follows. Our motivation is presented in Section II. In Section III, we introduce the basic design of DCTC. In Section IV, we introduce advanced design components to further improve the performance and practicality. In Section V we analyze the theoretical performance. In Section VI, we conduct experimental evaluation. In Section VII, we review state-of-the-art CTC works in relation to our design. Finally, we conclude our paper in Section VIII.

II. Motivation

This section begins with the insight on abundant CTC opportunities under today's typical environment, followed by the challenge we face in perturbing the timing of data packets.

A. Opportunity for CTC

State-of-the-art literatures achieve CTC either by injecting dummy packets [6], [21] or utilizing mandatory beacons [12]. The former approach introduces additional traffic that is disruptive to existing networks, with its significant spectrum occupancy. Although the latter is free from the issue as it utilizes existing beacons, it suffers from low CTC throughput due to the confined number of beacons and hence limited CTC opportunities. Therefore, from the two methods, we note that the best of both worlds can be reached by utilizing existing traffic with a sufficient volume.

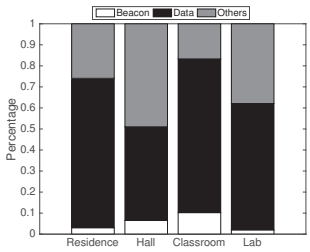


Figure 1: Distribution of network packets in four everyday life scenarios: residence, hall, classroom, and lab.

Fig.1 demonstrates the compositions of WiFi traffic experimentally observed in four typical everyday environments, including residence, hall, classroom, and lab. The figure shows that, unlike the small volume of beacons, the data packets take up the majority portion of the traffic in all four scenarios to reach over 40%. It is more than $7\times$ of the amount of beacons. This indicates utilizing existing data traffic not only enables establishing CTC with high spectrum efficiency (i.e., without dummy packets), but also provides sufficient opportunities to reach high CTC throughput.

B. Challenge of DCTC

Although the huge volume of data packets provide us with huge opportunity to establish CTC, perturbation of data packets may degrade the performance of upper layer applications, such as the drop in throughput.

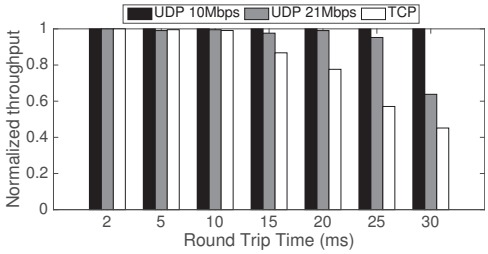


Figure 2: The impact of packet delay on throughput, normalized to the highest throughput of each experiment.

To study how severe packet delay (measured by round trip time, RTT) will affect the throughput of upper layer applications. We test the normalized throughput (normalized to the highest throughput) of both UDP and TCP traffic with different bandwidth using Iperf[1]. The UDP bandwidth is chosen from 10Mbps to 21Mbps (saturation bandwidth), while TCP throughput is tested by setting a large enough TCP window size, due to the fact that TCP bandwidth can not be directly set in Iperf. In Fig. 2, we find that, the throughput of TCP traffic decreases dramatically when RTT is higher than 10ms. That is because (i) TCP flow control mechanism restricts TCP throughput, making it inversely proportional to RTT in theory, and (ii) ACK timeout may be triggered when data packets are perturbed for a long time. Even though UDP traffic doesn't have flow control and ACK, its throughput starts to degrade when RTT is 20ms in saturation case. That is due to transmission queue overflow.

From the study, we find that the throughput of both TCP and UDP traffic, especially TCP traffic, will be affected by packet delay. Thus how to minimize packet perturbation time, so that to minimize the increment of packet delay, is the main challenge in establishing CTC via data packets.

III. System Design

In this section, we first introduce the objective of our design, followed by an overview of our system design, and finally the details of DCTC modulation/demodulation design.

A. Design Objective

Due to the critical impact of packet delay on the application throughput, our design aims at establishing CTC via data packets (known as DCTC) that enhances CTC throughput while keeping transparent to upper layer network applications. More specifically, our design features (i) packet perturbation with bounded delay even without the knowledge of incoming packet distribution, and (ii) dynamic DCTC throughput fitting incoming packet rate.

B. Design Overview

DCTC is a bidirectional communication system between two heterogeneous wireless communication technologies, WiFi and ZigBee. Without loss of generality, we use the communication from a WiFi sender to a ZigBee receiver to illustrate our main design. The opposite communication is based on similar principle. Fig 3 describes DCTC's design architecture in both the WiFi sender side and ZigBee receiver side, where the grey boxes are existing protocols/applications, and the white boxes are DCTC's layers.

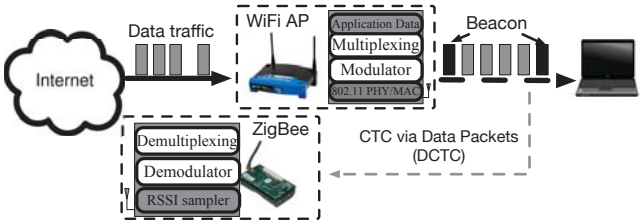


Figure 3: An overview of DCTC system design.

At the sender side, the top layer lies the existing applications. DCTC's opportunity comes from the data traffic of internet applications, without any modification to existing applications. Then, DCTC has its multiplexing layer, which supports concurrent transmission of multiple senders. Next, DCTC has its modulation layer, which shifts transmission timing of the packets in the transmission buffer with reference to WiFi beacons to construct DCTC symbols. Finally, since DCTC is built upon the existing 802.11 MAC/PHY layers, it follows the existing protocols, such as channel sensing, packet ACK and retransmission. These features enable DCTC to be transparent to the existing physical layer protocols.

At the receiver side, DCTC utilizes the existing sampling ability of the ZigBee nodes to sense the energy (RSSI) of the channel. Although the ZigBee receiver can not demodulate WiFi signal directly due to the incompatible physical layers, DCTC demodulator can demodulate DCTC symbols through RSSI sampling results. In addition, the DCTC receiver relies on its de-multiplexing layer to recognize different senders.

C. Modulation

In CTC scenario, WiFi beacon has been proved to be reliably demodulated by heterogeneous radio through its energy pattern [12]. However, employing that idea directly to data traffic is not practical. That is because regulating dynamic data traffic to be periodic may cause huge increase in packet delay or severe buffer control problem. We propose the modulation of DCTC which establishes detectable energy patterns while introducing little and delay bounded perturbation to existing data traffic.

Without loss of generality, we use the process of modulating 1-bit information at a single WiFi AP as an example. We regard the packets transmitted by other wireless transmitters as background noise.

A WiFi AP will broadcast WiFi beacons periodically due to the 802.11 protocol. Within a beacon period, we set some *critical time points* with period Δ, which is a user defined parameter shared with both sides in advance. Critical time points have alternating labels to indicate bit "0" and bit "1" respectively. To modulate a data packet with minimum delay, we perturb it to cover the nearest critical time point.

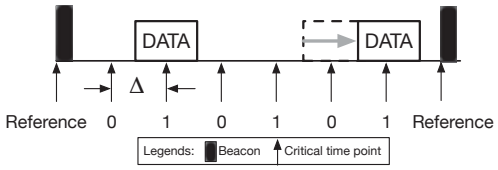


Figure 4: Illustration of DCTC symbol modulation

In Fig. 4, we illustrate how to modulate bit "1". To do that, we perturb all the data packets to cover their closest time point "1". The first packet has already covered critical time point "1", so we don't move it. The second one, however, doesn't cover critical time point "1", so we perturb it to cover the next time point "1". Note we do not regulate the random incoming data packets to be periodic, i.e., to cover all time point "1"s, but to slightly adjust the energy pattern according to its original incoming pattern. In this example, there are three critical time point "1"s, but data packets only cover two of them. To extend the design to multi-bit case, we can either cut a beacon period into small segments or a more advanced method discussed in Section IV-A.

D. Demodulation

To demodulate DCTC symbols, the ZigBee receiver needs to first synchronizes with the WiFi sender by detecting beacons of the transmitter from all kinds of noise packets in the air. Then it extracts DCTC symbols with reference to the beacons of the transmitter.

• **Step 1: Synchronization**: An 802.15.4-compliant ZigBee node recognizes WiFi beacons under the presence of channel noise by detecting their periodic energy pattern. The ZigBee node continuously senses the energy of the channel, and records the RSSI values from the RF chip in its flash. Then it quantizes the captured RSSI values to be high and low to indicate busy and idle channels (denoted as dark and white boxes in Fig. 5). The threshold is set to be -75dBm following the CCA (clear channel assessment) threshold for the 802.15.4 standard [10]. The sampling rate of the ZigBee node is $7.8KHz$, which makes a measurement spanning of 128μs to avoid time gaps in sampling adjacent data packets.

Next, the ZigBee node applies *folding* to the quantized RSSI sequence, which is a signal processing technique to extract periodic signal from noise [16]. The process of folding is as follows. The received RSSI sample sequence is divided by the beacon interval, which is a preset value for WiFi APs. All the samples are stacked together, and a column-wise sum is applied to calculate the number of samples with high RSSI values in each column, which is denoted as *fold sum*. Since WiFi beacons are periodic, they will align column-wise and preserve a higher fold sum than other signals.

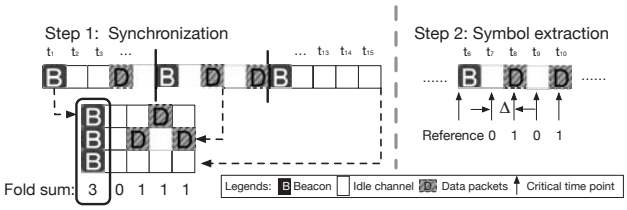
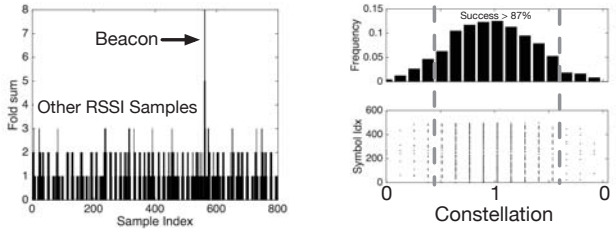


Figure 5: Illustration of DCTC symbol demodulation

In Fig. 5, we illustrate how to detect beacon positions from a quantized RSSI sequence. We cut the quantized RSSI sequence into 3 subsequences of length $T = 5$, where T is the period of beacons. Then we stack them together and calculate the fold sum. We find that the fold sum of the first column is the highest among all the columns. Then we detect the beacon position at the first column.



(a) Synchronization (b) Symbol extraction

Figure 6: DCTC symbol demodulation in practice

In practice, as illustrated in Fig. 6a, the beacon period is set to be $102.4ms$, which means there are 800 ZigBee RSSI samples in one beacon period. After folding, we find that the fold sum at index 572 exceeds the fold sums of the others, and thus we demodulate the beacon position.

• **Step 2: DCTC Symbols Extraction**: With reference to the WiFi beacon, we check quantized RSSI samples every Δ seconds at critical time points. Just as in the modulation process, critical time points have alternating labels "0" or "1". If the channel RSSI is high at a certain critical time point, we regard the successful reception of an according DCTC symbol "0" or "1". Then we accumulate the DCTC symbols we received within a beacon period. The dominating symbol in the whole beacon period is interpreted as the symbol conveyed.

In Fig. 6b, we illustrate how to extract DCTC symbol in practice. We depict the RSSI samples with high RSSI values in constellation figure. We find that over 87% of the high RSSI values are near critical time point "1", with the highest frequency appear at exactly the time point "1". Since the frequency of time point "1" is larger than that of time point "0", the beacon period is demodulated as symbol "1".

E. Design Features

The main features of DCTC design is that although we perturb data packets to embed DCTC symbols, its modulation/demodulation method has limited impact on the original network traffic and protocol shown as follows.

• **Bounded Traffic Delay**: The perturbation operation guarantees a bounded packet delay. Since we perturb data packets to the nearest critical time point, each data packet will be perturbed at most 2Δ to cover the correct time point. Considering that the incoming data packets arrive in random, so the amount of perturbation is uniformly distributed in the interval $[0, 2\Delta)$. As a result, the average amount of perturbation is Δ.

• **Limited Impact on Throughput**: The perturbation operation has limited impact on the packet throughput. Although the packets are perturbed to cover only one of the two kinds of critical time points, the traffic throughput won't degrade much. First, in most time, packet intervals are larger than the critical time point interval, i.e., data packets are perturbed to cover different critical time points. In that case, perturbation of one packet won't affect the timing of the next one. Second, even though the incoming traffic is dense, we still have room to flush out surplus data packets. We can flush out data packets between two critical time points, because the receiver only detects at critical time points. Data packets flushed within two critical time points won't be demodulated as any symbol, thus won't affect symbol demodulation. In this way, the perturbation has little impact on traffic throughput.

• **Compatibility with CSMA**: The perturbation operation is compatible with CSMA random backoff. In CSMA, a random backoff timer is set when there are data packets to transmit. The timer will count down whenever the channel is sensed to be idle. The actual time backed off depends on the environment and hard to estimate, which affects the accuracy of modulation. In our design, the modulation of data packets is after CSMA's random backoff, i.e., we add small additional time delay after normal CSMA backoff. Thus we can control the timing of packets accurately while maintaining compatible with CSMA protocol.

IV. Advanced design

In this section, we propose several advanced designs that help improve the throughput of DCTC while still maintaining transparent to upper layer applications. One is a multi-bit embedding technique, the other is a multiplexing technique.

A. Multi-bit Embedding

Although we can modulate n bits within a beacon period by separating a beacon period into n small segments. Its performance degrades at dynamic packet rate. Sometimes there are insufficient data packets within one segment to modulate a symbol while surplus data packets within another. To address that issue, the sender should be able to dynamically decide when to modulate a bit depending on the incoming packet rate. To do that, instead of using the whole beacon period to modulate one bit, the sender can repeat a symbol a certain number of times, above some threshold, then turn to the next bit. The threshold is set according to the channel noise level. Detailed discussion is in Section V-A.

One issue that needs to be solved is that, without using beacon as boundary, it becomes hard to distinguish neighboring symbols, especially when we repeat a symbol multiple times. For example, when the receiver receives four bit "1"s, we can not tell whether the sender is transmitting four repeats of a single bit "1" or two bits that are the same, i.e., two repeats of the first symbol "1" and two repeats of the second symbol "1".

To address this issue, we adopt a coding method similar to the hybrid ternary code [8], which transfers any 0-1 sequence to a new sequence where no consecutive bits are the same. To do that, we introduce a new bit "R", which means "Redo". To encode any 0-1 sequence, we simply scan from left to right, whenever we find a bit that is the same as its previous one, we change it to be "R". To decode the code, we just interpret "R" as its previous symbol. For example, a sequence "111100" will be coded as "1R1R0R", where no consecutive bits are the same. The only change to the basic modulation is that we now have three types of critical time points, "0", "1", and "R", instead of two.

To demodulate the symbols, the receiver scans at critical time points and uses three counters to count the number of each type of bits respectively. Whenever it has detected a threshold number of certain bit (e.g., five bit "1"s), it demodulates a bit (e.g., bit "1"). Then the receiver clears the three counters and disables the counter for the bit already demodulated (e.g. counter of bit "1"). That is because we know no consecutive bits are the same. We will enable the bit counter after we have demodulated another bit.

With the multi-bit embedding technique, we can achieve multi-bit embedding adapting to the dynamic data traffic at the sender side, while still being able to demodulate symbols at receiver side.

B. Multiplexing

Here we propose how to achieve multiplexing among multiple DCTC senders. That means multiple DCTC senders can transmit at the same time while each of their receivers can still demodulate DCTC symbols correctly. In this way, the accumulated DCTC throughput can be improved.

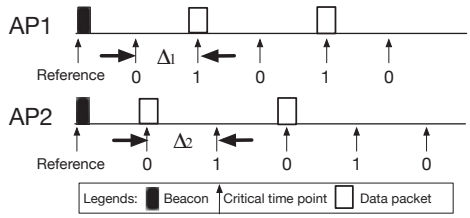


Figure 7: Illustration of DCTC multiplexing.

We achieve multiplexing by carefully choosing the critical time point intervals assigned to different APs, so that the critical time points from any two different WiFi APs will meet at most once within any beacon periods. As illustrated in Fig. 7, AP1 and AP2 have their critical time points with interval Δ_1 and Δ_2 respectively. They can modulate their DCTC symbol at their own critical time points without interfering with each other. To find such pairs of critical time point intervals, we have the following theorem.

Theorem IV.1. *Let CTP_1 be a sequence of critical time points with interval Δ_1 and CTP_2 be a sequence of critical time points with interval Δ_2. CTP_1 and CTP_2 will overlap at most once within beacon period T if*

$$LCM(\Delta_1, \Delta_2) \geq T, \tag{1}$$

where LCM is the the least common multiple.

Proof. We prove by contradiction. Assuming CTP_1 and CTP_2 overlap more than once with T. Without loss of generality, we say they meet at least at time t_1 and t_2, where $0 < t_2 - t_1 < T$. Then $t_2 - t_1$ is the common multiple of Δ_1 and Δ_2, and it is less than T, which conflicts with the fact that $LCM(\Delta_1, \Delta_2) \geq T$. □

Theorem. IV.1 guarantees that two critical time point sequences have minimum cross-talk interference between them. Thus we can achieve concurrent transmission of multiple DCTC senders.

V. Analytics

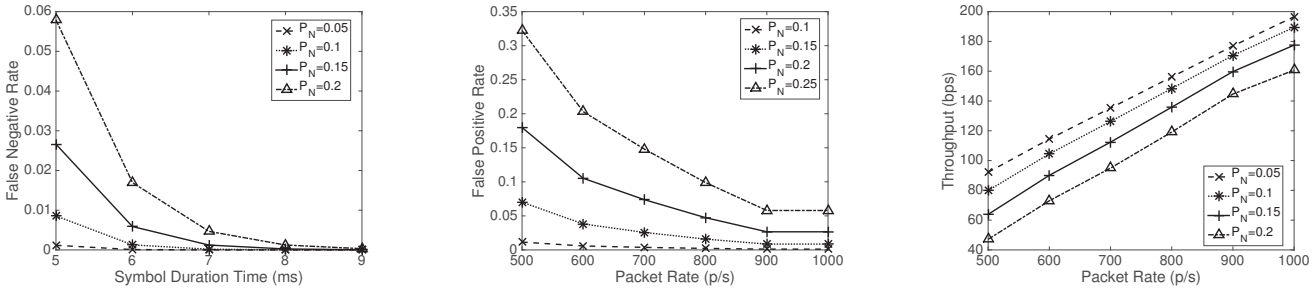
In this section, we analyze the symbol error rate as well as the throughput of DCTC.

P_N	Noise rate
N_s	Number of repeat symbols at the sender
N_r	Threshold number of repeat symbols at the receiver
R_s	Packet rate at the sender
Δ	Critical time point interval

Table I: Parameters

A. Symbol Error Rate

The wireless channel is shared by multiple wireless devices and is always crowded with wireless traffic. In CTC, packets from devices other than DCTC senders are regarded as noise. So the noise ratio P_N can also be regarded as the channel



(a) False negative error. (b) False positive error (c) Throughput

Figure 8: Simulation of DCTC's performance under noise

occupancy rate. The noise in the air brings two problems to the accuracy of DCTC: (i) it may occupy the correct time point at the sender side (but not heard by the receiver, due to hidden terminal issue) and prevent the sender to perturb data packets to the right time point, which we refer to as the *false negative problem*, or (ii) it may occupy the wrong time point at the receiver side, causing the false detection of a wrong time point, which we refer to as the *false positive problem*.

To address both problems, we repeat one symbol multiple times, denoted as N_s, at the sender. If the receiver can detect certain number of the repeats, denoted as N_r, it successfully demodulates a symbol. Thus the false negative rate P_{FN} can be formulated as

$$P_{FN} = \sum_{k=0}^{N_r-1} \binom{N_s}{k} (1-P_N)^k P_N^{N_s-k}, \tag{2}$$

which means the probability that we can detect more than or equal to N_r correct time points at the receiver side.

The false positive rate P_{FP} is affected by the packet rate at the sender R_s. The intuition is that, when packet rate is low, it takes more time for the sender to send one symbol. So the receiver has more chance to be affected by the false negative issue during one symbol time. The average time to send one symbol can be derived as $\frac{N_s}{R_s}$, which means the time DCTC sender takes to transmit N_s packets. During that time, the number of critical time point sampled at the receiver side is $\frac{N_s}{R_s \times \Delta}$. Among them, half number of the critical time points are the wrong ones. So the total number of wrong critical time points at the receiver side is $N_w = \frac{N_s}{2R_s \times \Delta}$. False positive problem happens when noise packets occupy more than N_r wrong critical time points, which can be formulated as

$$P_{FP} = 1 - \sum_{k=0}^{N_r-1} \binom{N_w}{k} P_N^k (1-P_N)^{N_w-k}. \tag{3}$$

With the false negative and false positive rate, we can derive the symbol error rate (SER) as

$$SER = (1-P_{FN}) \times (1-P_{FP}). \tag{4}$$

In Fig. 8a we depicts the false negative rate with increasing symbol duration, which is $\frac{N_s}{R}$ (R is fixed to $1000p/s$). We find that when we increase the number of symbols repeated at the sender, the false negative rate decreases dramatically. It decreases to under 1% when we repeat a symbol 7 times at the sender. In Fig. 8b, we study the false positive rate with increasing packet rate while fixing N_s to be 5. We find that false positive rate decreases under 10% when packet rate is larger than $800p/s$. That is because when packet rate increases, N_w decreases, so the receiver will have less chance to be affected by noise.

B. Throughput

We need to achieve maximum DCTC throughput, which depends on both the demodulation accuracy and DCTC bit rate. To modulate a symbol more data packets (higher N_s), we can guarantee higher symbol accuracy (P_s), but decrease the DCTC bit rate ($\frac{R}{N_s}$). To maximize DCTC throughput, we formulate it to be an optimization problem,

$$max_{N_s,N_r} SER \times \frac{R}{N_s}. \tag{5}$$

Solving the optimization problem is out of the scope of this paper. We can use some optimization toolboxes to find the solution.

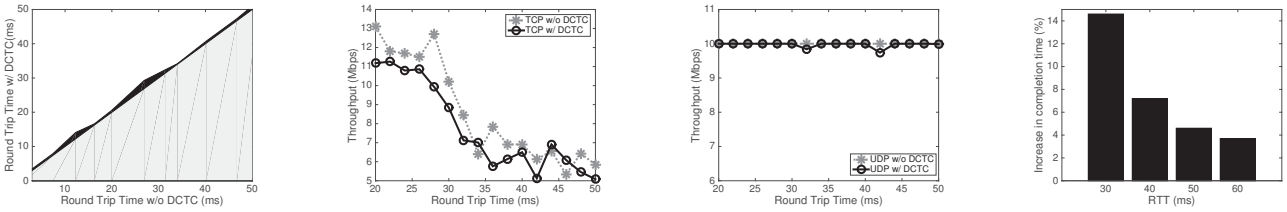
In Fig. 8c, we depict the DCTC throughput with increasing packet rate. We find that when the packet rate increases, DCTC throughput also increases. When the error rate is 20%, the DCTC throughput increases from $47bps$ to $160bps$. The throughput increases faster when packet rate is large because the decrease in SER.

VI. Evaluation

In this section, we evaluate DCTC's performance in throughput, transparency, error rate under noise, multiplexing and receiver overhead.

A. Setting

We have implemented bidirectional DCTC communication between WiFi and ZigBee nodes through data packets from applications. Fig. 10 shows the experiment setting of our design. We use a laptop as a data packet generator through network applications. The laptop is connected to a WiFi AP equipped with DCTC design. Although we have implemented DCTC on both off-the-shelf WiFi chipset (laptop) and WARP [5], which is a wireless research platform implemented with WiFi PHY/MAC layers, the following evaluation results are based on WARP platform, for its convenience in changing



(a) RTT w/ and w/o DCTC (b) TCP throughput w/ and w/o DCTC (c) UDP throughput w/ and w/o DCTC (d) Increment % of FTP file transmission time.

Figure 9: DCTC's transparency to upper layer applications

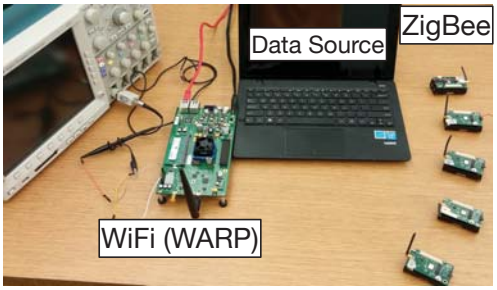


Figure 10: Experiment setting.

parameter setting. We also have 5 ZigBee-compliant MICAz nodes, which can sample the channel and capture RSSI values at $7.8KHz$, and records them within its $512KB$ on-board flash. WiFi AP works on WiFi channel 1, and ZigBee works on ZigBee channel $11-14$, overlapped with WiFi channel 1.

B. Throughput

We here compare the throughput of DCTC with the state of the art.

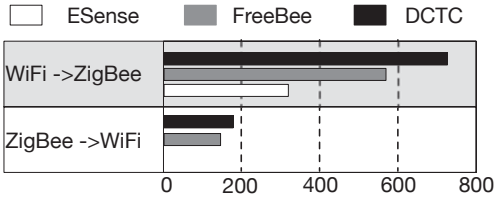


Figure 11: Throughput of bidirectional communication between WiFi and ZigBee.

• **WiFi→ZigBee**: Fig. 11 compares the achievable throughput of DCTC from WiFi to ZigBee with two state-of-the-art works, ESense and FreeBee. According to the author, with multiple ESense senders working in parallel, ESense achieves a bit rate of $1.63Kbps$. Since five consecutive packets are need in ESense to guarantee reliable transmission of a symbol, the actual throughput of ESense is $326bps$. Under the same setting, asynchronous FreeBee achieves $560bps$ in throughput via concurrent transmission of hundreds of FreeBee senders. For DCTC, the maximum throughput is restricted by the sampling interval at the receiver side. The ZigBee receiver samples one RSSI value every $128\mu s$, and we need one time point for each of the two symbols. In addition, we can guarantee reliable ($> 95\%$) symbol detection by repeating a symbol 5 times, which results in a throughput of $760bps$ for DCTC, which is $1.3\times$ of FreeBee, and $2.3\times$ of ESense.

• **ZigBee→WiFi**: Here we test the performance of DCTC from ZigBee network to WiFi. FreeBee achieves throughput of $146bps$ from ZigBee to WiFi. In DCTC, since WiFi can sample the channel much faster, the bottleneck is the transmission rate of ZigBee node. The maximum transmission rate of MICAz is $1000p/s$, also we can guarantee over 95% accuracy by demodulating 5 data packets. So DCTC's throughput from ZigBee to WiFi is $190bps$, which is $1.3\times$ of FreeBee.

C. Transparency

In this part, we evaluate DCTC's transparency in packet round trip time and throughput. In addition, we will also use some popular network applications, such as FTP, audio streaming, and video streaming to evaluate DCTC's impact on real network applications.

• **End-to-End Delay** Here we study DCTC's impact on network end-to-end delay. In the experiment, the critical time point interval is set to be $512\mu s$, and we use the Linux command "Ping" to test the RTT, which is an indicator of end-to-end delay, w/ and w/o perturbation. In addition, we use DummyNet to emulate different network end-to-end delay.

In Fig. 9a, the black area is the additional delay introduced by DCTC. We find that the RTT w/ DCTC is within $2ms$ of RTT w/o DCTC. Considering that it is natural to have $1ms$ fluctuation in the normal RTT due to dynamic network situation, results in Fig. 9a can show that our design guarantees a bounded perturbation delay.

• **End-to-End Throughput** Here we compare how our design will affect the throughput of normal WiFi traffic, both TCP and UDP traffic. We use Iperf[1] to generate TCP and UDP traffic. TCP is given a large enough window size and UDP is set a constant bandwidth of $10Mbps$.

From Fig. 9b, we find that when RTT is 20ms, the throughput of TCP traffic degrades about 15.3%. As the RTT increases, the throughput difference between TCP w/ and w/o DCTC becomes smaller. That is because DCTC has a bounded delay to original TCP traffic, and the throughput of TCP traffic is almost inverse proportional to RTT. As a result, when RTT is small, throughput degradation is large and it decreases as RTT increases. The impact of DCTC on TCP throughput can be omitted when RTT is larger than or equal to 40ms. From Fig. 9c, we find that the throughput of UDP traffic keeps stable as the round trip time increases. We find that the UDP throughput

w/ and w/o DCTC are almost the same, which verifies the transparency of DCTC to the network traffic.

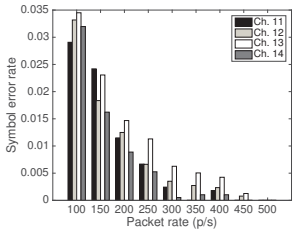
• **Impact on Network Applications** Here we use 3 applications: FTP, Audio streaming, and Video streaming. The performance of the applications are as follows:

(i) FTP traffic: We construct FTP traffic with the tool vsftpd [4] on Linux platform. We transfer a large file of size 55MB through an AP equipped with DCTC layer. Without DCTC, it takes about $48s$ to finish transmitting the file ($RTT = 30ms$). With DCTC, the increment of ftp completion time is shown in Fig. 9d. The increment of ftp completion time decreases as the RTT increases. When network RTT is $30ms$, the increment in completion time is 14.6%, and the increment decreases to 3.7% when network RTT is $60ms$. It shows that the impact on FTP is small and won't affect normal functionality, especially for long-distance file transmission. Since FTP is upon TCP protocol, the result is consistent with DCTC's impact on TCP throughput.

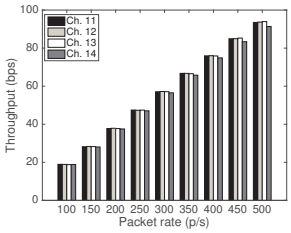
(ii) Audio & Video traffic: We set up an audio & video streaming server using the tool PLEX [3] and connect it to an AP equipped with DCTC layer. Then we connect and play the audio & video stream in web browser from a remote laptop. We find that even in a network with high RTT ($60ms$) we can play a MP3 audio stream and a high quality video stream ($>$ 720p) smoothly. The reason is that DCTC has a limited impact on the TCP and UDP throughput. So applications built upon them can work well as usual.

D. SER

We here evaluate the SER for ZigBee nodes to demodulate DCTC symbol. In the experiment, one laptop is connected to WiFi AP (WARP) and generates UDP packets. The WiFi AP establishes DCTC symbols using the data packets flowing through it at WiFi. The WiFi AP works at the 20MHz WiFi channel 1, which covers four 2MHz ZigBee channel from 11 to 14. We transmit more than 3,000 DCTC symbols with the packet rate of laptop varies from $100p/s$ to $500p/s$ and the DCTC throughput fixed to be $10bps$.



(a) Demodulation error rate with different packet rate.



(b) DCTC throughput with increasing packet rate.

Figure 12: DCTC's SER and throughput with increasing packet rate

In Fig. 12a, we find that the symbol error rate decreases as the packet rate increases. When packet rate is higher than $250p/s$, SER of all four ZigBee channels becomes lower than 1%. That is because with higher packet rate, we use more data packets to modulate one symbol, so that the symbol error rate is lower. We note that Fig. 12a shows that we can demodulate DCTC symbols with high accuracy at all four ZigBee channels, which provides the potential for cross-channel broadcast from WiFi to ZigBee through data traffic.

E. Impact of Traffic Volume

In Fig. 12b, we study the DCTC throughput as WiFi packet rate increases. We find that the DCTC throughput increases as WiFi packet rate increases. When packet rate is $500p/s$, DCTC can reach about $90bps$ in throughput. That means DCTC can adapt to the volume of network traffic and make full use of DCTC opportunity. In addition, the DCTC throughputs at different ZigBee channels are close. That is because a WiFi sender covers four ZigBee channels and can broadcast to ZigBee devices at these channels.

F. Multiplexing

In Fig. 13, we study the concurrent DCTC symbol transmission. We have five WiFi APs that are embedded with DCTC. The critical time point interval of these APs are set $10ms$, $11ms$, $13ms$, $15ms$, and $17ms$ respectively, which have minimum cross-talk interference between each two of them. We open these WiFi APs one after another and evaluate the DCTC throughput of a single AP as well as the aggregated throughput of multiple APs.

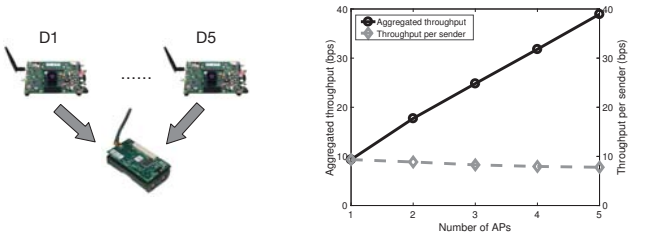


Figure 13: Concurrent DCTC symbol transmission with up to 5 APs (D1 - D5)

We find that when only one AP is on, its DCTC throughput can be as high as $9.3bps$. As more APs add in, the DCTC throughput of the AP slightly decreases. When there are 5 APs working together, the throughput of a single DCTC sender is about $7.7bps$, due to noise in the busy channel. However, the aggregated throughput of the five APs is almost $4.1\times$ the throughput of one AP.

G. Receiver Overhead

In this section we demonstrate the light-weight memory storage and computing cost for DCTC. So that it can work on off-the-shelf ZigBee nodes and won't disturb legacy network protocol.

• **Storage**: In DCTC, the most storage cost is in detecting the position of WiFi beacons. We need to store enough RSSI samples temporarily to detect the position of periodic WiFi beacon. We need to store 800 quantized RSSI samples in each beacon period, which occupies $800bits$. In addition, we need to fold 5 times to reliably demodulate beacon position, which

causes a total storage of $500Bytes$, less than 0.1% of MICAz's $512KByte$ memory.

• **Computation**: The steps of DCTC modulation can be divided into 4 parts: (i) sampling RSSI, (ii) calculating fold sum, (iii) locating beacon position, (iv) checking critical time points, and (v) demodulating DCTC symbol. Among them, (i) is automatically done whenever ZigBee nodes are open. After collecting enough RSSI samples (i.e., $\rho \times f \times T$), we need to do $\rho \times f \times T$ addition or comparison in (ii) and (iii). The cost of step (iv) and (v) are related to critical time point interval Δ, the number of symbols in one beacon period n, and number of concurrent APs k. The total cost of addition and comparison is $k \times \frac{f \times T}{\Delta \times n}$ per symbol. That means the computation cost increases linearly with the number of APs.

VII. Related Work

Network interference is a hot topic in networking. A lot of work in the literature deal with how to model the interference[13], [17] and avoid it[14], [18], [9]. CTC, however, takes advantage of such interference by conveying information across technologies via generating patterns embedded in the interference. The pattern is picked up at the receiver via RSS sensing capability, commonly available to various wireless technologies including WiFi and ZigBee.

In the literature, there are three main approaches to implement CTC: injecting dummy packet, generating customized signal, or exploring free channel. ESense [6] and Howies [21] belongs to the first category. They inject data packets of specified length and encode CTC symbols via packet lengths. Such approach is easy to implement and can achieve high throughput, but enforces overhead to today's saturated spectrum. GSense [19] belongs to the second category by prepending a customized preamble in front of legacy data packets, which carries CTC symbols. Such approach can achieve high throughput with little network overhead but requires a dedicated hardware and is not compatible with off-the-shelf devices. FreeBee[12] belongs to the third category, because it encodes CTC symbols in the timings of mandatory beacons without introducing dedicated packets. Moreover, it remains compliant to existing wireless standards for compatibility with off-the-shelf devices. However, the performance of the design is limited by the number of beacons. That is, as beacons are intermittent (e.g., every 102.4ms in WiFi) the opportunity for CTC is inherently limited, leading to a low throughput. A recent work [7] belong to the third category builds CTC from Bluetooth to WiFi through packet energy level. But they are specified to the channel hopping Bluetooth signal.

In contrast to previous works, DCTC has the following advantages: (i) utilization of existing data traffic to establish CTC, (ii) small and bounded impact on upper layer applications, (iii) higher throughput compared with the state-of-the-arts, and finally, (iv) compliance to existing protocols, indicating compatibility with off-the-shelf devices.

VIII. Conclusion

In this paper, we propose DCTC, a novel CTC system based on existing data traffic in the network. Different from other related studies in CTC, we focus on making full use of the large amount of data packets in the air to improve DCTC's throughput, while reducing its effect on original data traffic. Our approach not only generates detectable energy patterns, but also guarantees perturbation delay bound. We have implemented bidirectional DCTC communication between two heterogeneous wireless technologies, WiFi and ZigBee. Experiment results show that we can achieve reliable ($> 95\%$) bidirectional DCTC communication, while achieving $2.3\times$ throughput compared to the state of the art.

Acknowledgement This work was supported in part by US NSF Grant CNS-1525235 and CNS-1444021.

References

[1] Iperf. http://iperf.fr/.
[2] Micaz datasheet. http://www.memsic.com.
[3] Plex media server - your media on all your devices. https://www.plex.tv.
[4] vsftpd - secure, fast ftp server for unix-like systems. https://security.appspot.com/vsftpd.html.
[5] Wireless open-access research platform. http://warpproject.org/trac/.
[6] K. Chebrolu and A. Dhekne. Esense: communication through energy sensing. In *Proceedings of ACM MOBICOM*, pages 85–96. ACM, 2009.
[7] Z. Chi, Y. Li, H. Sun, Y. Yao, Z. Lu, and T. Zhu. B2w2: N-way concurrent communication for iot devices. In *Proceedings of ACM Sensys*, 2016.
[8] A. Glass, B. Ali, and E. Bastaki. Design and modeling of h-ternary line encoder for digital data transmission. In *Proceedings of IEEE ICII 2001*, volume 2, pages 503–507. IEEE, 2001.
[9] S. Gollakota, F. Adib, D. Katabi, and S. Seshan. Clearing the rf smog: making 802.11 n robust to cross-technology interference. In *Proceedings of ACM SIGCOMM*. ACM, 2011.
[10] J. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, B. Heile, et al. Ieee 802.15. 4: a developing standard for low-power low-cost wireless personal area networks. *network, IEEE*, 15(5):12–19, 2001.
[11] J. Huang, G. Xing, G. Zhou, and R. Zhou. Beyond co-existence: Exploiting wifi white space for zigbee performance assurance. In *Proceedings of IEEE ICNP*, pages 305–314. IEEE, 2010.
[12] S. M. Kim and T. He. Freebee: Crosstechnology communication via free sidechannel. In *Proceedings of ACM MOBICOM*, 2015.
[13] S. M. Kim, S. Wang, and T. He. cetx: Incorporating spatiotemporal correlation for better wireless networking. In *Proceedings of ACM Sensys*, 2015.
[14] S. M. Kim, S. Wang, and T. He. Exploiting causes and effects of wireless link correlation for better performance. In *Proceedings of IEEE INFOCOM*, pages 379–387, April 2015.
[15] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis. Surviving wi-fi interference in low power zigbee networks. In *Proceedings of ACM Sensys*. ACM, 2010.
[16] E. LOVELACE, J. Sutton, and E. Salpeter. Digital search methods for pulsars. *Nature*, 222:231–233, 1969.
[17] S. Wang, A. Basalamah, S. M. Kim, G. Tan, Y. Liu, and T. He. A unified metric for correlated diversity in wireless networks. *IEEE Transactions on Wireless Communications*, 15(9):6215–6227, Sept 2016.
[18] S. Wang, S. M. Kim, Y. Liu, G. Tan, and T. He. Corlayer: A transparent link correlation layer for energy-efficient broadcast. *IEEE/ACM Transactions on Networking*, 23(6):1970–1983, Dec 2015.
[19] X. Zhang and K. Shin. Gap sense: Lightweight coordination of heterogeneous wireless devices. In *Proceedings of IEEE INFOCOM*, pages 3094–3101, April 2013.
[20] X. Zhang and K. G. Shin. Cooperative carrier signaling: Harmonizing coexisting wpan and wlan devices. *Networking, IEEE/ACM Transactions on*, 21(2):426–439, 2013.
[21] Y. Zhang and Q. Li. Howies: A holistic approach to zigbee assisted wifi energy savings in mobile devices. In *Proceedings of IEEE INFOCOM*, pages 1366–1374. IEEE, 2013.