HoWiES: A Holistic Approach to ZigBee Assisted WiFi Energy Savings in Mobile Devices

Yifan Zhang and Qun Li
Department of Computer Science, The College of William and Mary, USA

***Abstract*—We propose 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 at 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%, 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% of energy consumed in scanning state and standby state respectively.**

I. Introduction

WiFi radio interface in mobile devices is attracting an increasing amount of applications ranging from mobile social networking [1]–[3] to mobile localization [4]–[6]. However, WiFi interface consumes a considerable amount of power when it is active, 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 the lack of WiFi coverage in many places. *Second*, during PSM (Power Save Mode) 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. Recent works [7], [8] show that users usually leave their smartphones idle for most of the time. The long idle time contributes to a non-negligible amount of WiFi energy consumption. *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 show that the power consumptions of WiFi scanning and PSM standby in a Samsung Galaxy S2 smartphone account for 65% and 11% of the entire system power consumption respectively, and recent works [9], [10] show that the wakeup contentions could cause up to four times more power consumption. To reduce these energy consumptions, we propose that the operations of a WiFi radio in those scenarios can be delegated 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 are reduced to the reasonably low power consumptions on ZigBee radio.

ZigBee radio (i.e., IEEE 802.15.4 [11] compliant radio) is designed for low power communication working on 2.4GHz ISM band, which coincides with most of the WiFi standards. Recently, more and more attempts have been made by both industry and academia to integrate ZigBee radios with smartphones for smart home applications [12], health monitoring [13], and energy savings [14], [15]. 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 [16], 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 uses ZigBee radio to wake up a WiFi interface when it detects that a WiFi network is available (during the network scanning) or that the AP would intend to communicate with the device (during PSM standby). Given that it is not possible for ZigBee to decode WiFi packets directly, it is necessary to build a channel that a WiFi AP can encode information so that ZigBee radio would understand. Fortunately, the same frequency band occupied by both ZigBee and WiFi enables ZigBee radio to sample background energy during WiFi transmission. This side channel, albeit with limited bandwidth, is sufficient for wireless AP to transmit information to ZigBee radios in mobile devices. We demonstrate that, by using a simple coding scheme, our system can create a communication channel in which an AP can convey thousands of messages to ZigBee radios. When ZigBee decodes the information transmitted through the side channel, it can either ignore the message if it is not for the device, or wake up the WiFi radio for communication. This 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 at the three significant WiFi energy saving opportunities in WiFi scanning, standby and wakeup respectively.

Our work is inspired by Esense [17], 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 each character in the alphabet can represent a different piece of information conveyed from WiFi radios to ZigBee radios. The major differences between our work and Esense, which are also the contributions of this work, 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. Our implemented system is able to reliably convey 2744 different messages from WiFi to ZigBee, while this number of Esense is 100. However, it is a challenging open problem 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.

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%, and our energy saving protocols, while maintaining the comparable wakeup delay to that of the standard 802.11 PSM, save 88% and 85% of energy consumed in WiFi scanning and standby respectively.

II. Related Work

Energy saving in WiFi scanning. To save the energy spent in scanning WiFi networks, several projects have considered, without turning on WiFi radios, predicting WiFi networks availability by using different context information [18], tracking and learning user movements [19], or collecting information about bluetooth devices and cell towers [20]. Similar to our solution, ZiFi [14] discovers WiFi networks with the assistance of ZigBee radios. The idea of ZiFi is using ZigBee to detect WiFi beacon patterns, which indicate the existence of WiFi networks. Our solution takes a different approach: we enable APs to advertise themselves by broadcasting messages that are understandable by ZigBee radios. Thus, an advantage of our solution is that with HoWiES, mobile devices are able to selectively wake up and associate to the APs.

TABLE I: 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%

Energy saving in WiFi standby. To save the energy spent in WiFi standby, researchers have proposed to turn off WiFi radios when they are idle, and wake them up through a low-power non-WiFi channel when there are incoming WiFi activities. Wake-on-wireless [21] establishes the low-power channel by attaching a additional device to both APs and WiFi clients. Cell2Notify [22] considers using cellular channel to wakeup WiFi radios for VOIP calls. In our system, we establish the low-power channel directly between APs and devices' ZigBee radios through which APs can wake up standby devices selectively.

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 [9] or multiple interfering APs [10]. In our system, 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.

III. Motivation and background

This work is motivated by the following observations made from our experimental measurements and investigations of current WiFi energy saving research literatures.

A. WiFi energy saving opportunities

We observe that there are multiple significant energy saving opportunities for WiFi stations (i.e., mobile devices operating as stations in a infrastructure WiFi network as specified in the IEEE 802.11 standards.) in several of their working states, which are detailed as follows.

Opportunity 1 - scanning state: The first significant WiFi energy opportunity lies in the scanning state. Stations in scanning state constantly iterate through all the channels to search available WiFi networks. We have measured the system power consumption 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 I), we can see that about 65% and 12% of the system power consumption are spent in WiFi scanning for the Galaxy S2 smartphone and the Lenovo T400 laptop respectively. Moreover, recent research shows that people spend only half of their daily life in areas with WiFi signal coverages [18], which means their WiFi devices would spend about 12 hours a day in the high-power scanning state if they do not turn off WiFi radio when they are

TABLE II: 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%

outside of WiFi coverages. Therefore, we are motivated to find an energy efficient way for mobile devices to discover WiFi networks instead of using power-hungry WiFi radios.

Opportunity 2 - standby state: The power management mode of WiFi stations can be either CAM (Constantly Awake Mode) or PSM (Power Save Mode). The difference between these two modes lies in when WiFi stations are in standby: a CAM station keeps its WiFi radio on all the time; a PSM station puts its WiFi radio into sleep (i.e., stay in a low-power state) for most of the time when there is no traffic, and periodically wakes up the radio to receive and check AP beacons, through which the AP informs the PSM stations about their packets buffered at the AP.

Table II 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 the device drivers. The Galaxy S2 smartphone consumes 33 mW more power, which accounts for about 11% of the overall system power, in the WiFi standby state than when the WiFi radio is turned off. This power overhead mainly comes from the periodic wakeup to check beacons, because when we increased the smartphone's wakeup interval, the power overhead decreased accordingly. The T400 laptop also consumes about 10 percent of its system power in the standby state. Recent works [7], [8] show that smartphone users usually leave their phones idle for most of the time, which makes the standby power consumption of WiFi radios even salient regarding saving energies for mobile devices. Ideally, WiFi radios should sleep without periodic wakeup or be completely turned off as long as there is no WiFi activities. Meanwhile, it must be possible to wake up the WiFi radios if there are incoming packets for them.

Opportunity 3 - energy waste due to wakeup contention: When multiple PSM stations working at the same channel and associated either with the same AP [9] or with multiple co-located APs [10], are waken up to receive buffered packets at the same time, the contention between these stations will make them stay awake but without performing any communication tasks, which further causes about up to 4 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.

B. ZigBee radio assisted WiFi energy savings

Compared with WiFi radios, ZigBee radios are more power efficient. Table III lists the power consumptions we measured of ZigBee radio CC2420 and WiFi radio BCM4330 in different operating modes. Since ZigBee is able to work at the same frequency band as WiFi while consumes significantly less energy, it would provide great assistance in saving WiFi energy for mobile devices if we could make ZigBee radios communicate with WiFi radios. Esense [17] is the first effort to enable

TABLE III: 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

communications between a WiFi radio and a ZigBee radio. The idea is using ZigBee radio to continuously sample the background energy in the air. Once there is a WiFi packet being transmitted, the sampling ZigBee radio will generate several consecutive samples whose energy readings are above a certain threshold, which we call *positive sample*s. Esense studies how the number of consecutive positive samples (denoted as $\#^{+}_{consec}$) distributes when sampling WiFi packets replayed from several public WiFi traces. Esense proposes that each of those rarely occurring $\#^{+}_{consec}$ when sampling the public WiFi traces can be used to convey a certain message from WiFi to ZigBee. The experimental results of Esense show that it is able to deliver up to 100 different messages from WiFi to ZigBee.

The message capacity achieved by Esense is far from enough for being applicable to WiFi energy savings in mobile devices, since there could be up to 2007 stations associated with an AP [23]. Therefore, we are motivated to study how to extend the WiFi-ZigBee message capacity by using combinations of different $\#^{+}_{consec}$ to represent a message. Based on our new WiFi-ZigBee message delivery scheme, we design and implement three protocols that exploit the three opportunities to save WiFi energies for mobile devices.

IV. System Design

A. WiFi-ZigBee message delivery scheme

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.

Alphabet construction. The alphabet $\mathcal{A}$ is a set of b packet sizes: $\mathcal{A} = \{S_1, \cdots, S_b\}$, 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 make message packets be distinguishable from normal WiFi packets. To this end, we carefully choose the predefined sizes for message packets and select the message packets transmission rate such that the air time of a message packet is longer than those of normal WiFi packets.

To study the air times of normal WiFi packets, we deployed WiFi sniffers in our office building and the university's library, both of which are heavy WiFi usage spots, and sniffed WiFi packets for three days. 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). This led to another observation that over 95% of all the sniffed packets had an air time less than 1 millisecond. Therefore, we ensure the air time of a message packet to be longer than those of normal WiFi packets by selecting 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 detail our choices of the predefined packet sizes for the alphabet later in Section V.

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

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

where b is the size of the alphabet $\mathcal{A}$, p_i represents the i-th of the l message packets and $I_{p_i,\mathcal{A}}$ is the index of the packet p_i's size in the alphabet $\mathcal{A}$, for example, $I_{i,\mathcal{A}} = j$ if the size of packet p_i is S_j $(S_j \in \mathcal{A}, 1 \leq j \leq b)$. Then the *capacity* of a message delivery scheme, which is the total amount of numbers that the scheme can encode, 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 3 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 3 packets with 200B, 100B and 200B respectively, essentially it sends out 3 digits with values of 1, 0 and 1 in that order, and the message is interpreted as number 5 (i.e., $1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 = 5$).

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

Algorithm 1: WiFi-ZigBee message detection/decoding

```
   Data: R, l, b, H, E and A = {S_1, ··· , 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;
```

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 come from a message packet or from a normal packet. If they come 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 come 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).

Self-correcting message encoding/decoding. Without considering hidden terminals' effects, which is a case we will

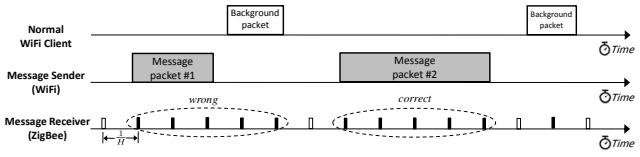

Fig. 1: An example of background packet interference.

discuss at the end of this section, message packets will not overlap with normal packets in time domain because of the 802.11 CSMA/CA scheme. However, since ZigBee radio cannot sample with an interval smaller than the IEEE 802.11 short interframe space (SIFS). it is possible that a ZigBee radio obtains the same number of energy samples for two message packets with different sizes if there are two packets sent with an interval smaller than the ZigBee radio's sampling interval. Figure 1 shows an example: A WiFi radio sends out two WiFi-ZigBee message packets on which a ZigBee radio normally will generate 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 2 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 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 $\mathcal{A}$=$\{S_1,\cdots,S_b\}$ is divided into p sub-alphabets as $\mathcal{A}_1$=$\{S_1,S_{p+1},S_{2p+1}\cdots\}$, $\mathcal{A}_2$=$\{S_2,S_{p+2},S_{2p+2},\cdots\}$, $\cdots$, $\mathcal{A}_p$=$\{S_p,S_{2p},\cdots S_b\}$. 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 3 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 300B, 200B and 100B 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 200B) was affected by background interference. In this case, the ZigBee radio replaces the size 200B in $\mathcal{A}_2$ with size 100B in $\mathcal{A}_1$, and reports the message value as $1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 = 1$.

B. HoWiES energy saving protocols

Based on the WiFi-ZigBee message delivery scheme, we design three HoWiES energy saving protocols that save energy consumed in WiFi scanning, standby and wakeup respectively. At the mobile device side, three components relate 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 to ZigBee as specified in the following protocol descriptions.

HoWiES scanning and association. Figure 2 shows the HoWiES scanning and association protocol. 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 (in the HoWiES scanning protocol) and to indicate buffered broadcast/multicast packets (in the HoWiES wakeup protocol). During the scanning process, a HoWiES client turn on its WiFi radio and associate to an AP based on the numbers encoded in the WiFi-ZigBee messages received. For example, a system operator can configure open APs to encode "1" in their network advertisement WiFi-ZigBee messages, and configure encrypted APs to encode "2". Then mobile devices can selectively turn on their WiFi radios based on whether the encountered networks is encrypted. 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, 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

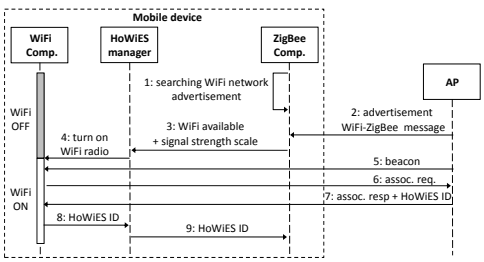

Fig. 2: HoWiES scanning and association operations.

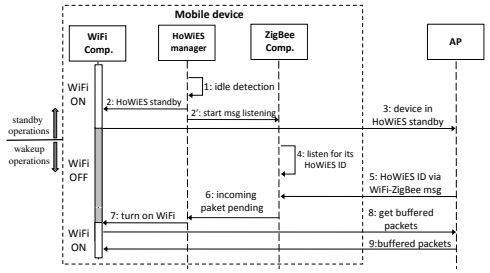

Fig. 3: HoWiES standby and wakeup operations.

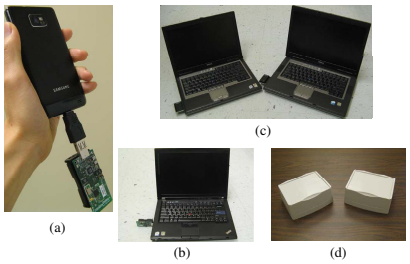

Fig. 4: HoWiES system implementation.

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).

HoWiES standby. This protocol puts mobile devices into HoWiES standby by turning off the WiFi radio and informing AP about the status change on the mobile devices. The upper half of Figure 3 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 certain 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.

HoWiES wakeup. The bottom half of Figure 3 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).

Since APs wake up its HoWiES standby clients one at a time, this approach naturally solves the wakeup contention problem causing by waking up multiple WiFi clients associated with the same AP. However, if multiple interfering APs (i.e., APs that can hear each other) 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 [10]. To solve the problem, 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}. The value of T_{dur} is determined in the same way as the length of a fair share is determined in [10]. Interfering APs coordinate their beacon periods [10] to ensure their wakeup periods do not overlap with each other.

C. Discussions

Dealing with hidden terminals. In 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 be concatenated in time domain at a certain place between them. In this case, the concatenated packet 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.

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 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.

V. System Implementation

We have implemented the HoWiES system with the devices shown in Figure 4. The system has two types of entities: HoWiES clients/APs. HoWiES clients are implemented in two mobile platforms: a smartphone platform (Samsung Galaxy S2) and a laptop platform (Lenovo T400). 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) and a standalone AP platform (Wiligear WBD-500 integrated radio platform).

A. HoWiES client

A HoWiES client has three major components: the WiFi component (consisting of the WiFi radio and the WiFi driver),

the ZigBee component (consisting of the CC2420 ZigBee radio and the message detection/decoding TinyOS module) and the HoWiES manager.

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.

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).

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 overheads in Galaxy S2 (33 mW). To solve this issue, we adopted a solution similar to [24], where the sensor is put to sleep periodically for energy savings. We have reduced the energy sampling power consumption of TelosB mote to 5 mW by duty cycling the ZigBee radio. In our implementation, a ZigBee radio samples background energy only during the wakeup period of the AP that its hosting device is associated with. 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). 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.

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.

B. HoWiES AP

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 experimentally quantified how stable the CC2420 radio generates energy samples in sampling packets with a fixed length. We found that the CC2420 radio we used can produces 4 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 for the alphabet: $\mathcal{A} = \{300, 390, \cdots, 1470\}$. 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" sub-section.

TABLE IV: Reliability and accuracy of the implemented 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%

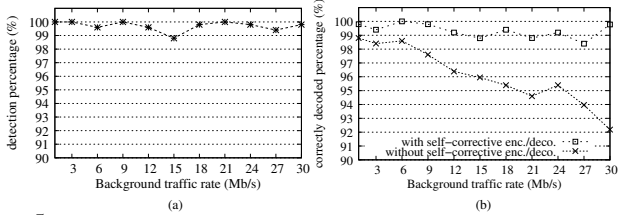

Fig. 5: Reliability and accuracy of the implemented WiFi-ZigBee message delivery scheme in the controlled experiment.

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

VI. System Evaluation

A. WiFi-ZigBee message delivery

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.

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 2744, 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 IV shows the results. For the total 20,000 WiFi-ZigBee messages,

99.5% of them were detected by the HoWiES client. Within all the detected messages, the HoWiES client correctly decoded 96.6% 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%. 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.

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 1500 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. Figure 5 (a) presents the message delivery's reliability performance. For all the tested background traffic bandwidths, our scheme can correctly detect at least 99% of them. Figure 5 (b) 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%. 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%.

Message delivery overheads. To evaluate the message delivery overheads imposed on network throughput, we tested the iperf UDP bandwidth 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, $\cdots$, 60, 80 and 100. Figure 6 (a) 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

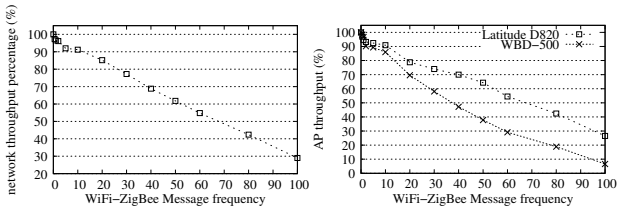

Fig. 6: HoWiES WiFi-ZigBee message delivery overheads.

on network throughput. With the sending frequencies of 5 and 10, the tested iperf connection still had 90% of its bandwidth. Then the network bandwidth decreased approximately in linear as the message sending frequency increased.

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. Figure 6 (b) 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.

B. Energy gain achieved by the energy saving protocols

Power measurement setup and methodology. To measure the power consumption in the T400 laptop, we use the smart battery interface come with the operating system. According to [25], the smart battery interface is highly accurate when the battery interface reading rate is low. Since we are only interested in long term energy consumptions, the smart battery interface satisfies our requirements. To measure the power consumption in the smartphone, we use the Monsoon power monitor [26], which provides accurate power readings for hand-held mobile devices. When we measure the power of a device, we turn 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 measure the baseline system power consumption (i.e., system power consumption without running any WiFi operations). Then we measure the system power when the device is continuously performing the targeted WiFi operation. Finally, the difference between the two values is the power consumption for the WiFi operation.

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 shows that the T400 laptop, the Galaxy S2 smartphone and the HoWiES client spend 1740 mW, 501 mW and 61 mW for WiFi scanning respectively. Figure 7 (a) shows the energy generated by the WiFi scanning operation as the

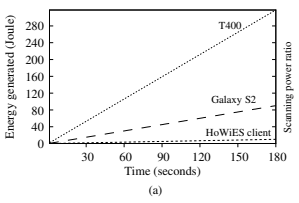

Fig. 7: Energy gain on the WiFi scanning state.

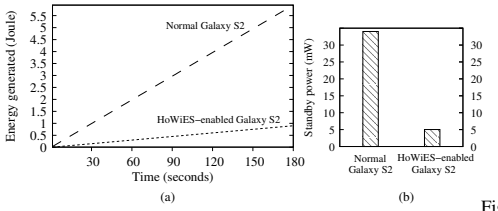

Fig. 8: Energy gain on the WiFi standby state.

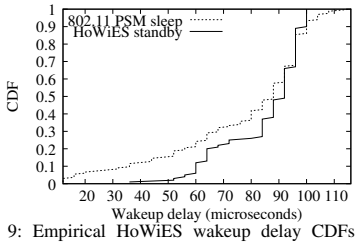

Fig. 9: Empirical HoWiES wakeup delay CDFs of a normal Galaxy S2 and a HoWiES-enabled Galaxy S2.

time elapses in a 3 minutes duration. Figure 7 (b) 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.

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. Our measurement shows that the normal Galaxy S2 and the HoWiES-enabled Galaxy S2 consumes 33 mW and 5 mW in the standby state respectively. Figure 8 (a) shows the energy generated during standby as the time elapses in a 3 minutes duration. Figure 8 (b) compares the standby power 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.

C. HoWiES wakeup delay

We evaluate 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. On the clients side, the wakeup interval of the normal Galaxy S2 is set to a beacon period, which is the default setting used by the WiFi driver. For the HoWiES-enabled Galaxy S2, it goes to sleeping state once it enters HoWiES standby, and keeps sleeping until it is waken up by a WiFi-ZigBee message. Figure 9 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 is still room to improve the wakeup latency in our implementation. For example, currently an AP is using a user space program to transmit message packets. This will 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.

VII. 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.

Acknowledgment

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grants CNS-1117412 and CA-REER Award CNS-0747108.

References

[1] Z. Yang, B. Zhang, J. Dai, A. C. Champion, D. Xuan, and D. Li, "E-smalltalker: A distributed mobile system for social networking in physical proximity," in *ICDCS*, 2010.
[2] J. Teng, B. Zhang, X. Li, X. Bai, and D. Xuan, "E-shadow: Lubricating social interaction using mobile phones," in *ICDCS*, 2011.
[3] W. Wei, F. Xu, and Q. Li, "Mobishare: Flexible privacy-preserving location sharing in mobile online social networks," in *INFOCOM*, 2012.
[4] S. Sen, R. R. Choudhury, and S. Nelakuditi, "Spinloc: spin once to know your location," in *HotMobile*, 2012.
[5] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "Spot localization using phy layer information," in *MobiSys*, 2012.
[6] H. Han, F. Xu, C. C. Tan, Y. Zhang, and Q. Li, "Defending against vehicular rogue aps," in *INFOCOM*, 2011.
[7] H. Falaki et al., "Diversity in smartphone usage," in *MobiSys*, 2010.
[8] A. Shye et al., "Characterizing and modeling user activity on smartphones: summary," in *SIGMETRICS*, 2010.
[9] E. Rozner et al., "NAPman: network-assisted power management for wifi devices," in *MobiSys*, 2010.
[10] J. Manweiler and R. R. Choudhury, "Avoiding the rush hours: WiFi energy management via traffic isolation," in *MobiSys*, 2011.
[11] IEEE-SA, *IEEE Std 802.15.4-2006*.
[12] Texas Instruments News Center, *TI demonstrates world's first Android development platform bringing ZigBee and ZigBee RF4CE to Smartphones and Tablets*, 2011.
[13] M. Keally et al., "PBN: Towards Practical Acitvity Recognition Using Smartphone-Based Body Sensor Networks," in *SenSys*, 2011.
[14] R. Zhou et al., "ZiFi: wireless LAN discovery via ZigBee interference signatures," in *MobiCom*, 2010.
[15] T. Jin et al., "WiZi-Cloud: Application-transparent dual ZigBee-WiFi radios for low power Internet access," in *Infocom*, 2011.
[16] TazTag, *TPH-ONE*, 2012, http://www.taztag.com/PR_TPH-ONE_revMD_V4.pdf.
[17] K. Chebrolu and A. Dhekne, "Esense: communication through energy sensing," in *MOBICOM*, 2009.
[18] A. Rahmati and L. Zhong, "Context-for-wireless: context-sensitive energy-efficient wireless data transfer," in *MobiSys*, 2007.
[19] A. J. Nicholson and B. D. Noble, "BreadCrumbs: forecasting mobile connectivity," in *MobiCom*, 2008.
[20] G. Ananthanarayanan and I. Stoica, "Blue-Fi: enhancing Wi-Fi performance using bluetooth signals," in *MobiSys*, 2009.
[21] E. Shih et al., "Wake on wireless: : an event driven energy saving strategy for battery operated devices," in *MobiCom*, 2002.
[22] Y. Agarwal et al., "Wireless wakeups revisited: energy management for VOIP over WiFi smartphones," in *MobiSys*, 2007.
[23] IEEE-SA, *IEEE Std 802.11-2007*.
[24] S. Ren, Q. Li, H. Wang, X. Chen, and X. Zhang, "Analyzing Object Detection Quality Under Probabilistic Coverage in Sensor Networks," in *IWQoS*, 2005.
[25] M. Dong and L. Zhong, "Self-constructive high-rate system energy modeling for battery-powered mobile systems," in *MobiSys*, 2011.
[26] Monsoon Solutions, www.msoon.com/LabEquipment/PowerMonitor.