Exploiting ZigBee in Reducing WiFi Power Consumption for Mobile Devices

Yifan Zhang and Qun Li, *Senior Member, IEEE*

Abstract—We present HoWiES, a system that saves energy consumed by WiFi interfaces in mobile devices with the assistance of ZigBee radios. The core component of HoWiES is a WiFi-ZigBee message delivery scheme that enables WiFi radios to convey different messages to ZigBee radios in mobile devices. Based on the WiFi-ZigBee message delivery scheme, we design three protocols that target three WiFi energy saving opportunities in scanning, standby and wakeup respectively. We have implemented the HoWiES system with two mobile devices platforms and two AP platforms. Our real-world experimental evaluation shows that our system can convey thousands of different messages from WiFi radios to ZigBee radios with an accuracy over 98 percent, and our energy saving protocols, while maintaining the comparable wakeup delay to that of the standard 802.11 power save mode, save 88 and 85 percent of energy consumed in scanning state and standby state respectively.

Index Terms—WiFi, 802.11, energy savings, ZigBee, 802.15.4, mobile devices

1 Introduction

Energy saving in mobile devices has received an increasing amount of attention recently [1], [2], [3], [4], [5], [6]. WiFi radio is one of the most commonly used wireless communication interfaces in mobile devices, and is a major source of energy consumption affecting user experience. We observe that there are three scenarios where a WiFi radio has to stay active without performing any real communications. *First*, a WiFi radio has to stay active to scan for networks in the scanning state. The power consumption for network scanning is considerably salient for lack of WiFi coverage in many places. *Second*, during Power Save Mode (PSM) standby, a WiFi radio needs to constantly switch to active to receive wireless access point (AP) beacons, and check if the AP has buffered its packets. Long WiFi idle durations (Section 3.1) lead to a non-negligible amount of WiFi energy consumption even during PSM standby. *Third*, when waken up from PSM standby, a WiFi radio has to stay active doing nothing while waiting for its turn to communicate with the AP if there are multiple devices contending for the channel.

The WiFi radio power consumptions in the above scenarios are significant: our measurements showed that the power consumptions of WiFi scanning and PSM standby in a Samsung Galaxy S2 smartphone accounted for 65 and 11 percent of the entire system power consumption respectively, and recent works [7], [8] showed that the wakeup contentions could cause up to four times more power consumption. To reduce WiFi power consumption in the these scenarios, we propose to delegate those WiFi operations to a low power ZigBee radio. In this case, WiFi radio will be turned off when there is no packet to transmit and receive, and the ZigBee radio is responsible for discovering the presence of WiFi networks and detecting if the AP intends for the device to communicate. This way, the significant power consumptions on WiFi radio in those scenarios can be reduced to reasonably low power consumptions of ZigBee radio.

ZigBee radio (i.e., IEEE 802.15.4 [9] compliant radio) is designed for low power communication working on 2.4 GHz ISM band, which coincides with most of the WiFi standards. Recently, an increasing amount of attempts have been made by both industry and academia to integrate ZigBee radios with smartphones for smart home applications [10] and energy savings [11], [12]. Indeed, as we did in our system implementation, a ZigBee radio can be directly connected to a mobile device via USB interface. With the unveiling of the first Android phone with ZigBee capability [13], and due to its usefulness in many areas, we believe that ZigBee will become a standard interface in mobile devices in the near future.

In this work, we design and implement HoWiES, a system that utilizes ZigBee radio to perform WiFi scanning and periodic AP beacon checking for the WiFi interface in the same device. The scenario is that a mobile device keeps its WiFi interface offline, while using its ZigBee radio to discover WiFi networks (during WiFi scanning) and to check if the associating AP has buffered the device's packets (during WiFi PSM standby). Given that it is not possible for ZigBee to decode WiFi packets directly, it is necessary to build a channel, through which a WiFi AP can convey information to ZigBee radio in mobile devices. Since the AP does not need feedbacks from the mobile device in the intended scenarios, a unidirectional channel from WiFi AP to ZigBee radio is sufficient. Fortunately, the same frequency band occupied by both WiFi and ZigBee allows ZigBee radio to sample background energy of WiFi transmission, which enables WiFi AP to convey information to ZigBee radio by using different WiFi transmission patterns. We demonstrate that, by using a simple coding scheme, our system can

• *The authors are with the Department of Computer Science, College of William and Mary, McGlothlin-Street Hall, Williamsburg, VA 23187-8795. E-mail: {yzhang, liqun}@cs.wm.edu.*

Manuscript received 3 May 2013; revised 23 Feb. 2014; accepted 16 Mar. 2014. Date of publication 6 Apr. 2014; date of current version 27 Oct. 2014.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2014.2315788

create a unidirectional communication channel in which an AP can deliver thousands of messages to ZigBee radios. When ZigBee decodes the information transmitted through the side channel, it either ignores the message if it is not for the device, or wakes up the WiFi radio for communication.

The WiFi-ZigBee message delivery scheme is the foundation of the entire HoWiES system. Based on this foundation, we have designed and implemented three protocols that specifically target the three significant WiFi energy saving opportunities in WiFi scanning, standby and wakeup respectively. We have implemented the HoWiES system on two mobile device platforms and two AP platforms. The evaluation results showed that our system could effectively save energy consumed by WiFi radios in mobile devices.

Our work is inspired by Esense [14], the first work proposing to enable information delivery from WiFi radio to ZigBee radio by using energy sampling. The scenario is that when there is a WiFi packet being transmitted in the air, a ZigBee radio, which is continuously sampling background energy, will generate a certain number (denoted as $\#^{+}_{consec}$) of consecutive positive energy samples (i.e., samples with energy readings greater than a certain threshold). Esense mainly studies the distributions of $\#^{+}_{consec}$ when sampling WiFi traffic generated by replaying several public WiFi traces. Esense proposes and validates that those $\#^{+}_{consec}$ that rarely appear can form an alphabet and different characters in the alphabet can be used to represent different information conveyed from WiFi to ZigBee. The major differences between this work and Esense, which are also our main contributions, can be summarized as follows.

First, instead of letting each character in the alphabet correspond to a piece of information, we study how to use the combinations of the characters to form different messages. With our method, it is possible to expand the message capacity *infinitely* while using only a small size of alphabet. In our prototype system, we are able to reliably convey 2,744 different messages from WiFi to ZigBee, while this number of Esense is 100. As we will discuss later, the enlargement of the message capacity is helpful for applying the message delivery scheme in wider application scenarios. However, it is a challenging to design a message encoding/decoding scheme that forms/detects messages by using/interpreting the combinations of alphabet characters, since the existence of normal WiFi packets will make the scheme suffer from detection errors. To address this challenge, we design a self-correcting message encoding/decoding scheme that can effectively reduce skewed messages due to the interference from normal WiFi packets.

Second, instead of focusing on studying the feasibility to enable WiFi radios to communicate with ZigBee radios, we target at designing and implementing a practical system that saves WiFi energy for mobile devices in different aspects by using our WiFi-ZigBee message delivery scheme. Although ZigBee radios are more power efficient than WiFi radios, it is not trivial to design and implement a system that saves WiFi energy in several aspects with the assistance of ZigBee radios. For example, an active ZigBee radio consumes comparable amount of energy to a WiFi radio that is in PSM standby. To save the energy a WiFi radio spends during standby, we have managed to get the ZigBee interface to synchronize with the wireless AP, and to duty-cycle the ZigBee the interface to reduce its power consumption.

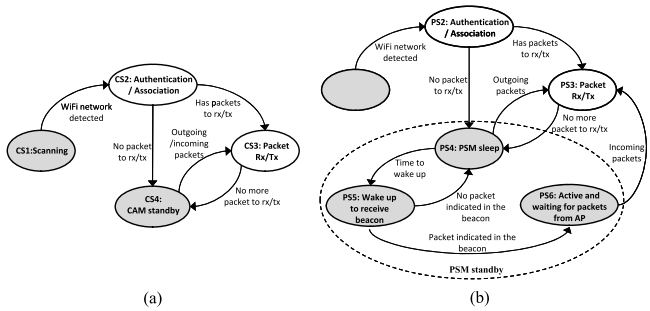


Fig. 1. Operating state diagrams of (a) CAM and (b) PSM stations. Shaded states have room for energy savings.

Third, instead of using trace-driven experiments, we evaluate our system with extensive real-world experiments. Our evaluation results show that our system can convey thousands of messages from WiFi radios to ZigBee radios with an accuracy over 98 percent, and our energy saving protocols, while maintaining the comparable wakeup delay to that of the standard 802.11 PSM, save 88 and 85 percent of energy consumed in WiFi scanning and standby respectively.

2 Background and Related Work

2.1 Background: WiFi Power Management

The power management mode of WiFi stations can be either Constantly Awake Mode (CAM) or Power Save Mode. CAM stations keep their WiFi radios active all the time. Fig. 1a shows the operating states of a CAM station. After detecting and associating with a WiFi AP, the CAM station switches its working state between "Receiving/Transmitting" ("Rx/Tx" for short) and "standby" (i.e., transitions between CS3 and CS4 in Fig. 1a). Since WiFi radio is active all the time, batteries in a CAM station drain at a rapid speed.

To save energy wasted in the CAM standby state, 802.11 Power Save Mode has been introduced [15]. Fig. 1b depicts the operating states of PSM stations. Similar to CAM stations, PSM stations also switch their working states between "Rx/Tx" and "standby" during operations. The difference is that PSM stations do not always keep their WiFi radios active in the standby state. Instead, PSM stations put their WiFi radios into sleep (i.e., state PS4 in Fig. 1b) most of the time during standby. In the sleep state, WiFi radios consumes very low power but is not able to receive or transmit. PSM stations in sleep state switch to "Rx/Tx" state whenever they have outgoing packets (i.e., transition PS4 to PS3). To receive incoming packets, a sleeping PSM station needs to periodically switch its WiFi radio to active (i.e., transition PS4 to PS5) to receive its AP's beacons, through which the AP advertises buffered packets for its sleeping clients. If there is no packet indicated in the beacon for the PSM station, the station simply goes back to the PSM sleep state (i.e., transition PS5 to PS4). Otherwise, the station stays active and waits for its incoming packets from the AP (i.e., transition PS5 to PS6). Then the station further switches to the "Rx/Tx" state on receiving the first incoming packet from its AP (i.e., transition PS6 to PS3). Upon the completion of the receptions/

transmissions, depending on detailed implementation, the station goes back to the sleep state either immediately or after a fixed amount of time without incoming or outgoing packets (i.e., transition PS3 to PS4).

2.2 Related Work

Energy saving in WiFi scanning. A common way for WiFi scanning energy savings is to perform WiFi availability prediction. For example, several works have considered, without turning on WiFi radios, predicting WiFi networks availability by using different context information [16], tracking and learning user movements [17], or collecting information about bluetooth devices and cell towers [18]. Though these solutions are shown to have reasonably good performance regarding WiFi availability prediction, they rely on certain information that may not always be available to perform prediction, and the prediction cannot achieve near-100 percent accuracy. To address these issues, ZiFi [11] proposes to detect WiFi networks with the assistance of ZigBee radios. The idea of ZiFi is to use ZigBee to detect WiFi beacon patterns, which indicate the existence of WiFi networks. Mobile devices in WiFi scanning state keep their WiFi interfaces offline, and turn them on when WiFi beacon patterns are detected by ZigBee radios.Similar to ZiFi, our solution also utilizes ZigBee radios in detecting WiFi availability.However, we take a different approach: we enable APs to advertise themselves by broadcasting messages that are understandable by ZigBee radios. Thus, an advantage of our solution over ZiFi is that with HoWiES, WiFi radios can be selectively waken up when suitable APs are encountered, which allows for further energy savings during scanning. For example, APs can encode WiFi network SSIDs, capabilities (e.g., with Internet connection or not), and association requirements (e.g., open or password verification needed) using our WiFi-ZigBee message delivery scheme. Consequently, a ZigBee radio is able to wake up the WiFi interface in the same device only when encountering suitable WiFi networks. Furthermore, the approach suggested by ZiFi cannot be used to save energy in the states of WiFi standby and WiFi wakeup.

Energy saving in WiFi standby. Two different approaches have been proposed to save energy in WiFi standby (i.e., when WiFi radio is idle). The first approach proposes to turn off WiFi radios when they are idle (i.e., state PS4 in Fig. 1b), and wake them up through a low-power non-WiFi channel when there are incoming WiFi activities. For example, Wake-on-wireless [19] establishes the low-power channel by attaching an additional device to both APs and WiFi clients. Wake-on-WLAN [20] uses ZigBee radio at the receiving end to detect incoming WiFi signal so as to wake up the WiFi radio. Though the idea of using ZigBee radio to sense WiFi energy is similar to ours, Wake-on-WLAN is designed to work in long distance point-to-point connection scenario, where the presence WiFi signal indicates the receiving end is intended for communication. Therefore, the ZigBee radio simply needs to wakeup WiFi radio whenever energy is sensed. Whereas we target normal WiFi networks where WiFi activities present all the time. As a result, we need to, by using ZigBee radio, identify which client is intended to be waken up by the AP. Cell2Notify [21] uses cellular channel to wakeup WiFi radios for VOIP calls. The other approach to save WiFi standby energy is suggested by E-MiLi [22], which proposes to reduce power consumption in WiFi idle listening state (i.e., state PS6 in Fig. 1b) by reducing clock-rates of WiFi chips. Since the two approaches target different portions of WiFi standby, they are complementing each other in saving WiFi standby energy.

In this work, we adopt the first approach: we establishes a low-power channel between APs and devices' ZigBee radios, through which APs can wake up standby devices selectively. The fundamental difference between our work and the existing works is our interference-resistant WiFi-ZigBee message delivery scheme, which enables WiFi senders to actively transmit a large amount of information to ZigBee radios in receivers without adding new hardwares. Whereas Wake-on-wireless [19] needs new devices at both sender and receiver side, and Wake-on-WLAN [20] and Cell2Notify [21] only enable receivers to detect the existence of the low-power channels. Therefore, our approach can be applied to wider application scenarios with less deployment overheads.

Energy saving in WiFi wakeup. Recent works have shown and addressed the energy waste problems caused by wakeup contentions between WiFi clients that belong to the same AP [7] or multiple interfering APs [8]. In HoWiES, our solution naturally solves the problem of wakeup contentions between clients associated with the same AP by waking up WiFi clients one at a time. To alleviate wakeup contention between clients associated with different APs, we coordinate APs such that there are not two interfering APs wake up their client at the same time (more details in Section 4.2.3).

Information encoding utilizing WiFi packet length. iPoint [23] designs a passive information kiosk system, where cellphones send their requests to the kiosk device by broadcasting WiFi packets with different lengths, and the kiosk device transmits information back to cellphones via an optical channel. In the application scenario of iPoint, there is no background WiFi traffic. Moreover, iPoint is equipped with a specially designed rectifier that can accurately detect the lengths of WiFi packets. On the contrary, our system has to deal with background WiFi traffic with off-the-shelf ZigBee radio, which imposes more significant challenges on accurate information encoding/decoding. Esense [14] is another work utilizing WiFi packet length in information delivery. We have introduced Esense and elaborated the differences between Esense and our work in Section 1.

3 Motivation

3.1 WiFi Energy Saving Opportunities

We observe the following three significant energy saving opportunities for WiFi stations.

Opportunity 1—scanning state. WiFi stations in scanning state constantly iterate through all the channels to search for available WiFi networks. We have measured the system power of two mobile platforms, a Samsung Galaxy S2 smartphone and a Lenovo T400 laptop, in the WiFi radio scanning state. From the measurement results (Table 1), we can see that about 65 and 12 percent of the system power consumption are spent in WiFi scanning for the smartphone

TABLE 1
System Power Consumption in WiFi Scanning State

	with WiFi scanning	with WiFi off	scanning/overall pert.
Galaxy S2	766 mW	265 mW	65.4%
T400	14498 mW	12732 mW	12.2%

and the laptop respectively. Therefore, we are motivated to find an energy efficient way for mobile devices to perform WiFi scanning.

Opportunity 2—standby state. As introduced in Section 2.1, in standby state, a Constantly Awake Mode station keeps its WiFi radio on all the time, and a Power Save Mode station puts its WiFi radio in a low-power state most of the time when there is no traffic, and periodically wakes up the radio to receive and check AP beacons for buffered packets. Table 2 presents the measurement results of the standby state power consumption of a Galaxy S2 smartphone and a T400 laptop, which are by default configured as PSM and CAM stations respectively by their device drivers. The smartphone consumes 33 mW more power, which accounts for about 11 percent of the overall system power, in the WiFi standby state than when the WiFi radio is turned off. This power overhead mainly comes from periodic wakeup to check beacons, because when we prolonged the smartphone's wakeup interval, the power overhead decreased accordingly. The laptop also consumes about 10 percent of its system power in the standby state.

We have performed an experimental study to investigate how much time stations stay in the standby state during WiFi sessions. In this experiment, we developed and deployed a WiFi activity recorder in our office building and in our university's library, both of which are with heave WiFi usage. The recorder sniffed WiFi packets and recorded their MAC addresses, packet types (e.g., data, management or control), packet sizes, data rates, received signal strength (RSS) and the packet reception times. To process the data, we first identified WiFi stations based on MAC addresses and packet types, and filtered out those stations whose packets have low RSS values, as the recorder may miss some of their packets because of low SNR. Then we analyzed the WiFi packets of the remaining stations to study how much time they were idle during WiFi sessions. Based on the 15 hours of WiFi activity data collected in 5 days, we identified 151 unique stations in 218 WiFi sessions. For each interval between two consecutive packets of the same WiFi session, if the interval (calculated by comparing at the two packet reception times) is longer than 5 seconds, we marked the interval as a WiFi standby period. Fig. 2a plots the CDF of WiFi session lengths, and (b) plots the CDF of standby percentages in the corresponding WiFi sessions. The results suggest that over 70 percent of stations spent more than 60 percent of their time in standby during their WiFi sessions. During these idle periods, WiFi radios stay in the standby state because of the absence of WiFi radio interface

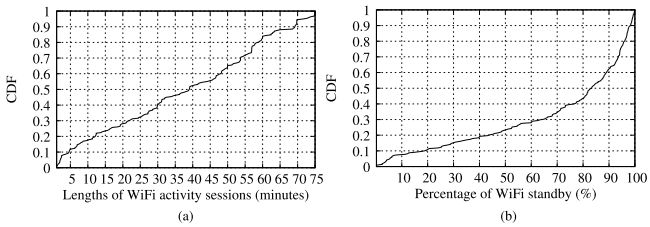


Fig. 2. The experiment on WiFi standby time: (a) is the CDF of WiFi session lengths. (b) is the CDF of the WiFi standby percentages in the corresponding WiFi sessions.

activity, which is caused by either long user think time (e.g., time spent on reading web page), or user behaviors of leaving their smartphones unused for most of the time [24], [25].

The significant standby power overhead (about 10 percent) and the large proportion of WiFi standby time over the entire WiFi session motivate us to design an energy-efficient way for WiFi standby. Ideally, WiFi radios should sleep without periodic wakeup or be completely off as long as there is no WiFi activities. Meanwhile, it must be still possible to wake up the WiFi radios if there are incoming packets for the WiFi radios.

Opportunity 3—energy waste due to wakeup contention. When multiple PSM stations working at the same channel and associated either with the same AP [7] or with multiple co-located APs [8], are waken up to receive buffered packets at the same time, the contention between the stations will force them to stay active without performing any real communications. This idle waiting could further causes about up to four times more energy consumption. Motivated by these research results, we want our approach to wake up standby WiFi radios to avoid these energy-expensive wakeup contentions.

3.2 ZigBee Radio Assisted WiFi Energy Savings

Compared with WiFi radios, ZigBee radios are more power efficient. Table 3 lists the power consumptions we measured of ZigBee radio CC2420 and WiFi radio BCM4330 in different operating modes. Since ZigBee consumes significantly less energy than WiFi, it would provide great assistance in saving WiFi energy for mobile devices if we could make ZigBee radios communicate with WiFi radios. Fortunately, the same frequency band (i.e., the 2.4 GHz band) used by ZigBee and most WiFi standards allows ZigBee radios to sample WiFi transmission energy. We exploit this feature in designing our WiFi-ZigBee message delivery scheme (Section 4.1).

As we discussed in Section 1, Esense [14] presents the first effort in enabling unidirectional communication from WiFi to ZigBee. However, the message capacity of Esense is small (about 100), limiting its application scenarios. For example, as we will show later, when applying WiFi-ZigBee message delivery in saving WiFi standby energy, each client of an AP

TABLE 2
System Power Consumption in WiFi Standby State

	with WiFi standby	with WiFi off	standby/overall pert.
Galaxy S2	298 mW	265 mW	11.1%
T400	14078 mW	12732 mW	9.6%

TABLE 3
Power Consumption of CC2420 and BCM4330

	CC2420 (ZigBee)	BCM4330 (WiFi)	ZigBee/WiFi ratio
Rx/Tx	56 mW	435 mW	0.129
Idle/Standby	1.2 mW	33 mW	0.036

needs to be assigned with an ID, which corresponds to a unique WiFi-ZigBee message. Consequently, a WLAN with multiple interfering APs would need more than 100 WiFi-ZigBee messages to assign to their clients for WiFi standby energy savings, because clients of interfering APs should be assigned with unique IDs (otherwise APs would wake up clients that are not associated with them). Furthermore, even more WiFi-ZigBee messages are needed if we consider applying WiFi-ZigBee message delivery in other scenarios. For example, when applying WiFi-ZigBee message delivery in saving WiFi scanning energy, as we discussed in Section 2.2, APs can use different messages to advertise their IDs and capabilities (e.g., SSIDs, with Internet connection or not, open AP or not), so that WiFi interfaces can be selectively waken up for further energy saving. In this case, the amount of WiFi-ZigBee messages needed in this scenario is proportional to the number of APs. Therefore, in order to allow the concept of WiFi-ZigBee message delivery to be applied in more scenarios, we need to design a WiFi-ZigBee message delivery scheme with large message capacity.

An alternative approach to enable WiFi APs to communicate with ZigBee radios is proposed by WiZi-Cloud [12]. WiZi-Cloud enables flexible ZigBee-based bidirectional communication between two WiFi devices by integrating additional ZigBee radios with WiFi devices. As we discussed previously, to save energy in the three intended scenarios, a unidirectional channel from AP to ZigBee radio is sufficient. Our WiFi-ZigBee message delivery scheme enables this channel without changing AP hardware. Finally, we choose ZigBee over Bluetooth [26], because ZigBee has better mandatory receive sensitivity level (-85 dBm according to [9]) than WiFi (-80 dBm according to [15]), while Bluetooth's receive sensitivity level is much lower (-70 dBm according to [26]). Low receive sensitivity would cause bluetooth not to be able to sense WiFi transmissions even when the device is within WiFi AP's range.

4 System Design

4.1 WiFi-ZigBee Message Delivery Scheme

4.1.1 The High Level Idea

Let us assume the messages that WiFi radios can deliver to ZigBee radios correspond to different numbers. A WiFi radio encodes the number that it wants to convey to a ZigBee radio by sending a sequence of WiFi packets (called *WiFi-ZigBee message packets*), whose sizes are chosen from a group of predefined values, using a fixed transmission rate. These predefined packets sizes form the *alphabet* of our message delivery scheme. The ZigBee radio determines the size of each packet by sampling background energy, and obtains the number that the WiFi radio wants to convey by interpreting the combination of packet sizes.

4.1.2 Alphabet Construction

The alphabet $\mathcal{A}$ is a set of b packet sizes: $\mathcal{A} = \{S_1, \ldots, S_b\}$, where $S_1 < \cdots < S_b$. In order to ensure that ZigBee radios can detect a WiFi-ZigBee message (abbreviated to "message" in later descriptions), we need to allow message packets to be distinguishable from normal WiFi packets. To this end, we first studied air times of normal WiFi packets by examining the 15-hours long WiFi traces obtained in the standby time experiment (Section 3.1.) By looking at the sizes and the transmission rates of the sniffed packets, we observed that WiFi packets transmitted using low transmission rates were small in size (these packets were usually 802.11 management/control frames like beacons and ACKs), and packets that were large in size were usually transmitted using high transmission rates (these packets were usually for massive data transmission like video streaming). Fig. 3 shows the CDF of the sniffed packet air times. It shows that over 95 percent of all the sniffed packets had an air time less than 1 millisecond. Therefore, to ensure WiFi-ZigBee message packets to be distinguishable from normal WiFi packets, we select large sizes for massage packets and sending them at the lowest transmission rate. Meanwhile, the difference between two adjacent predefined message packet sizes should be set appropriately to ensure ZigBee will not generate the same number of energy samples for message packets with different sizes. We will elaborate our choices of the predefined packet sizes for the alphabet later in Section 5.

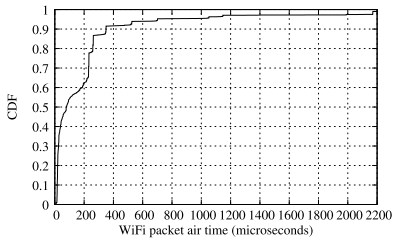


Fig. 3. WiFi packets airtime distribution.

4.1.3 WiFi-ZigBee Message Encoding

A WiFi radio encodes a WiFi-ZigBee message M by sending a sequence of l message packets, whose size are chosen from the alphabet $\mathcal{A}$, using the transmission rate R. Here we call l the *length* of the message. The value of the message is calculated as

$$v(M) = \sum_{i=1}^{i=l} (I_{p_i,\mathcal{A}} - 1) b^{i-1}, \tag{1}$$

where b is the size of the alphabet $\mathcal{A}$, p_i represents the ith of the l message packets, and $I_{p_i,\mathcal{A}}$ is the index of the packet p_i's size in the alphabet. For example, $I_{i,\mathcal{A}} = j$ if the size of packet p_i is S_j ($S_j \in \mathcal{A}, 1 \leq j \leq b$). The *capacity* of a message delivery scheme, which is the total amount of numbers that can be encoded, is b^l. Here R, l, b and $\mathcal{A}$ are fixed and shared between WiFi and ZigBee radios.

For instance, for a WiFi-ZigBee message delivery scheme where WiFi radios encode each message by transmitting three WiFi packets with sizes chosen from 100 and 200 bytes, the alphabet $\mathcal{A}$ is $\{100, 200\}$, the size of the alphabet b is 2, and the message length l is 3. The total number of messages that an WiFi radio can convey to a ZigBee radio is $2^3 = 8$ (i.e., the capacity of the scheme is 8). If a WiFi radio encodes a message by sending a sequence of three packets with 200, 100 and 200 B respectively, essentially it sends out three digits with values of 1, 0 and

1 in that order, and the message is interpreted as number 5 (i.e., $1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 = 5$).

4.1.4 Parameters Selection

As discussed in Section 4.1.2, to allow ZigBee radios to distinguish message packets from background packets, we ensure air times of message packets to be longer than the maximum packet air time of normal WiFi packets. Note that since the maximum air time of ZigBee packets are smaller than that of WiFi packets, message packets are also longer than normal background ZigBee packets if there exist some. Suppose in a WiFi network with base transmission rate R_w, a WiFi-ZigBee message delivery scheme should choose message packet transmission rate R and the smallest message packet size S_1 such that they satisfy $\frac{S_1}{R} > \frac{1500}{R_w}$, where 1,500 is the Ethernet MTU [27]. Meanwhile, to guarantee ZigBee radios will not have the same energy sampling count for two message packets with different sizes, the difference between two adjacent message packet sizes should be at least $\frac{2R}{H}$, where H is the background energy sampling frequency of ZigBee.

4.1.5 WiFi-ZigBee Message Detection and Decoding

Algorithm 1 presents the algorithm that ZigBee radios use to detect and decode WiFi-ZigBee messages. ZigBee radios detect WiFi-ZigBee messages by continuously sampling background energy with a frequency H. If a sample's energy reading is greater than a threshold E, the sample is a "positive" sample, otherwise it is a "negative" sample. In the algorithm, the variable PC (positive sample counter) records the number of the most recent consecutive positive energy readings that ZigBee radios have sampled, and the variable IC (message packet interval counter) records the time since the last message packet in terms of energy sample count. There are three working states in the algorithm. In the waiting message (WAITING_MSG) state (line 4-6), a ZigBee radio is waiting for a new WiFi-ZigBee message. Upon obtaining a positive sample it switches to the packet receiving (PKT_IN_PROGRESS) state (line 6). In the PKT_IN_PROGRESS state, the ZigBee radio keeps incrementing PC as it continuously gets positive samples (line 16). Upon receiving a negative sample, it decides whether the consecutive positive samples just observed came from a message packet or from a normal packet. If they came from a message packet (line 18-25), the ZigBee radio increments the message packet counter (line 19) and records the index of the packet's size in the alphabet (line 20). If all the message packets have been detected, it reports the message value based on the formula (1) (line 22), resets counters and switches back to the WAITING_MSG state (line 23). If there are message packets pending, it switches to the waiting message packet (WAITING_PKT) state (line 25). In the case that the consecutive positive samples came from a normal packet (line 27-34), the ZigBee radio switches back to the WAITING_MSG state directly if no message packet has been detected (line 28); otherwise, it counts the consecutive positive samples just observed into message packet interval (line 30). If the message packet interval is greater than a threshold, it switches back to the WAITING_MSG state (line 32). Otherwise, it goes to the WAITING_PKT state (line 34). In the WAITING_PKT state, the ZigBee radio keeps counting the message packet interval as they obtains negative samples (line 11), and ceases the decoding process if the interval is greater than the threshold (line 13). It goes to the PKT_IN_PROGRESS state once it obtains a positive sample (line 9).

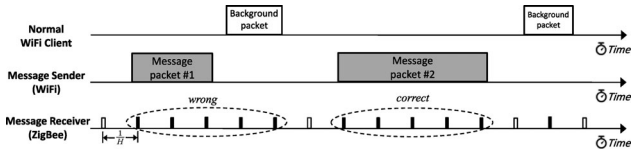


Fig. 4. An example of background packet interference.

Algorithm 1: WiFi-ZigBee message detection/decoding

Data: R, l, b, H, E and $\mathcal{A} = \{S_1, \cdots, S_b\}$.
Result: Report message value M once a message is detected.

```
1  PC, IC, i ← 0; state ← WAITING_MSG;
2  while ZigBee listening is enable do
3      Sample background energy, store the reading in e;
4      if (state == WAITING_MSG) then
5          if (e > E) /*on positive sample*/ then
6              PC ← 1; state ← PKT_IN_PROGRESS;
7      else if (state == WAITING_PKT) then
8          if (e > E) /*on positive sample*/ then
9              PC ← 1; state ← PKT_IN_PROGRESS;
10         else
11             IC++;
12             if (IC ≥ INTERVAL_TIME_OUT) then
13                 PC, IC, i ← 0; state ← WAITING_MSG;
14     else if (state == PKT_IN_PROGRESS) then
15         if (e > E) /*on positive sample*/ then
16             PC++;
17         else
18             if (PC ≥ HS_1/R) /*message packet detected*/ then
19                 i++;
20                 I_i ← j, if PC − ⌊HS_j/R⌋ < 2;
21                 if (i == l) /*message detected*/ then
22                     Report M = Σ_{i=1}^{i=l} (I_i − 1) b^{i−1};
23                     PC, IC, i ← 0; state ← WAITING_MSG;
24                 else
25                     IC ← 1; PC ← 0; state ← WAITING_PKT;
26             else
27                 if i == 0 /*no message packet has been detected*/ then
28                     PC, IC, i ← 0; state ← WAITING_MSG;
29                 else
30                     IC ← IC + PC;
31                     if (IC ≥ INTERVAL_TIME_OUT) then
32                         PC, IC, i ← 0; state ← WAITING_MSG;
33                     else
34                         PC ← 0; state ← WAITING_PKT;
```

4.1.6 Self-Correcting Message Encoding/Decoding

Without considering hidden terminals' effects, which is a case we will discuss later in Section 7, message packets will not overlap with normal packets in time domain because of the 802.11 CSMA/CA scheme. Due to the existence of background WiFi traffic, it is possible that a ZigBee radio obtains the same number of energy samples for two message packets with different sizes. Fig. 4 shows an example: A WiFi radio sends out two WiFi-ZigBee message packets on which a ZigBee radio normally generates 2-3 and 4-5 energy samples respectively. However, before the ZigBee radio could get the first sample after the first packet is transmitted, the channel is taken by another normal WiFi client, which transmits a packet causing the ZigBee radio to generate two positive samples on it. Then the positive samples of the first message packets is mistakenly counted as 5 instead of 3, which makes the ZigBee radio believe it has detected two message digits with the same value. We call this kind of problem *background (packet) interference*.

To address the above issue, we design a self-correcting message encoding/decoding algorithm, which extends the base encoding/decoding algorithm. With the self-correcting scheme, ZigBee radios can still extract the correct value of a message with high possibility even if background interferences exist. The essential idea of our approach is to enlarge the step length between message packet sizes such that the step length can accommodate multiple normal packets sent between two message packets. However, simply increasing the step length reduces b, the base of a message, which further reduces the capacity of the message delivery scheme. Our approach strikes a good balance between alleviating the impact of background interference and retaining the message delivery scheme capacity.

The fundamental observation supporting the self-correcting scheme is that when background interference happens, it only affects a minority amount of all the message packets of a WiFi-ZigBee message. Thus, we can utilize the majority of correctly detected message packet sizes to help correcting those wrongly detected message packet sizes. With the self-correcting scheme, the alphabet $\mathcal{A} = \{S_1, \ldots, S_b\}$ is divided into p sub-alphabets as $\mathcal{A}_1 = \{S_1, S_{p+1}, S_{2p+1} \ldots\}$, $\mathcal{A}_2 = \{S_2, S_{p+2}, S_{2p+2}, \ldots\}$, $\ldots$,$\mathcal{A}_p = \{S_p, S_{2p}, \ldots\ S_b\}$. To encode a message, a WiFi radio uses packet sizes in one randomly chosen sub-alphabet. To decode a message, a ZigBee radio gets the sizes of all the message packets using Algorithm 1. If all the sizes are from the same sub-alphabet, the ZigBee radio can calculate the message value directly. Otherwise, it indicates that there were background interferences happened to the message packets. In this case, the ZigBee radio first identify the correct sub-alphabet (notated as $\mathcal{A}_c$) as the sub-alphabet to which the majority packet sizes belong. Then it converts each of those packet sizes that are not in $\mathcal{A}_c$ to the value in $\mathcal{A}_c$ that is immediately smaller than the current wrong size. This approach extends the difference between two adjacent predefined packet sizes in the alphabet by a factor of p, which makes it possible to tolerate multiple interfering background packets. Meanwhile, the capacity of the message delivery scheme is shrunk from b^l to $b(\frac{b}{p})^{l-1}$.

For instance, suppose there is a message delivery scheme where the alphabet is $\mathcal{A} = \{100, 200, 300, 400\}$ and message length is 3. An self-correcting scheme with two sub-alphabet (i.e., $p = 2$) allows WiFi radios to send a WiFi-ZigBee message by transmitting three packets with sizes chosen from one of the two sub-alphabets: $\mathcal{A}_1 = \{100, 300\}$ and $\mathcal{A}_2 = \{200, 400\}$. If a ZigBee radio detects that the sizes of the three message packets are 300B, 100B and 300B, which are from the same sub-alphabet, it can directly conclude that the message value is $1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 = 5$. If the packet sizes are 300, 200 and 100 B respectively, it indicates that $\mathcal{A}_1$ is the correct sub-alphabet as there are two packet sizes chosen from $\mathcal{A}_1$, and that the second packet (whose size is 200 B) was affected by background interference. In this case, the ZigBee radio replaces the size 200 B in $\mathcal{A}_2$ with size 100 B in $\mathcal{A}_1$, and reports the message value as $1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 = 1$.

4.2 HoWiES Energy Saving Protocols

Based on the WiFi-ZigBee message delivery scheme, we designed three HoWiES energy saving protocols that save energy consumed in WiFi scanning, standby and wakeup, respectively. At the mobile device side, there are three components related to HoWiES operations: The *WiFi component* performs the ordinary 802.11 operations. The *ZigBee component* acts as a receiver in the WiFi-ZigBee message delivery scheme. The *HoWiES manager* is a software component that connects the components of WiFi and ZigBee and performs all the HoWiES management operations. At the AP side, each AP has a pool of WiFi-ZigBee message numbers, each of which is assigned to deliver a certain piece of information from WiFi radio to ZigBee radio. Within the message number pools of different APs, there are a set of common numbers that are used to advertise the existence of WiFi networks and to indicate buffered broadcast/multicast packets. Meanwhile, each AP uniquely assigns a number, called HoWiES ID, to each of its clients such that it can wake up a specific client in HoWiES standby.

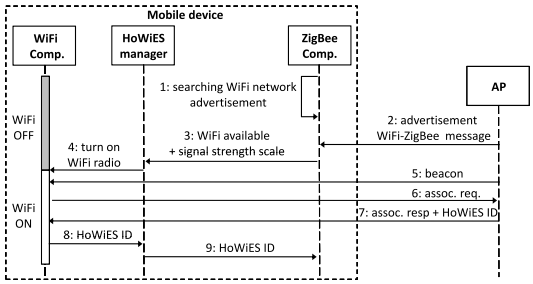


Fig. 5. HoWiES scanning and association operations.

4.2.1 HoWiES Sanning and Association

The HoWiES scanning and association protocol targets reducing power consumption in WiFi scanning and association. The protocol establishes a connection between APs and HoWiES-capable mobile devices. Fig. 5 shows the protocol.

HoWiES scanning. With this protocol, mobile devices trying to search and join a HoWiES-enable WiFi network keep their WiFi radios off while using the ZigBee radio to detect WiFi network advertisement messages broadcast regularly by HoWiES-enabled APs (Op.1). Among all the WiFi-ZigBee message numbers, APs use a set of common numbers to advertise their networks. During the scanning process, a HoWiES client determines whether to turn on its WiFi radio and associate to an AP based on the numbers encoded in the WiFi-ZigBee messages received. For example, to allow mobile devices to be able to determine whether to turn on WiFi radios and associate to APs based on whether the WiFi network is encrypted, the system operator can let open APs encode number 1 in their network advertisement WiFi-ZigBee messages while encrypted APs encode number 2 in the messages. In this case, mobile devices can selectively turn on their WiFi radios based on whether the encountered networks is encrypted.

HoWiES association. Upon detecting an advertisement message (Op.2), the ZigBee component notifies the HoWiES manager about the presence of a WiFi network and the scale of the WiFi signal strength calculated based on the energy samples of the message (Op.3). The HoWiES manager turns on the WiFi radio if the WiFi network meets the device's needs (Op.4). The WiFi radio sends an association request,

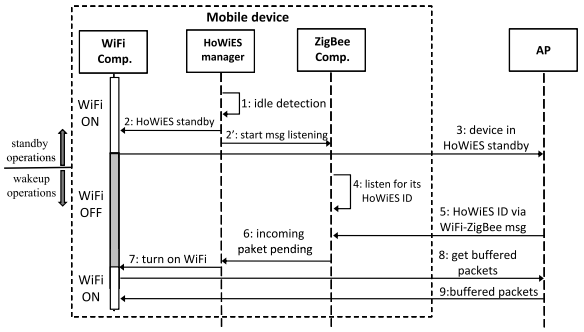


Fig. 6. HoWiES standby and wakeup operations.

indicating that the request issuer is HoWiES-capable, to the AP based on the information in the WiFi beacons (Op.5 and 6). If the association succeeds, the AP chooses a number from its message number pool to assign to the newly associated client as its HoWiES ID, and puts this ID in the association response (Op.7). Finally, the WiFi component extracts the ID from the association response and send it to the ZigBee radio via the HoWiES manager (Op.8-9).

4.2.2 HoWiES Standby

The HoWiES standby protocol aims to save power consumption caused by the periodical wakeup in standard PSM standby: WiFi stations in HoWiES standby do not need to periodically wake up and check AP beacons for buffered packets. Instead, they exploit their ZigBee interfaces to achieve the same goal. The HoWiES standby protocol specifies the process of how mobile devices switch to HoWiES standby.

Protocol details. The upper half of Fig. 6 shows the protocol. The HoWiES manager keeps monitoring the WiFi traffic on the mobile device (Op.1). On detecting that the WiFi radio has been idle for a predefined amount of time, the HoWiES manager notifies the WiFi radio to go into HoWiES standby state (Op.2). Then the WiFi radio informs the AP that it will switch to the HoWiES standby state and then turns itself off for energy savings (Op.3). Right after notifying the WiFi component to switch to HoWiES standby, the HoWiES manager turns on the ZigBee radio for WiFi-ZigBee message listening during standby (Op.2′). With this protocol, WiFi radios in HoWiES standby devices do not need to switch to active periodically to check beacons for buffered packets. Instead, they can just sleep all the time till the ZigBee radio detects wakeup messages sent from the AP.

Inactivity prediction. In the above design, we let both CAM devices and PSM devices switch to HoWiES standby only when their HoWiES managers predict that the durations of inactivity are longer than a threshold. The purpose of this rule for CAM devices is obvious: we want to reduce the performance impact to CAM devices brought by standby wakeup delays as performance has higher priority in CAM devices. For PSM devices, this rule will greatly reduce the overheads generated by WiFi-ZigBee messages on both network throughput and AP performances. As we will see later in Section 6.1.2, the message delivery scheme we have implemented has a negligible amount of overheads on network throughput and AP performances when the message sending frequency is less than 10 (i.e., 10 messages per second). However, when the sending frequency is larger than 10, the overheads increase linearly as the message sending frequency increases. Therefore, putting a PSM station into HoWiES standby only when the HoWiES manager predicts that the station will stay idle for a long duration will significantly reduce the overheads. The WiFi inactivity prediction could be achieved by combining statistical WiFi traffic history analysis [28] and user WiFi usage pattern learning, which is out of the scope of this paper.

4.2.3 HoWiES Wakeup

The HoWiES wakeup protocol specifies how WiFi stations wake up from HoWiES standby by utilizing their ZigBee interfaces. In the meantime, the protocol also tries to reduce power consumption overheads caused by wakeup contentions among WiFi stations associating with different APs.

Protocol details. The bottom half of Fig. 6 shows the HoWiES wakeup operations. During standby, the ZigBee component keeps listening for WiFi-ZigBee messages encoding the device's HoWiES ID (Op.4). Once the AP has buffered incoming packets for a HoWiES standby client, it wakes up the client by sending out a WiFi-ZigBee message that encodes the HoWiES ID assigned to the client in the association process (Op.5). If the buffered packets are broadcast/multicast packets, a common number, instead of the HoWiES ID, is encoded in the message. If there are multiple clients that have buffered packets, the AP wakes them up one by one in a FIFO manner. The ZigBee component informs the HoWiES manager about the buffered packets if it detects the number encoded by a WiFi-ZigBee message matches the device's HoWiES ID (Op.6). Then the HoWiES manager turns on the WiFi radio (Op.7), which in turn gets the buffered packets from the AP (Op. 8-9).

Wakeup contention reduction. The HoWiES wakeup protocol naturally solves the wakeup contention problem caused by waking up multiple WiFi clients associated with the same AP, because with the wakeup protocol, an AP wakes up its HoWiES standby clients one at a time. However, if multiple interfering APs (i.e., APs that are within each other's communication range) wake up their own clients at the same time, the awake times of the clients due to the wakeup contentions could be extended by a factor of 5 [8]. To solve this problem, we adopt a solution similar to that of SleepWell [8]. In our solution, we let each AP exclusively occupies a repeated *wakeup period*, during which it can wake up its clients to get their buffered packets, such that wakeup periods of any two interfering APs do not overlap. An AP's wakeup period starts at the beginning of each of its beacon period (i.e., right after a beacon is sent out), and lasts a duration of T_{dur}[1]. T_{dur} is calculated as the length of a beacon period averaged by the number of interfering APs. The interfering APs coordinate their beacon periods to ensure their wakeup periods do not overlap with each other.

1. In our system implementation, we optimized the T_{dur} by enabling the AP to indicate the end of a wakeup period when there is no client to wake up. See Section 5.1.3 for details.

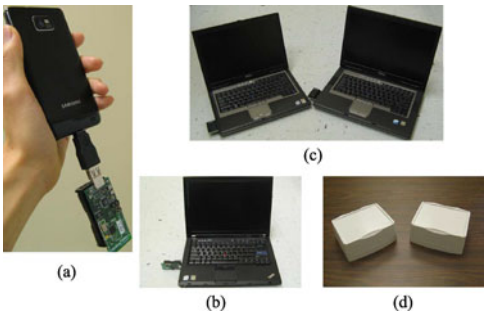


Fig. 7. HoWiES system implementation.

5 System Implementation

We have implemented a prototype HoWiES system with the devices shown in Fig. 7. The system consists of two types of entities: HoWiES clients and HoWiES APs. HoWiES clients are implemented in two mobile platforms: a smartphone platform (Samsung Galaxy S2, Fig. 7a) and a laptop platform (Lenovo T400, Fig. 7b). We enable ZigBee in both mobile platforms by integrating each of them with a TelosB mote that has a CC2420 ZigBee radio via USB interface. HoWiES APs are implemented in two AP platforms: a laptop platform (Dell Latitude D620/D820, Fig. 7c) and a standalone AP platform (Wiligear WBD-500 integrated radio platform, Fig. 7d). Table 4 lists the OS and the WiFi driver used in each device.

5.1 HoWiES Client

A HoWiES client has three major components: the WiFi component (consisting of the WiFi radio and the driver), the ZigBee component (consisting of the CC2420 ZigBee radio and the message detection/decoding TinyOS module), and the HoWiES manager.

5.1.1 Background Energy Detection

The CC2420 ZigBee radio has an RSSI register that records the RSS averaged over 8 symbol periods. The TinyOS provides an interface for programs to read the value of the RSSI register. However, according to our experience, the native TinyOS interface needs around 500 μs to get an RSS reading from the register. To increase the RSS sampling rate (so as to have more packet sizes for the alphabet), we have managed to reduce the RSS reading interval to about 150 μs. In our implementation, we set the ZigBee RSS reading interval to 180 μs (i.e., $H = 5555$) for stable performances.

5.1.2 Message Detection/Decoding

The ZigBee component continuously detects and decodes all WiFi-ZigBee messages by running Algorithm 1, and notifies the HoWiES manager about the messages that are related to the hosting mobile device (e.g., WiFi network advertisements and the device's HoWiES ID).

5.1.3 Duty Cycling ZigBee Radio

According to our measurement, the power consumption that a TelosB mote has when it is sampling background energy is about 60 mW, which is higher than the standby WiFi power overhead in Galaxy S2 (33 mW). To solve this issue, we duty-cycle the ZigBee interface in HoWiES clients: the ZigBee interface is put into active only during when the HoWiES AP tries to wake up its clients. Specifically, a HoWiES client's ZigBee radio samples background energy only during the wakeup period of its AP. To synchronize ZigBee radios with the corresponding APs' wakeup periods, we let APs broadcast the durations of their current wakeup periods (i.e., T_{dur}) via beacons. Then the HoWiES manager enables ZigBee energy sampling only in the first T_{dur} of time of the corresponding AP's beacon period (recall that each AP's wakeup period starts at the beginning of its beacon period). Meanwhile, to avoid the case where a HoWiES client's ZigBee radio stays active during its AP's wakeup period while the AP has no client to wake up, we enabled the HoWiES AP to indicate there is no more client to wake up within the current wakeup period by sending out a special WiFi-ZigBee message. Upon receiving this special WiFi-ZigBee message, the HoWiES client immediately puts its ZigBee radio into sleep. Furthermore, before an AP has to adjust its beacon period (because of topology changes of interfering APs), it wakes up all its HoWiES standby clients to let them be able to re-synchronize to its new wakeup period. By using the previous duty-cycling operations, we were able to reduce the energy sampling power consumption of TelosB mote to 5 mW in our evaluation experiments.

TABLE 4
OSes and WiFi Drivers of Implementation Devices

Device	Operating System	Driver
Samsung Galaxy S2	Android 2.3 (Linux 2.6.35)	DHD
Lenovo T400	Ubuntu 10.04 (Linux 2.6.32)	ath9k
Dell Latitude D820	Ubuntu 10.04 (Linux 2.6.32)	madwifi
Wiligear WBD-500	OpenWrt 8.09 (Linux 2.6.26)	madwifi

5.1.4 The HoWiES Manager

The HoWiES manager is implemented as a Linux kernel module in the mobile device's OS. It is responsible for turning on/off WiFi radios as specified in the protocols, controlling background energy sensing in ZigBee radio and relaying information between the WiFi and the ZigBee components. The HoWiES manager communicate with the ZigBee component via USB serial connection.

5.2 HoWiES AP

5.2.1 WiFi-ZigBee Message Parameters Selection

In our implementation, HoWiES APs send out a WiFi-ZigBee message by transmitting 3 packets (i.e., $l = 3$) with a transmission rate of 1 Mb/s (i.e., $R = 1$ Mb/s). We performed experiments to find out how stable the CC2420 radio generates energy samples in sampling packets with a fixed length. Ideally, the CC2420 radio will generate the same number of samples if the lengths of sensed packets are fixed. However, in our experiments, we found that the CC2420 radio could produce four different numbers of energy samples for the same WiFi packet size. Therefore, to ensure ZigBee will not generate the same number of energy samples for two message packets with different sizes, we set the difference between two adjacent packet sizes in the alphabet to 90 bytes (i.e., $\frac{4R}{H}$), which gives us 14 packet sizes

TABLE 5
Reliability and Accuracy of the WiFi-ZigBee Message Delivery Scheme in the Uncontrolled Experiment

Reliability	Accuracy	
Total msg detected/sent	Correct msg/detected (w/o self-correction)	Correct msg/detected (w/ self-correction)
19,904/20,000	19,223/19,904	19,737/19,904
99.5%	96.6%	99.2%

for the alphabet: $\mathcal{A} = \{300, 390, \ldots, 1470\}$. Thus, the smallest air time for the message packet is 2.4 millisecond, which is larger than the air times of all the sniffed WiFi packets obtained in our experiment described in the "Alphabet construction" section. As a result, our system can deliver 2,744 (i.e., b^l) different messages from WiFi radios to ZigBee radios. Since an AP usually maintains connections with tens of clients (for example, the default maximum clients per AP is 128 in the madwifi driver), the message capacity our implementation has achieved is adequate for a normally deployed WiFi network.

5.2.2 WiFi-ZigBee Message Packets Transmission

HoWiES APs transmit message packets using a user space packet sending program implemented with the libpcap library [29]. The user space program and the WiFi driver located in kernel space are connected by using the Linux usermode-helper API.

6 System Evaluation

6.1 WiFi-ZigBee Message Delivery

6.1.1 Reliability and Accuracy

The message delivery scheme needs to be *reliable*, which means HoWiES clients should reliably detect WiFi-ZigBee messages sent by HoWiES AP without firing any false alarms (i.e., reporting messages when there is none). Meanwhile, the message delivery scheme needs to be *accurate*, which means HoWiES clients should be able to correctly decode the detected messages.

Uncontrolled experiment. We have performed an uncontrolled experiment to evaluate the reliability and the accuracy performances of the implemented message delivery scheme in real WiFi environments. We deployed a HoWiES AP and client pair in the university's library, and performed the experiment from 8 PM to 10 PM, a time section during which the library are full of students surfing web and watching online videos, in several days. In the experiment, the AP sent different numbers to the client in different rounds. In each round, the HoWiES AP randomly chose a number from 1 to 2,744, encoded the number into a WiFi-ZigBee message and transmitted the message for 100 times with an interval of 100 ms. The chosen number is recorded such that we can use it as ground truth when deciding if the client has correctly decoded the messages. The HoWiES client detected and decoded the messages using the base message encoding/decoding algorithm (i.e., without using the self-correcting scheme), and output the results to a data file for analysis. We ran the experiment for 200 rounds. Table 5 shows the results. For the total 20,000 WiFi-ZigBee messages, 99.5 percent of them were detected by the HoWiES client. Within all the detected messages, the HoWiES client correctly decoded 96.6 percent of them. We then examined all the wrongly decoded messages as follows. We marked an wrongly decoded message as correctable using the self-correcting scheme with 2 sub-alphabets (i.e., $p = 2$), if the following conditions are satisfied. First, there is only one message packet whose size is wrongly detected (since we use $l = 3$, one is the maximum minority number). Second, the wrong size's index in the alphabet is greater than the actual size's index in the alphabet by 1 (if using $p = 3$, this value is 2). We found that after using the self-correcting algorithm, the accuracy of the message decoding increased to 99.2 percent. We further examined what caused the rest uncorrectable messages. There are two reasons. The first reason is that some messages have more than one message packet whose size is wrongly detected. The second reasons is that although there is only one wrong message packet size, the energy samples count for that packet is less than the expected value. This might be because of the imperfection of CC2420 hardware implementation of energy detection.

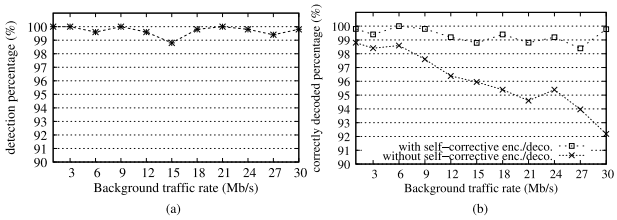


Fig. 8. Reliability and accuracy of the WiFi-ZigBee message delivery scheme in the controlled experiment.

Controlled experiment. We also conducted a controlled experiment to study how the message delivery reliability and accuracy performances respond to the changes of background traffic. In this experiment, we produced background traffic by establishing a direct iperf UDP connection between two 802.11g WiFi nodes (UDP packet size was 1,500 bytes). We varied the connection bandwidth between the two nodes and observed how our message delivery scheme responded to that. We have tested background traffic bandwidth from 1 Mb/s to the *saturated* bandwidth (30 Mb/s) with a step length of 3 Mb/s. Similar to the uncontrolled experiment, the HoWiES AP transmitted messages encoding a randomly selected number, without using the self-correcting algorithm, for 100 times in each round. With each background traffic bandwidth, we performed the test for 100 rounds. Fig. 8a presents the message delivery's reliability performance. For all the tested background traffic bandwidths, our scheme can correctly detect at least 99 percent of them. Fig. 8b shows the accuracy performance. Without using the self-correcting encoding/decoding algorithm, the accuracy ratio decreased as the background traffic bandwidth increased. For the saturated background traffic bandwidths, the accuracy percentage was 92 percent. Similar to the uncontrolled experiment, we analyzed all the wrongly decoded messages and marked those that were correctable. After applying the self-correcting algorithm, the accuracy percentages for all the background traffic bandwidths increased to at least 98 percent.

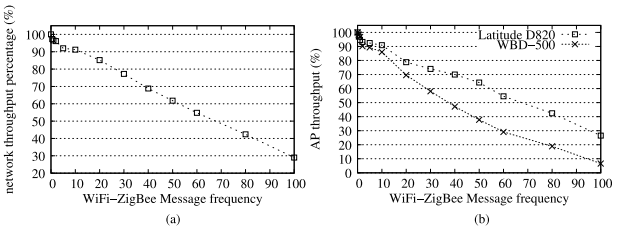


Fig. 9. HoWiES WiFi-ZigBee message delivery overheads.

6.1.2 Message Delivery Overheads

Overhead on network throughput. To evaluate the message delivery overheads imposed on network throughput, we tested the iperf UDP bandwidth (packet size was 1,500 bytes) between two *directly* connected WiFi nodes while a HoWiES AP was sending WiFi-ZigBee messages with different frequencies in vicinity. We have tested the message sending frequencies (Hz) of 0.5, 1, 2, 5, 10, 20 , . . . , 60, 80 and 100. Fig. 9a shows the experiment result. With the message sending frequencies (Hz) of 0.5, 1, and 2, there were only a negligible amount of throughput degradation on network throughput. With the sending frequencies of 5 and 10, the tested iperf connection still had 90 percent of its bandwidth. Then the network bandwidth decreased approximately in linear as the message sending frequency increased.

Overhead on AP performance. To evaluate the overheads imposed on AP performances, we established an iperf UDP connection between two WiFi node *via* a HoWiES AP. Then we tested the bandwidth between the two WiFi nodes while the HoWiES AP varied the WiFi-ZigBee sending frequencies in the same way as in the previous network overhead experiment. We tested our implementation on two different AP platforms: the Dell Latitude D820 laptop and the Wiligear WBD-500 standalone AP. Fig. 9b shows the experiment result. Similar to the network overhead experiment, both AP platforms has a small amount of throughput degradation when the message sending frequency is smaller than 10. When the sending frequency is higher than 10, the throughputs on both platforms decreased linearly as the message sending frequency increased. The WBD-500 standalone AP had a faster performance drop than the Dell laptop. This is because the standalone AP has more constrained computational resources.

The conclusion. From the experiments we learn that our message delivery scheme has a negligible amount of overheads on both network throughput and AP performance when the message sending frequency is less than 10 (i.e., 10 messages per second). When the sending frequency is higher than 10, the overheads increase linearly as the frequency increases. These results support the needs of inactivity prediction in the HoWiES standby protocol.

6.2 Energy Gain Achieved by the Energy Saving Protocols

6.2.1 Power Measurement Setup and Methodology

To measure the power consumption in the T400 laptop, we used the smart battery interface come with the operating system. According to [30], the smart battery interface is highly accurate when the battery interface reading rate is low. Since we were only interested in long term energy consumptions, the smart battery interface satisfied our requirements. To measure the power consumption in the smartphone, we used the Monsoon power monitor [31], which provides accurate power readings for hand-held mobile devices. When we measured the power of a device, we turned off all the unnecessary applications and services, and shut down the power-hungry LED screen. To get the power consumption value for a WiFi operation (e.g., scanning or standby) in a device, we first measured the baseline system power consumption (i.e., system power consumption without running any WiFi operations). Then we measured the system power when the device was continuously performing the targeted WiFi operation. Finally, the difference between the two values was calculated as the power consumption of the WiFi operation.

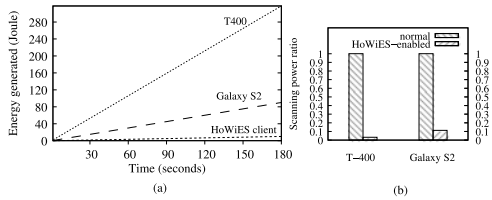


Fig. 10. Energy gain on WiFi scanning state.

6.2.2 Energy Gain in WiFi Scanning

We measured the WiFi scanning power consumptions of three devices: a normal T400 laptop, a normal Galaxy S2 smartphone and a HoWiES client. Our measurement showed that the T400 laptop, the Galaxy S2 smartphone and the HoWiES client spent 1,740, 501 and 61 mW for WiFi scanning respectively. Fig. 10a shows the energy generated by the WiFi scanning operation during a 3 minutes period. Fig. 10b shows the percentages of WiFi scanning power reduction of the HoWiES client when compared to the normal mobile devices. From the result we can conclude that our scheme can effectively reduce power consumptions for the WiFi operation in mobile devices.

6.2.3 Energy Gain in WiFi Standby

To evaluate the power savings achieved in the WiFi standby state, we compared a Galaxy S2 smartphone and its HoWiES-enabled version. In the experiment, three WiFi clients were associated with the AP. AP beacon frequency was set to 10 beacons/second. The AP waked up all the three clients in each beacon period. Our measurement showed that the normal Galaxy S2 and the HoWiES-enabled Galaxy S2 consumed 33 and 5 mW in the standby state respectively. Fig. 11a shows the energy generated in standby during a 3 minutes period. Fig. 11b compares the standby power

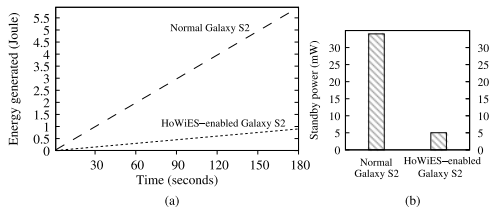


Fig. 11. Energy gain on WiFi standby state.

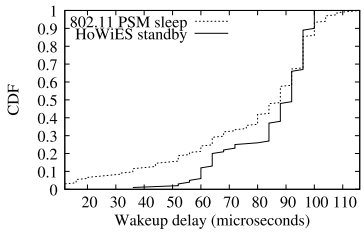


Fig. 12. Empirical HoWiES wakeup delay CDFs.

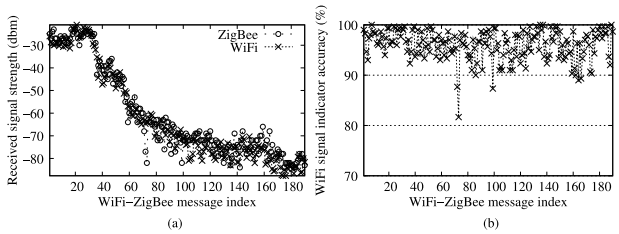


Fig. 13. Accuracy of ZigBee based WiFi signal strength indicator.

consumption between the two subjects. Although the absolute value of power consumption gain is small at the first glance, it is still quite meaningful considering that users usually leave the WiFi radios in their mobile devices idle most of the time.

6.3 HoWiES Wakeup Delay

We evaluated the delay performance of our implemented system in terms of waking up a standby client. To do the evaluation, we instrumented the WiFi device driver in AP to record the times that a 802.11 PSM standby client and a HoWiES-standby client needs to wake up and get their buffered packets: when the first incoming packet of a standby client is enqueued, the AP records the packet enqueue time T_s and wakes up the standby client to get its packets (through either standard 802.11 PSM wakeup operation or HoWiES wakeup operation). The AP records the time T_e when the client notifies the AP that it is ready to receive the buffered packets. The time that the client used to perform the wakeup operation is calculated as $T_e - T_s$.

On the clients side, the wakeup interval of the normal Galaxy S2 was set to a beacon period, which is the default setting used by the WiFi driver. For the HoWiES-enabled Galaxy S2, it went to sleeping state once it enters HoWiES standby, and kept sleeping until it was waken up by a WiFi-ZigBee message. Fig. 12 shows the empirical CDF of time that a normal Galaxy S2 and a HoWiES-enabled Galaxy S2 needs to wake up. Through the figure we can see that the wakeup delay of our implemented system is already comparable to that of a normal 802.11 PSM client. Actually there was still room to improve the wakeup latency in our implementation. For example, currently an AP used a user space program to transmit message packets. This would incur some extra time in the kernel-user space communication. Moreover, the user space program cannot set its packets to have higher transmission priority than other packets, which may cause more extra time between two message packets. the HoWiES wakeup delay to the future work.

6.4 WiFi Signal Strength Indicator by Using ZigBee

In this experiment, we evaluate how accurate the WiFi SSIs generated by ZigBee radios are when compared to SSIs that are generated by a WiFi radio. In the experiment, we let a HoWiES AP transmit WiFi-ZigBee messages continuously with an interval of 1 second. Then we walked further away from the HoWiES AP carrying a HoWiES client, which ran a program that captured all the WiFi-ZigBee message packets using a WiFi sniffer while the ZigBee radio was continuously detecting, decoding and recording messages. To process the data, we first correlated the 3 WiFi packets (recall that each WiFi-ZigBee messages is encoded by three WiFi packets) with the corresponding WiFi-ZigBee message. The SSI of each WiFi-ZigBee message generated by the ZigBee radio is calculated as the average of all the positive energy samples of the message, and the SSI generated by the WiFi radio is the average of the three corresponding WiFi packet's SSI. We plot the SSI values generated for each message by both radios in Fig. 13a, and plot the accuracy percentage of each SSI generated by the ZigBee radio compared to the corresponding WiFi generated SSI. From the figure we can see that ZigBee generated SSI can accurately reflect the actual WiFi signal strength.

7 Additional Discussion

Dealing with hidden terminals. When designing the self-correcting message encoding/decoding scheme, we assume that two WiFi packets will not overlap in time domain due to 802.11 CSMA/CA. However, if there are two hidden nodes transmitting without knowing each other, their packets could overlap in time domain. In this case, the two overlapping packets may have an air time equal to a WiFi-ZigBee message packet, causing a sampling ZigBee radio to have wrong detections. Similar to the existing solutions dealing with the hidden terminal problems, we address this issue by using retransmissions: when an AP sends a message encoding a client's HoWiES-ID to wake up the client, it will keep sending the message with a certain interval until the client wakes up and fetches the buffered packets.

Standby clients wake up overheads. Our evaluation (Section 6.1.2) shows that the WiFi-ZigBee message delivery scheme has a negligible amount of overheads on both network throughput and AP performance when the message sending frequency is less than 10 (i.e., 10 messages per second). When the sending frequency is higher than 10, the overheads increase linearly as the frequency increases. In our current protocols, frequently waking up a HoWiES standby client would cause AP to send WiFi-ZigBee messages with a high frequency. Therefore, in the HoWiES wakeup protocol, the HoWiES manager in a mobile device puts the WiFi radio into HoWiES standby only when the it predicts that the client will stay idle for a long duration (Section 4.2.2).

System deployment overheads. Usually there is a tradeoff between the overheads of deploying a system and the benefits of the system. This is also true with HoWiES. For example, the deployment of HoWiES needs more installation efforts than Esense [14], since our approach requires protocol changes at AP side. However, in exchange, our approach can be applied to wider application scenarios where larger WiFi-ZigBee message capacity is needed (e.g., AP advertisements for flexible WiFi wakeup during scanning and client wakeup

during WiFi standby, as discussed in Section 3.2). Another example is that WiZi-Cloud [12] enables bidirectional ZigBee-based communication between two WiFi devices by integrating additional ZigBee radios with the WiFi devices, while our system achieves unidirectional communication from APs to ZigBee radios without changing AP hardware. As for the deployment of our system, although it may be difficult to deploy the system widely by changing all the APs, we envision the system can be deployed in house-wide or campus-wide WiFi networks.

Applicable frequency band. WiFi devices working on the 5 GHz frequency band cannot benefit from HoWiES, because ZigBee, which works on the 2.4 GHz frequency band, cannot sense 5 GHz WiFi transmissions. For the reason that the 2.4 GHz band and the 5 GHz band have their own advantages over each other, these two bands tend to coexist for WiFi in the future. Fortunately, since most WiFi APs and devices can work on both the 2.4 GHz band the 5 GHz band, users can opt to configure their APs and devices to work on the 2.4 GHz band to take advantage of HoWiES.

Working range between WiFi AP and ZigBee receiver. HoWiES utilizes ZigBee to perform non-communication operations for WiFi to achieve WiFi energy savings. A concern about this approach is the effectiveness of ZigBee receiver to sense AP's WiFi packets within AP's radio range. As we have experimentally shown in Section 6.4, ZigBee is able to accurately report WiFi signal strength within the radio range of the WiFi sender. This is because ZigBee has a comparable receive sensitivity as WiFi (in fact, the mandatory receive sensitivity of ZigBee, which is -85 dBm [9], is even better than that of WiFi, which is -80 dBm [15]). Therefore, there should be no range issue with HoWiES when it is used in devices with single WiFi antenna (which is the case for all current smartphones). When HoWiES is used in Multiple-Input and Multiple-Output (MIMO) WiFi communications, ZigBee may not be able to reach the full ranges of MIMO APs. However, range-optimized MIMO communications occupy only a small fraction of WiFi communication scenarios. Therefore, HoWiES can still be fully functional in most WiFi usage scenarios, and with most WiFi devices (e.g., all current smartphones).

Variable message length. In our current design, all WiFi-ZigBee messages have the same length (i.e., use the same number of packets to encode different messages). A promising way to increase the efficiency of the message delivery scheme and to reduce message delivery overheads is to use less packets to encode those frequently used messages and more packets to encode those rarely used messages (which is an idea similar to Huffman coding). We leave this exciting improvement to our future work.

8 Conclusion

We have presented HoWiES, a Wifi energy saving system that achieves WiFi energy savings in three different aspects: scanning energy saving, standby energy saving and standby wakeup contention reduction. The foundation of the HoWiES system is a novel WiFi-ZigBee message delivery scheme that enables WiFi radios to deliver different information to ZigBee radios. Our extensive evaluations show that our implementation of the WiFi-ZigBee message delivery scheme works accurately and reliably with reasonable overheads, and that the whole system can effectively save energy for WiFi devices.

Acknowledgments

The authors would like to thank all the reviewers for their insightful comments. This work was partly supported by the US National Science Foundation (NSF) Grants CNS-1320453, CNS-1117412, CAREER Award CNS-0747108.

References

[1] H. Han, Y. Liu, G. Shen, Y. Zhang, and Q. Li, "DozyAP: Power-efficient Wi-Fi tethering," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Serv.*, 2012, pp. 421–434.

[2] Y. Zhang and Q. Li, "HoWiES: A holistic approach to ZigBee assisted WiFi energy savings in mobile devices," in *Proc. IEEE Conf. Comput. Commun.*, 2013, pp. 1366–1374.

[3] F. Xu, Y. Liu, Q. Li, and Y. Zhang, "V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics," in *Proc. 10th USENIX Conf. Netw. Syst. Des. Implementation*, 2013, pp. 43–56.

[4] Y. Zhang, X. Wang, X. Liu, Y. Liu, L. Zhuang, and F. Zhao, "Towards better CPU power management on multicore smartphones," in *Proc. Workshop Power-Aware Comput. Syst.*, 2013, article 11, pp. 11:1-11:5.

[5] Y. Zhang, C. C. Tan, and Q. Li, "CacheKeeper: A system-wide web caching service for smartphones," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2013, pp. 265–274.

[6] W. Li, Y. Cui, X. Cheng, M. Al-Rodhaan, and A. Al-Dhelaan, "Achieving proportional fairness via AP power control in multirate WLANs.," *IEEE Trans. Wireless Commun.*, vol. 10, no. 11, pp. 3784–3792, Nov. 2011.

[7] E. Rozner, V. Navda, R. Ramjee, and S. K. Rayanchu, "NAPman: Network-assisted power management for wifi devices," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Serv.*, 2010, pp. 91–106.

[8] J. Manweiler and R. R. Choudhury, "Avoiding the rush hours: WiFi energy management via traffic isolation," in *Proc. 9th Int. Conf. Mobile Syst., Appl., Serv.*, 2011, pp. 253–266.

[9] *IEEE-SA*, IEEE Std 802.15.4-2006.

[10] Texas Instruments News Center, *TI demonstrates world's first Android development platform bringing ZigBee and ZigBee RF4CE to Smartphones and Tablets*, 2011.

[11] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma, "ZiFi: Wireless LAN discovery via ZigBee interference signatures," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw.*, 2010, pp. 49–60.

[12] T. Jin, G. Noubir, and B. Sheng, "WiZi-Cloud: Application-transparent dual ZigBee-WiFi radios for low power Internet access," in *Proc. IEEE Conf. Comput. Commun*, 2011, pp. 1593–1601.

[13] TazTag. (2012). *TPH-ONE* [Online]. Available: http://www.taztag.com/PR_TPH-ONE_revMD_V4.pdf

[14] K. Chebrolu and A. Dhekne, "Esense: Communication through energy sensing," in *Proc. 15th Annu. Int. Conf. Mobile Comput. Netw.*, 2009, pp. 85–96.

[15] *IEEE-SA*, IEEE Std 802.11-2007.

[16] A. Rahmati and L. Zhong, "Context-for-wireless: Context-sensitive energy-efficient wireless data transfer," in *Proc. 5th Int. Conf. Mobile Syst., Appl. Serv.*, 2007, pp. 165–178.

[17] A. J. Nicholson and B. D. Noble, "BreadCrumbs: Forecasting mobile connectivity," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 46–57.

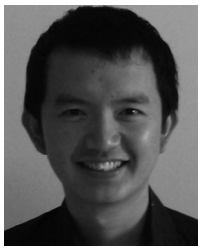
[18] G. Ananthanarayanan and I. Stoica, "Blue-Fi: Enhancing Wi-Fi performance using bluetooth signals," in *Proc. 7th Int. Conf. Mobile Syst., Appl., Serv.*, 2009, pp. 249–262.

[19] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on wireless: An event driven energy saving strategy for battery operated devices," in *Proc. 8th ACM Int. Conf. Mobile Comput. Netw.*, 2002, pp. 160–171.

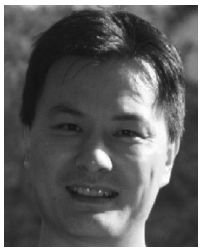
[20] N. Mishra, K. Chebrolu, B. Raman, and A. Pathak, "Wake-on-WLAN," in *Proc. 15th Int. Conf. World Wide Web*, 2006, pp. 761–769.

[21] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. K. Gupta, "Wireless wakeups revisited: energy management for VOIP over WiFi smartphones," in *Proc. 5th Int. Conf. Mobile Syst., Appl. Serv.*, 2007, pp. 179–191.

[22] X. Zhang and K. G. Shin, "E-MiLi: Energy-minimizing idle listening in wireless networks," in *Proc. 17th Annu. Int. Conf. Mobile Comput. Netw.*, 2011, pp. 205–216.
[23] H. Javaheri and G. Noubir, "iPoint: A platform-independent passive information kiosk for cell phones," in *Proc. IEEE 7th Annu. Commun. Soc. Conf. Sensor Mesh Ad Hoc Commun. Netw*, 2010, pp. 1–2.
[24] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in smartphone usage," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Serv.*, 2010, pp. 179–194.
[25] A. Shye, B. Scholbrock, G. Memik, and P. A. Dinda, "Characterizing and modeling user activity on smartphones: Summary," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst.*, 2010, pp. 375–376.
[26] Bluetooth SIG,*BLUETOOTH SPECIFICATION Version 4.0*, 2010.
[27] IETF Network Working Group, *RFC-1191*, 1990.
[28] J. Liu and L. Zhong, "Micro power management of active 802.11 interfaces," in *Proc. 6th Int. Conf. Mobile Syst., Appl., Serv.*, 2008, pp. 146–159.
[29] Libpcap library. [Online]. Available: http://www.tcpdump.org/
[30] M. Dong and L. Zhong, "Self-constructive high-rate system energy modeling for battery-powered mobile systems," in *Proc. 9th Int. Conf. Mobile Syst., Appl., Serv.*, 2011, pp. 335–348.
[31] Monsoon solutions. [Online]. Available: www.msoon.com/LabEquipment/PowerMonitor



Yifan Zhang received the BS degree in computer science from the Beihang University, Beijing, China, in 2004, and is currently working toward the PhD degree in computer science at the College of William and Mary, Williamsburg, VA. His research interests include wireless networks, mobile computing systems, and operating systems.



Qun Li received the PhD degree in computer science from Dartmouth College, Hanover, NH, in 2004. He is an associate professor with the Department of Computer Science, College of William and Mary, Williamsburg, VA. His research interests include wireless networks, sensor networks, RFID, and pervasive computing systems. He received the US National Science Foundation (NSF) Career Award in 2008. He is a senior member of the IEEE.