ZiFi: Wireless LAN Discovery via ZigBee Interference Signatures

Ruogu Zhou[1], Yongping Xiong[2], Guoliang Xing[1]*, Limin Sun[3], Jian Ma[4]
[1]Department of Computer Science and Engineering, Michigan State University, USA
[2]Institute of Computing Technology, [3]Institue of Software, Chinese Academy of Sciences, China
[4]Nokia Research Center, Beijing, China
{zhouruog,glxing}@cse.msu.edu; {xyp,sunlimin}@is.iscas.ac.cn; jian.j.ma@nokia.com

ABSTRACT

WiFi networks have enjoyed an unprecedent penetration rate in recent years. However, due to the limited coverage, existing WiFi infrastructure only provides intermittent connectivity for mobile users. Once leaving the current network coverage, WiFi clients must actively discover new WiFi access points (APs), which wastes the precious energy of mobile devices. Although several solutions have been proposed to address this issue, they either require significant modifications to existing network infrastructures or rely on context information that is not available in unknown environments. In this work, we develop a system called *ZiFi* that utilizes ZigBee radios to identify the existence of WiFi networks through unique interference signatures generated by WiFi beacons. We develop a new digital signal processing algorithm called Common Multiple Folding (CMF) that accurately amplifies periodic beacons in WiFi interference signals. ZiFi also adopts a constant false alarm rate (CFAR) detector that can minimize the false negative (FN) rate of WiFi beacon detection while satisfying the user-specified upper bound on false positive (FP) rate. We have implemented ZiFi on two platforms, a Linux netbook integrating a TelosB mote through the USB interface, and a Nokia N73 smartphone integrating a ZigBee card through the miniSD interface. Our experiments show that, under typical settings, ZiFi can detect WiFi APs with high accuracy ($< 5\%$ total FP and FN rate), short delay (~ 780 *ms*), and little computation overhead.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms, Design, Experimentation, Measurement, Performance

*Correspondence author

MobiCom'10, September 20–24, 2010, Chicago, Illinois, USA.

Keywords

WiFi Network Discovery, Digital Signal Processing, Stochastic Signal Detection, Interference, Power Management

1. INTRODUCTION

In recent years, 802.11-based wireless LANs, also known as WiFi networks, have enjoyed an unprecedent penetration rate. In particular, they are increasing deployed to provide Internet access in mobile environments. However, due to the limited coverage, existing WiFi infrastructure is only capable of providing intermittent connectivity for the users with high mobility. WiFi-enabled devices (e.g., laptops, PDAs, and smartphones) must actively discover new WiFi access points (APs) once they leave the coverage of current network. However, this approach wastes the precious energy of mobile devices due to excessive listening and scanning operations of WiFi network interface cards (NICs).

Several solutions have been proposed to address the aforementioned issue. The first approach utilizes a secondary low-power radio that communicates with peer radios on WiFi APs to find connectivity opportunities or reduce the energy consumption of data transfers [24] [25] [12] [11] [20] [17] [15]. However, this approach requires significant modifications to existing network infrastructures. The second approach predicts the availability of WiFi based on *context* information. Cellular cell-tower information [13] or together with Bluetooth contact-patterns [22] have been used to improve WiFi prediction accuracy. However, such a context-aware approach requires extensive training based on historical information and hence is not feasible in unknown environments.

In this work, we develop a system called *ZiFi* for discovering the availability of WiFi coverage for mobile users. The design of ZiFi is motivated by the fact that low-power radios such as ZigBee and Bluetooth often not only physically collocate with WiFi NICs but also share the same open radio frequency band with them. Leveraging the inter-platform interference caused by such coexistence, ZiFi enables ZigBee radios to identify the unique interference signatures generated by WiFi signals. As a result, a mobile device can use a ZigBee radio to detect the existence of WiFi APs in a purely passive manner, and only wakes up the WiFi NIC when WiFi connectivity is available. To capture WiFi interference signatures, ZiFi utilizes the received signal strength (RSS) indicator available on ZigBee-compliant radios. However, we observed that the statistics of WiFi RSS samples, such as power magnitude, time duration, and inter-arrival gap, exhibit surprising resemblance with those of other RF sources, and hence provide little hint about the existence of

WiFi. Motivated by this observation, ZiFi is designed to search for 802.11 *beacon frames* in RSS samples. Periodic beacon broadcasting is mandatory in WiFi infrastructure networks and hence provides a reliable means to indicate WiFi coverage. However, beacons are extremely scarce in normal WiFi traffic as hundreds of data frames are likely transmitted between two beacon instances. Without being able to decode incoming signals, finding beacon frames in RSS samples is like finding a needle in a haystack. To address this challenge, ZiFi adopts novel digital signal processing (DSP) and stochastic signal detection techniques to reliably identify the periodic interference patterns caused by WiFi beacon frames.

We envision the approach of ZiFi to be increasingly feasible as more mobile devices are equipped with both low-power and high-power NICs that work in the same open radio spectrum. For instance, numerous ZigBee modules [10] have USB interface and hence can be easily connected to WiFi-enabled laptops. Several cell phone vendors (e.g., Nokia and Pantech & Curitel) also provide smartphones [7] with built-in ZigBee interface or ZigBee modules [29] that can be connected to smartphones (e.g., through miniSD interface). ZiFi can also be easily implemented on other platforms (e.g., some Bluetooth radios [1] [8]) that offer the RSS sampling interface. We make the following key contributions in this paper.

1. We develop a novel DSP algorithm called Common Multiple Folding (CMF) that amplifies unknown periodic signals in RSS samples. A key advantage of CMF is that it can minimize the computational cost of processing unknown signals whose possible periods lie in a wide range. We then develop a constant false alarm rate (CFAR) detector that can minimize the false negative (FN) rate of classifying periodic signals as 802.11 beacons while satisfying the user-specified upper bound on false positive (FP) rate.
2. We present an analytical framework that characterizes the detection performance of ZiFi by the FN and FP rates. Our results not only guide the selection of optimal detection thresholds for beacon detector but also allow to predict the opportunities of WiFi coverage based on empirically measured channel parameters.
3. We have implemented ZiFi on two platforms, a Linux netbook integrating a TelosB mote through the USB interface, and a Nokia N73 smartphone [29] integrating a ZigBee card through the miniSD interface. Our extensive experiments on a testbed consisting of wireless routers, netbooks, smartphones, and TelosB motes show that, under typical settings, ZiFi can detect WiFi APs with high accuracy ($< 5\%$ total FP and FN rate), short delay (~ 780 *ms*), and little computation overhead.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 describes the system architecture of ZiFi. Section 4 and Section 5 present the design of ZiFi and the framework for detection performance analysis. Section 6 presents the experimental results and Section 7 concludes the paper.

2. RELATED WORK

The idea of waking up high-power radio using a secondary low-power radio was first proposed in Wake-on-Wireless [24]. Several recent systems including On-Demand-Paging [12], Cell2notify [11], and CoolSpots [20], also propose to use a secondary radio to either help detect WiFi signal or reduce the energy consumption of WiFi data transfers. Wake-on-WLAN [17], S-WOW [18], and Esense [15] allow ZigBee and WiFi radios to communicate through sensing specially designed codes. However, the above solutions suffer from at least one of the following issues. (1) They assume a "cooperative" setting where substantial software and/or hardware modifications to existing network infrastructures can be made, which hinders their wide deployment. Esense [15] requires WiFi APs to transmit special codes not defined in 802.11 standard to communicate with the ZigBee radio on WiFi clients. In Cell2notify [11], VoIP servers are modified to notify clients of incoming VoIP calls on cellular interface and carry out the calls on WiFi interface. Wake-on-Wireless [24] uses a proxy server to receive incoming Internet traffic on WiFi interface and wake up WiFi client on the low-power interface. CoolSpots [20] selects most power-efficient interface for communication when both APs and clients are equipped with dual Bluetooth/WiFi interfaces. (2) As the secondary low-power radio has significantly shorter communication range, the existing solutions often rely on additional proxy servers to achieve satisfactory network discovery range. In contrast to these solutions, ZiFi completely relies on the ZigBee interface on WiFi clients to detect the existence of WiFi APs and requires no modification to WLAN infrastructure. Moreover, ZiFi detects WiFi signal by passively sensing its energy, which ensures a similar detection range as WiFi interface.

There exist several portable USB-based spectrum analyzers [6] that are designed to be used with mobile devices for scanning spectrum usage in 2.4 GHz band. As they all provide RSS sampling interface, ZiFi can be easily implemented without requiring a secondary ZigBee radio. Moreover, they often provide desirable features such as high-resolution and enhanced amplitude range which would potentially improve the accuracy of ZiFi. There also exist portable spectrum scanners [5], often referred to as *WiFi detectors*, which are specially designed to find WiFi signal. They usually work in standalone mode but may also be modified to wake up WiFi NICs on mobile devices [25]. Unlike ZiFi that only detects the existence of APs, WiFi detectors often provide more information about of APs, e.g., SSIDs and security settings. However, these features require the use of 802.11 radios and hence come with high power consumption. Moreover, as specialized hardware, they have not gained popularity in WiFi community.

In [19], BreadCrumbs is proposed to build the mobility model of a mobile device by tracking its movement from GPS and use the model to predict the connectivity opportunities. Cellular cell-tower information or together with Bluetooth contact-patterns have been used in [13] [22] to predict the existence of WiFi. A key drawback of these context-aware approaches is that they rely on historical information and hence cannot be applied to unknown environments. Moreover, they often require extensive offline training in order to achieve satisfactory runtime prediction accuracy. In contrast, ZiFi detects WiFi coverage by in-situ processing of signals transmitted by APs, which does not need offline training or collection of context information.

3. SYSTEM ARCHITECTURE

ZiFi is designed for two different types of platforms to discover WiFi APs: the platforms (e.g., smartphones) that have both built-in ZigBee and WiFi interfaces, and the platforms that can connect a WiFi node with an external ZigBee node. Fig. 1 shows two platforms of the second type on which ZiFi has been implemented. Fig. 1 (a) is a Nokia N73 mobile phone that integrates a ZigBee module through the miniSD interface. Fig. 1 (b) is an ASUS Linux netbook that connects a TelosB mote (equipped with a ZigBee-compliant CC2420 radio [21]) through the USB interface.

(a) (b)

Figure 1: Two different platforms on which ZiFi has been implemented. (a) is a Nokia N73 mobile phone and a ZigBee card integrated via the miniSD interface. The top of figure shows front and back views of the ZigBee module and its main components. (b) is an ASUS netbook and TelosB mote integrated via the USB interface.

Fig. 2 shows the system architecture of ZiFi. The *RSS sampler* reads the built-in received signal strength indicator (RSSI) register of ZigBee radio at a designated frequency. The RSS samples are then processed by a *RSS shaper* that adjusts the RSS values to mitigate noise (e.g., the data frames) in the beacon detection. The shaped RSS samples are then processed by the Common Multiple Folding (CMF) algorithm. CMF is a digital signal processing algorithm that amplifies the periodic signals in RSS samples. A key advantage of CMF is that it can minimize the cost of amplifying unknown signals whose possible periods lie in a wide range. The amplified RSS samples are fed into a *constant false alarm rate (CFAR) [27] beacon detector* that classifies a periodic signal as genuine WiFi beacons if its amplitude exceeds a threshold. By adopting a theoretically derived threshold, the beacon detector can minimize the false negative (FN) rate while satisfying the user-specified upper bound on false positive (FP) rate. Finally, if WiFi beacons are detected, the *radio controller* turns on the WiFi NIC. In this paper, we also present an analytical framework that models the FN and FP rates of beacon detection based on the utilization ratio of wireless channel. The utilization ratio is measured from RSS samples by the *channel profiler*. The analytical FN and FP models guide the selection of optimal detection thresholds for ZiFi's beacon detector.

As discussed above, ZiFi utilizes energy sensing through the RSSI of ZigBee radio to detect the existence of WiFi APs. ZiFi can be easily implemented on other radio platforms that provide the RSSI interface. For instance, a few existing Bluetooth radios [1] [8] provide RSSI although it is not a mandatory feature in Bluetooth standard.

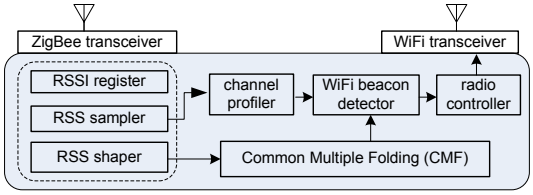


Figure 2: System architecture of ZiFi.

4. DESIGN OF ZI-FI

4.1 Background on 802.11 and 802.15.4

ZiFi is designed for unlicensed 2.4 GHz ISM band that has been adopted by most 802.15.4 and 802.11 networks. When operating in the same or adjacent frequencies, 802.11 and 802.15.4 radios can interfere with each other. In particular, only 4 among total 16 channels of 802.15.4 are orthogonal with the channels of 802.11. Moreover, several commodity 802.15.4 radios (e.g., CC2420 [2]) have programmable channel center frequencies within the 2.4 GHz band. Thus, there is abundant opportunities for 802.15.4 radios to sense the existence of 802.11 transmissions. A received signal strength indication (RSSI) register is typically provided by commodity 802.15.4 radios, which samples the channel every 32 *us* and the value (dBm) is averaged for 8 symbols (128 *us*). The minimum sensitivity of RSSI is above -85 dBm. ZiFi senses 802.11 transmissions by sampling the RSSI register of 802.15.4 radio, and searches for periodic beacon signals in the RSSI samples. Periodic beacon broadcasting is mandatory in 802.11 infrastructure networks. The typical length of beacon frame ranges from 80 to 200 bytes depending on the amount of management information (e.g., supported rates and security settings) it carries.

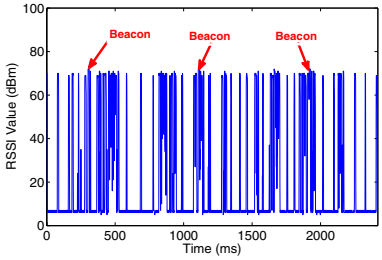


Figure 3: The RSS samples collected by a CC2420 radio when a nearby 802.11 AP actively transmits. The RSS samples of 802.11 beacons are labeled.

4.2 Exploring the Design Space

ZiFi is designed to discover WiFi APs in a variety of scenarios including mobile and delay tolerant networks. The design objectives of ZiFi include the following: 1) *High accuracy.* We characterize the accuracy of AP detection using *false positive (FP)* and *false negative (FN)* rates. In particular, FPs falsely trigger the wake-up of NICs leading to energy waste while FNs mean the misses of opportunities of WiFi connectivity. 2) *Low delay.* This is of particular importance for mobile environments. For instance, recent war driving experiments in Boston metropolitan area [14] showed that the median duration of WiFi connectivity at vehicular speeds is only 13 seconds. Fast WiFi discovery is thus required to utilize such short connectivity windows. 3) *Low*

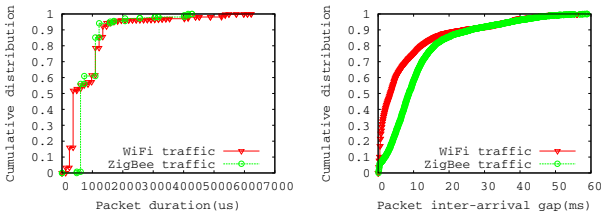


Figure 4: The CDFs of frame sizes and inter-arrival gaps of WiFi and ZigBee traffic. The WiFi trace is collected from an 802.11b WLAN consisting of 3 APs and 8 mobile clients while the ZigBee trace is collected from a 10-node sensor network testbed in our lab.

overhead. Due to the resource constraints of mobile devices, the computation and memory overhead in WiFi discovery must be small.

The design of ZiFi leverages on the fact that, when 802.15.4 radio operates on the same or adjacent channels as 802.11 APs, the signals from 802.11 APs can be sensed through RSSI. Fig. 3 shows the time series of RSS samples gathered from a TelosB mote equipped with ZigBee-compliant CC2420 radio when a nearby 802.11 AP actively transmits. However, as ZigBee radios cannot directly decode 802.11 frames, RSS statistics provide little information about the nature and source of incoming signals. ZiFi addresses this challenge by searching for unique interference signatures in RSS samples.

We now discuss several possible approaches to detecting WiFi signals from RSS samples. First, one can explore the statistics of RSS samples, such as power magnitude, time duration, and inter-arrival gap, to find distinctive features of WiFi traffic. However, these features vary significantly with environments, version of 802.11, and application traffic. Moreover, they may well resemble the interference from other RF sources such as ZigBee nodes or Bluetooth nodes transmitting on overlapping channels. Fig. 4 shows the cumulative distribution of frame sizes and inter-arrival gaps of two WiFi and ZigBee traffic traces. The surprising resemblance between these two traces make it challenging to reliably detect the existence of WiFi networks.

Instead of relying on statistical traffic analysis, ZiFi searches for 802.11 beacon signals from RSS samples. Several properties of 802.11 beacons make them ideal for detection. First, they are broadcast *periodically* by APs and hence lead to periodic traces in RSS samples. Second, beacons are typically broadcast at the *lowest* modulation rate, which makes it easier to capture by the RSSI register of low-rate ZigBee radios. However, a key challenge is that, hundreds of data frames are typically transmitted between two beacon frames, causing heavy noise in RSS samples. Moreover, the RSS time duration provides little hint as there is a large overlap between the in-air times of data and beacon frames. As RSS samples are time domain digital signals, several signal processing techniques such as *Fast Fourier Transformation (FFT)* and *Autocorrelation*, can be used to detect periodic patterns from noisy measurement. However, our experimental results in Section 6 show that their performance are highly sensitive to the intensity of noise, making them ill-suited for identifying beacons in moderate to high traffic workload. Moreover, both of them impose high computation overhead for mobile devices like smartphones. ZiFi adopts a novel digital signal processing algorithm called Common Multiple Folding (CMF) that can reliably identify periodic WiFi beacons at small delay and overhead. In the following, we discuss the details of the design of ZiFi.

4.3 RSS Sampling and Shaping

The RSS sampler of ZiFi reads the RSSI register of ZigBee radio every T *us* for total D *us*. T and D are referred to as RSS *sampling period* and *sampling window size*, respectively. The sampling period should be short enough to capture the transmission of 802.11 beacon frames. However, a short sampling period leads to high overhead for resource-constrained ZigBee nodes. We have carefully analyzed the formats and transmission rates of beacons defined by different 802.11 versions, and determined that a sampling period of 122 *us* allows to capture at least two RSS samples for each beacon frame transmission. The details of our analysis are omitted due to space limitation and can be found in [30]. In our implementation on TelosB motes, 122 *us* take 4 ticks of the on-board clock.

After enough RSS samples are collected, the RSS shaper adjusts the power magnitude of them to mitigate the noise in the following beacon detection stage. The shaper applies the following two criteria to RSS samples in order: 1) The magnitude of an RSS sample is set to zero if it is below -90 dBm because, even if the sample contains beacon, a low RSS indicates poor signal quality from the AP and low probability of successful client association. 2) The magnitude of all remaining RSS samples are set to 1 dBm. 3) The magnitude of S consecutive non-zero samples will be set to zero if $S \notin [s_1, s_2]$. A cluster of such samples is typically generated by WiFi data traffic. We now discuss how to determine s_1 and s_2 based on beacon size and 802.11 transmission rates. The 802.11 beacon frame has a size between 80 and 200 bytes. When possible 802.11 transmission rates are considered, the in-air time of a beacon frame is from 256 to 1720 *us* [30], which corresponds to an RSS sample count in $[\frac{256}{T}, \frac{1720}{T}]$, where T is the RSS sampling period. Therefore, a number of consecutive samples can be removed if the count lies outside this range. After the above three steps, the magnitude of RSS samples is either 0 or 1 and the number of consecutive non-zero RSS samples is within $[\frac{256}{T}, \frac{1720}{T}]$.

4.4 Common Multiple Folding Algorithm

We have developed a novel digital signal processing algorithm called Common Multiple Folding (CMF) that can identify periodic signals from an RSS series. CMF has several key advantages including high accuracy and low computation/memory overhead. CMF is based on a technique called *folding* that was first used to search pulsar in the radio noise received by a large radio telescope [23,26]. We first briefly describe the basic idea of folding and then discuss the details of CMF.

4.4.1 Basic Idea of Folding

Suppose $\mathcal{R}$ represents the time series of N RSS samples and $\mathcal{R}[i]$ $(i \in [1, N])$ is the RSS magnitude in the ith sampling instance. The objective of folding is to search for a periodic signal with period of P. The series is first divided into smaller sequences with length of P at different starting points (e.g., phases). For each folding operation, the sequences are added together in an element-wise fashion. If the phase of folding happens to align with that of the peri-

odic signal, the magnitude of the sum will be amplified at a period of P while the sum of noise in the series is likely smaller due to their non-periodicity. The sum of folding consists of P elements of $\mathcal{R}$:

$$\mathcal{F}_P[i] = \sum_{j=0}^{\lfloor N/P \rfloor - 1} \mathcal{R}[i + j \cdot P] \quad (1)$$

$\mathcal{F}_P[i]$ is referred to as the ith *folding result* and the maximum is referred to as the *folding peak* of period P. It can be seen that the folding operation requires $N - P$ number of additions. When P is unknown, folding can be performed for each possible period and the maximum folding and P can then be found as the period that yields the maximum folding result. In [26], the fast folding algorithm (FFA) was developed to implement the above approach while reducing the redundant additions in the folding of different periods. However, FFA is mainly designed for searching for non-integer periods between P_0 and $P_0 + 1$ and it requires the ratio N/P_0 to be power of 2. The number of additions required by FFA is $Nlog_2(N/P_0)$.

4.4.2 Common Multiple Folding

A key challenge of searching for WiFi beacons from an RSS series is that the period(s) of beacons is not only unknown but also has a wide range. For instance, the LinkSys WRT54G2 wireless router allows to set any beacon period between 20.48 and 5120 *ms* at a step of 1.024 *ms* although the default setting is 102.4 *ms*. In practical scenarios, the beacon period is typically around 100 $\sim$ 200 *ms*, which still lends to tens of possible beacon settings. As a result, applying folding iteratively for this range incurs high complexity. In particular, the complexity of FFA would be $O(|\mathcal{P}| \cdot N \cdot lg(N))$, where $\mathcal{P}$ is the set of possible beacon periods. As discussed in Section 4.3, N could be large due to the high RSS sampling rate required to capture a beacon transmission.

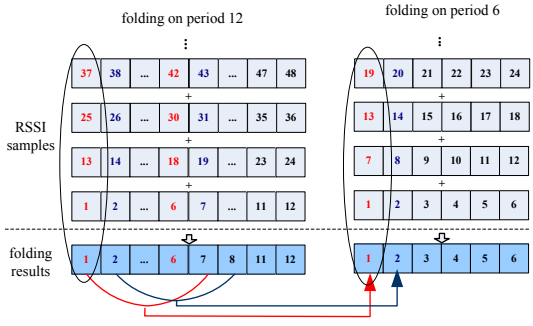


Figure 5: The folding operations for period 12 and 6. The numbers are the indices of raw RSSI samples or folding results. Each element in the folding result of period 12 is computed by summing the RSS samples vertically. Each element in the folding result of period 6 is computed by summing 6 elements of folding result of period 12.

We now present a novel algorithm called *Common Multiple Folding* (CMF) that can *minimize* the total number of additions required to fold on multiple periods. The complexity of CMF is $O(\lg|\mathcal{P}| \cdot lcm(\mathcal{P}) + N)$ where $lcm(\mathcal{P})$ is the least common multiple (LCM) of the numbers in $\mathcal{P}$. The design of CMF is based on the observation that the folding result of period P can be efficiently computed from that of period Q if Q is an integer multiple of P, *i.e.*, $Q \ mod \ P = 0$. Formally, given folding result $\mathcal{F}_Q$, only $Q - P$ additions are needed to obtain $\mathcal{F}_P$. In comparison, total $N - P$ additions are needed to compute $\mathcal{F}_P$ directly from original N RSS samples. For example, Fig. 5 illustrates that the folding result of period 6 can be calculated by an additional folding operation on the result of period 12. For instance, the first element in the folding result of period 6 can be computed by a single addition of the the first and seventh elements in the folding result of period 12, *i.e.*, $\mathcal{F}_6[1] = \mathcal{F}_{12}[1] + \mathcal{F}_{12}[7]$. In total, $12 - 6 = 6$ additions are required to fold on period 6 if the folding results of period 12 are already available. In comparison, total $N - 6$ additions would be needed if the folding is directly applied to the original RSS samples.

Based on the above example, a promising approach to reducing the computational cost is to first fold on the LCM of all periods in $\mathcal{P}$, and then reuse the results to fold on each of the periods. In order to maximize the utility of intermediate folding results, this idea can be applied *recursively* by partitioning $\mathcal{P}$ into subsets and folding on the LCM of all periods in each subset. This process can be naturally encoded by a tree where a node represents a period set and its LCM and all children of the node constitute the partition of the set. We refer to such a tree as *CMF tree*. Fig. 6 shows two CMF trees that differ in how to partition the period set at each node.

Once a CMF tree is constructed from a given period set, folding on all periods in the set can be performed by traversing the tree in the breadth-first order and folding on the LCM of each node. For instance, in the left tree in Fig. 6, one can first fold on 2520 (by N-2520 additions for total N RSSI samples), and then on 72 (by 2520-72=2448 additions) and 70 (by 2520-70=2450 additions) etc., which results in total $N + 2654$ additions. It can be seen that a similar procedure for the right tree in Fig. 6 requires total $N + 2832$ additions. This example shows that the computation cost of a CMF tree depends on how it partitions the period set at each node. An interesting question is how to find the *optimal* CMF tree that yields the least number of additions.

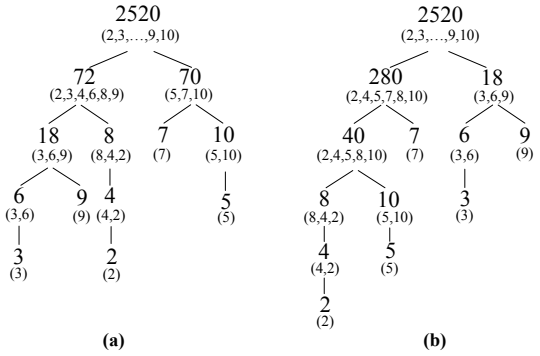


Figure 6: Two CMF Trees for folding on periods from 2 to 10. Each node represents a set of periods and the LCM of the periods. The tree in (a) is optimal and requires $N + 2654$ additions while the tree in (b) requires $N + 2832$ additions where N is the total number of RSSI samples.

Suppose node n is on the CMF tree constructed for period set $\mathcal{P}$ and n is associated with period set $\mathcal{P}_n$. We have proved the following properties of the optimal CMF tree: 1) if $lcm(\mathcal{P}_n) \in \mathcal{P}$, node n has at most one child; otherwise, it has exactly two children; 2) suppose $lcm(\mathcal{P}_n) \notin \mathcal{P}$ and

the two children of n are associated with period sets $\mathcal{P}_{n1}$ and $\mathcal{P}_{n2}$, respectively, then $\mathcal{P}_{n1}$ and $\mathcal{P}_{n2}$ are a 2-partition of $\mathcal{P}$ and $lcm(\mathcal{P}_{n1}) + lcm(\mathcal{P}_{n2})$ is the minimum among all 2-partitions of $\mathcal{P}$. According to property 1), a node has at most one or two children depending on whether its LCM belongs to period set $\mathcal{P}$. Intuitively, this property leads to a deep tree and reduces the number of additions because the folding on a node can reuse the results of its parent. According to property 2), the optimal partition of a period set is the one whose LCMs have the minimum sum. We now briefly discuss why this property leads to the minimum number of additions. Suppose we count the total number of additions while a CMF tree is being traversed from node n. The number of additions for folding on $\mathcal{P}_{n1}$ and $\mathcal{P}_{n2}$ is $(lcm(\mathcal{P}_n) - lcm(\mathcal{P}_{n1})) + (lcm(\mathcal{P}_n) - lcm(\mathcal{P}_{n2}))$, which reduces to $2 \cdot lcm(\mathcal{P}_n) - (lcm(\mathcal{P}_{n1}) + lcm(\mathcal{P}_{n2}))$. After accounting for the folding cost of n_1 and n_2's children, the total number of additions becomes:

$$
\begin{aligned}
&2 \cdot lcm(\mathcal{P}_n) - (lcm(\mathcal{P}_{n1}) + lcm(\mathcal{P}_{n2})) + \backslash \\
&2 \cdot lcm(\mathcal{P}_{n1}) - \sum lcm(c(n1)) + 2 \cdot lcm(\mathcal{P}_{n2}) - \sum lcm(c(n2)) \\
&= 2 \cdot lcm(\mathcal{P}_n) + lcm(\mathcal{P}_{n1}) + lcm(\mathcal{P}_{n2}) - \sum (lcm(c_{n_1}) + lcm(c_{n_2}))
\end{aligned}
$$

where c_{n_i} represents the child set of n_i and $\sum lcm(c_{n_i})$ represents the sum of LCMs of period sets associated with c_{n_i}. The above derivation shows that the contribution of nodes n_1 and n_2 in the total folding cost is $lcm(\mathcal{P}_{n1}) + lcm(\mathcal{P}_{n2})$. Therefore, this result is consistent with property 2) of the optimal CMF tree that minimizes the total LCMs of a period set partition.

Based on properties 1) and 2), we have developed an algorithm to find the optimal CMF tree, which is shown in Algorithm 1. The algorithm recursively builds a CMF tree from the root node (step 1-3). For each node newly added to the tree, function *CreateTree* finds its children and the associated period sets and the LCMs. At step 4 and 5, all possible partitions of the current node's period set and their LCMs are found. At step 6, according to property 2) of the optimal CMF tree, the partition that has the minimum sum of LCMs is chosen for generating new child node(s). In step 4-6, a straightforward exhaustive search is used to enumerate possible partitions. We have developed a dynamic programming procedure to reduce the computation cost. We note that the CMF tree algorithm is only executed *offline* and hence the computation cost is not a major issue. Once a tree is found, it can be stored on mobile devices (at a space cost of $O(|\mathcal{P}| \cdot \lg |\mathcal{P}|)$) to guide the search of true beacon periods.

4.5 CFAR Beacon Detector

The output of CMF contains the folding results of all periods in $\mathcal{P}$. The next task of ZiFi is to identify which results correspond to true WiFi beacons. However, this is not trivial due to the following reasons. 1) Non-beacon signals such as WiFi data frames or interference from other RF sources may also exhibit periodicity. The resulted strong folding peaks may be falsely detected as beacons. 2) A beacon frame must compete for the channel with other pending frames and defer its transmissions when the channel is busy. As a result, the transmission times of beacons may become aperiodic leading to detection misses. 3) An RSS sample may participate in the folding of multiple periods. Therefore, the RSS

Algorithm 1 Optimal CMF Tree Construction.

Input: $\mathcal{R}$ - RSS samples; $\mathcal{P}$ - set of possible beacon periods.
Output: $\mathcal{T}$ - the constructed CMF tree.

1: $\mathcal{T}.root = CreateNode(LCM(\mathcal{P}), \mathcal{P})$
2: $CreateTree(\mathcal{T}.root)$
3: *return* $\mathcal{T}$

function $CreateTree(root)$
4: $\mathcal{W} = \{W | W$ *is an [1,2]-partition of* $root.periodSet\}$
5: $\mathcal{Y} = \{Y | Y$ *is the LCM set of* $W \in \mathcal{W}\}$
/*find the partition with the minimum sum of LCMs*/
6: $< \hat{Y}, \hat{W} >= arg\min_{Y \in \mathcal{Y}, W \in \mathcal{W}} sum(W)$
7: ***for all*** *tuple* $< L, S > \in < \hat{Y}, \hat{W} >$ ***do***
8: $child = CreateNode(L, S)$
9: $root.addchild(child)$
10: $CreateTree(child)$
11: ***end for***

function $Node\ CreateNode(L, S)$
12: $node = new\ Node()$
13: $node.lcm = L$; $node.periodSet = S$
14: return $node$

samples of a real beacon signal are essentially noise to the detection of beacons of other periods. We refer to such noise as *cross-period* noise. Fig. 7 illustrates cases 1) and 2) in the folding result of a TelosB mote trace. We also used the trace collected by a WiFi sniffer to label each RSS sample of the TelosB mote and identify three types of peaks due to beacons, data frames, and deferred beacons. It can be seen that the deferred beacons often cause a cluster of small folding peaks because of their random backoff delays.

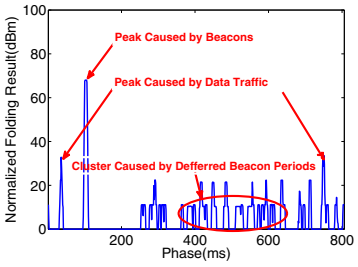


Figure 7: The folding peaks of beacons, data frames, and deferred beacons.

We have developed a constant false alarm rate (CFAR) detector [27] to identify WiFi beacons from folding results of CMF. A CFAR detector minimizes the FN rate while satisfying the FP rate upper bound specified by user [27]. We choose a CFAR detector for our problem because it can explore the fundamental trade-offs between the FN and FP rates of beacon detection. For instance, although using a high threshold to detect folding peaks reduces FPs, it may cause excessive FNs when the folding peaks of real beacons are not strong. A user using ZiFi may specify the FP upper bound based on the energy budget of its mobile devices while allowing ZiFi to automatically minimize the FN rate, i.e., the probability of missing WiFi connectivity.

A challenge of designing a CFAR detector for our problem is to model the detection FPs and FNs caused by non-beacon signals and 802.11 backoff. Our analysis in Section 5 showed that the inaccuracy of beacon detection is closely dependent on channel utilization. Intuitively, a busier channel likely reduces the signal to noise ratio in beacon detection due to

more interference from periodic noise. At the same time, the likelihood that a beacon frame suffers from backoffs is also higher. Based on our analytical results in Section 5, the beacon detector can use the optimal detection threshold to achieve desired upper bound on FP rates while minimizing the FN rate. We further optimize the beacon detector by adopting a cross-period noise reduction mechanism. Specifically, when a beacon signal is identified, all the RSS values of it are removed before the folding is performed on another period.

The pseudo code of beacon detector is shown in Algorithm 2. It takes as input the RSS samples $\mathcal{R}$, user-specified upper bound on FP rate (denoted by FP), and outputs the periods and beacons that are detected. Initially, the channel utilization U is estimated based on $\mathcal{R}$ as follows:

$$U = \frac{|\{\mathcal{R}[i] \mid (\mathcal{R}[i] \in \mathcal{R}) \wedge (\mathcal{R}[i] \neq 0)\}|}{|\mathcal{R}|} \quad (2)$$

In Eq. (2), the channel is deemed as busy if the RSS sample has a non-zero magnitude. At step 2, the detector computes the detection threshold α based on U and FP according to our analytical result in Section 5. At step 3, the detector runs CMF to perform folding on RSS samples $\mathcal{R}$ for all the periods in $\mathcal{P}$. The folding results are stored in $\{\mathcal{F}_P\}$ where $\mathcal{F}_P$ is the set of folding results of period P. At step 4, the maximum folding result is first normalized by factor P and then compared against the threshold α. We note that the normalization is needed because the number of additions in a folding result is inversely proportional to the period. A folding peak greater than the threshold indicates a successful detection. The period and phase of the maximum folding peak are then used to find the RSS samples of detected beacon. The magnitude of these RSS samples are set to zero at step 9 to reduce the cross-period noise. The above process is then repeated for finding beacons of other periods until the maximum folding peak is smaller than the detection threshold.

Algorithm 2 Beacon Detector.

Input: $\mathcal{R}$ - RSS samples; FP - user-specified FP rate upper bound; $\mathcal{P}$ - set of possible beacon periods.
Output: $\mathcal{P}^*$ - set of periods of detected beacons, initially empty.

1: Compute channel utilization U using $\mathcal{R}$ by Eq. (2).
2: Compute threshold α using U and FP by Eq. (4).
/*perform folding for all possible periods*/
3: $\{\mathcal{F}_P | P \in \mathcal{P}\} = Common\_Multi\_Folding(\mathcal{R}, \mathcal{P})$.
/*find the max normalized folding result of each period*/
4: $\{\mathcal{F}_P^{max} | P \in \mathcal{P}; \mathcal{F}_P^{max} = P \cdot \max_i \mathcal{F}_P[i]\}$.
5: Sort $\{\mathcal{F}_P^{max}\}$ in the descending order.
6: **for all** $\{\mathcal{F}_P^{max} \mid P \in \mathcal{P}\}$ **do**
7: **if** $\mathcal{F}_P^{max} \geq \alpha$ **then**
8: $\hat{i} = arg\max_i \cdot \mathcal{F}_P[i]$
/*remove RSS values of detected beacons.*/
9: $\forall j \in [0, \lfloor N/P \rfloor - 1]$, $\mathcal{R}[\hat{i} + j \cdot P] = 0$
10: $\mathcal{P}^* = \mathcal{P}^* \cup \{P\}$
11: *goto 1*
12: **else**
13: *return*
14: **end if**
15: **end for**

4.6 Controlling WiFi Interface

A challenge in the design of ZiFi is that WiFi APs operate on unknown channels. A straightforward solution is to run ZiFi on each of the 11 802.11 channels in 2.4 GHz band. However, it leads to significant detection delay. We employed several mechanisms to address this issue. First, due to the overlap between 802.11 and 802.15.4 channels [28], we found that running ZiFi on four 802.15.4 channels (1,5,8,11) can reliably detect the APs running on all 11 802.11 channels. Moreover, ZiFi can turn on WiFi NIC once the first AP is discovered. The rationale is that the delay for a WiFi NIC to scan all 802.11 channels is short (typically shorter than 800 *ms* in our measurement). ZiFi is able to find multiple beacons with different periods, which are likely from different APs. Thus an interesting issue is how to aid WiFi NIC to choose the best AP without incurring the scanning delay. A possible solution is to estimate the signal quality of each AP based on the RSS samples of detected beacons (before they are changed in RSS shaper) and then notify the WiFi NIC driver of the best channel for association. We have implemented a preliminary version of this approach. However, the full evaluation is left for future work.

5. PERFORMANCE ANALYSIS

As discussed in Section 4.5, the detection performance of ZiFi is inherently probabilistic due to several error sources: the periodicity of non-beacon WiFi signals, beacon backoff delays, and cross-period noise. We focus on modeling the first two factors as ZiFi effectively mitigates the cross-period noise by iteratively removing RSS samples of detected beacons. Our analysis on FN and FP rates can guide the selection of optimal detection thresholds for ZiFi beacon detector.

Our analysis is based on the following channel model. We assume that the channel utilization ratio U is a time-invariant constant during the RSS sampling window. That is, the traffic is uniformly distributed at random on the channel. Our experiments show that this assumption is reasonable because ZiFi only requires a very short RSS sampling window (typically shorter than a second, see Section 6).

5.1 Analysis of FP Rate

According to the beacon detector (Algorithm 2), a beacon signal is detected when the folding peak is no lower than the threshold α. Therefore, FPs occur if noise leads to such a folding peak while beacon signals are absent. In the following, we derive FP rate when the channel utilization ratio of non-beacon signals is U which is estimated from RSS samples according to Eq. (2). However, RSS samples may contain real beacons, which leads to an overestimation of U that is defined for non-beacon signals only. However, this does not introduce substantial errors because the number of beacons is extremely small. As the sampling period of ZiFi is 122 *us* (see Section 4.3), there are about 839 RSS samples between two occurrences of beacon signals when the beacon period is set to the default length of 102.4 *ms*. We assume that $N \mod P = 0$. This condition can be enforced by removing the extra RSS samples. Our analysis can also be easily extended to the case where $N \mod P \neq 0$. Let $f(P, \alpha)$ denote the probability of detecting a (false) beacon signal of period P. It can be seen that $f(P, \alpha)$ is equal to the probability that the maximum of normalized folding results is no lower than α. Formally, $f(P, \alpha)$ is given by:

$$
\begin{aligned}
f(P,\alpha) &= Prob\left\{max_{i\in[1,P]} P\cdot\mathcal{F}_P[i] \geq \alpha\right\} \\
&= 1 - \prod_{i\in[1,P]} Prob\left\{\mathcal{F}_P[i] < \frac{\alpha}{P}\right\} \\
&= 1 - \left(Prob\left\{\mathcal{F}_P[i] < \frac{\alpha}{P}\right\}\right)^P
\end{aligned}
$$

According to Eq. (1), the ith folding result of period P, $\mathcal{F}_P[i]$, is the sum of N/P RSS samples. Note that the RSS shaper sets the value of each sample as either 1 or 0 (see Section 4.3). According to our channel utilization model, the probability that an RSS sample is 1 (e.g., the channel is busy) is equal to U. Therefore, the probability that $\mathcal{F}_P[i]$ is smaller than $\frac{\alpha}{P}$ can be computed as follows:

$$
f(P,\alpha) = 1 - \left(1 - \sum_{k=\lfloor\frac{\alpha}{P}\rfloor}^{N/P} \binom{\frac{N}{P}}{k} U^k (1-U)^{\frac{N}{P}-k}\right)^P \tag{3}
$$

The overall FP rate (denoted by $\mathcal{FP}$) is equal to the probability that a FP occurs for any period $P \in \mathcal{P}$ where $\mathcal{P}$ is the period set searched by ZiFi:

$$
\mathcal{FP} = 1 - \prod_{P\in\mathcal{P}} (1 - f(P,\alpha)) \tag{4}
$$

For a given FP upper bound, detection threshold α can be easily computed by Eq. (4). To reduce computation overhead, we discretize the possible FP range, compute corresponding α values offline, and store them in a table for online lookups. We note that a small FP bound is often desired in order to reduce unnecessary WiFi NIC wake-ups. Therefore, the storage cost of α table is small.

5.2 Analysis of FN Rate

Although WiFi beacons are scheduled to transmit at fixed times, they must follow CSMA and defer their transmissions when the channel is busy. Such a scenario is depicted in Fig. 8(a). Backoff delays due to channel contention hurt the periodicity of beacons, which leads to low folding peaks and FNs in the detection. In this section, we derive the FN rate of beacon detection based on an 802.11 CSMA backoff model.

According to 802.11, an AP with a pending beacon must sense the channel before transmission. If the channel is idle for a period of time equal to a distributed interframe space (DIFS), it broadcasts the beacon immediately. Otherwise, if the channel is sensed busy (either immediately or during the DIFS), the AP waits for the channel to be idle for the DIFS and then starts the exponential backoff procedure. It generates a random backoff interval uniformly chosen in the range $(0, CW-1)$, where CW is called the minimum contention window. We now derive the distribution of backoff time of beacons. In 802.11, the backoff procedure is performed in slotted time with a slot duration of 20 *us*. As shown in Fig. 8(a), the backoff time X (slots) is the sum of waiting time for the idle channel (X_1 slots) and the extra waiting time due to the busy channel within the backoff interval (X_2 slots). Thus, $X = X_1 + X_2$. We first derive the distribution of delay X and then derive the FN rate based on the impact of delay X on folding.

X_1 follows the geometric distribution because the probability of busy channel is equal to channel utilization ratio U. The CDF of X_1 is given by:

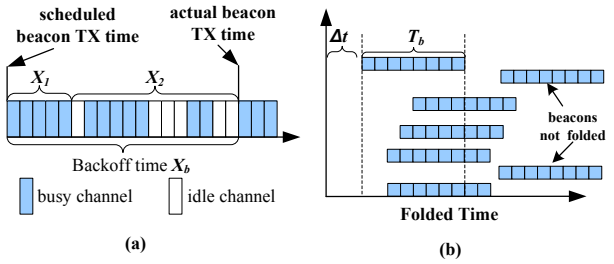


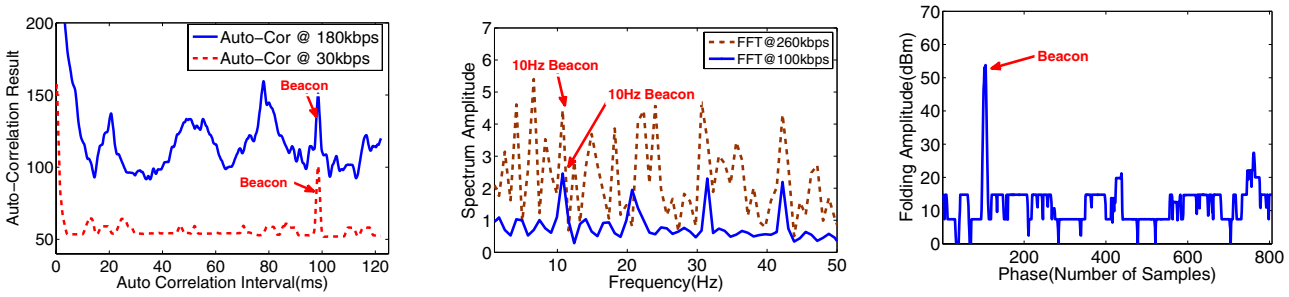
Figure 8: Beacon backoff and impact on folding result. (a) shows the different parts in the backoff delay. (b) shows the temporal distribution of beacons when RSS samples are folded from the transmission time of the top beacon. The beacons that fall outside the T_b window do not contribute to the folding peak.

$$
Prob(X_1 \leq x) = \sum_{i=0}^{x} U^i(1-U) \tag{5}
$$

According to 802.11 standard, when the channel becomes busy during the backoff interval, the backoff timer will be frozen and resumed once the channel is idle again. Therefore, the waiting time due to the busy channel within the backoff interval is equal to the time window within which the number of idle slots is exactly the chosen backoff interval and the last slot is idle. Moreover, the random backoff interval follows the uniform distribution. That is, the probability that a backoff interval is chosen is $1/CW$. In summary, the PDF and CDF of X_2 can be derived as follows:

$$
\begin{aligned}
Prob(X_2 = x) &= \frac{1}{CW} \sum_{i=0}^{min\{CW-1,x\}} \binom{x-1}{i} (1-U)^{i+1} U^{x-1-i} \\
Prob(X_2 \leq x) &= \sum_{j=1}^{x} Prob(X_2 = j)
\end{aligned} \tag{6}
$$

The backoff delay has a complex impact on folding peaks. In particular, when beacons have different backoff delays, the value of folding peak depends on how these delays are aligned with each other in the folding window. Suppose the beacon in-air time is T_b (slots) and the beacon that is transmitted at time instant t has a backoff delay δ. We assume that the start of folding window is aligned with t. Fig. 8(b) shows the temporal distribution of all beacons in such a case. If a beacon has a backoff delay larger than $\delta + T_b$, it is not included in the folding. For other beacons, they contribute to the folding peak equally as their RSS samples are set to be the same value after passing the ZiFi RSS shaper (see Section 4.3). We now derive the resulted FN rate (denoted by $P_{FN}(\delta)$) due to the exclusion of some beacons with large backoff delays. We note that $P_{FN}(\delta)$ is an *upper bound* on the real FN rate because the highest folding peak (which is examined by the beacon detector) may not be generated by the folding from time instant t. A FN occurs when no more than $\lfloor\frac{\alpha}{P}\rfloor$ RSS samples among all N/P samples added in the folding have a non-zero value, where P is the real beacon period. As discussed above, these RSS samples correspond to the beacons whose backoff delays must fall in the time window $[\delta, \delta + T_b]$. Therefore, $P_{FN}(\delta)$ can be derived as follows:



(a) Autocorrelation, 30 and 180 Kbps. (b) FFT, 100 and 260 Kbps (c) ZiFi, 400 Kbps

Figure 9: Performance of Autocorrelation, FFT, and ZiFi at different WiFi data traffic rates.

$$f_{FN}(\delta) = \sum_{k=1}^{\lfloor \frac{\alpha}{P} \rfloor} \binom{N}{k} Prob(\delta \le X \le \delta + T_b)^k \cdot \quad (7)$$

$$(1 - Prob(\delta \le X \le \delta + T_b))^{\frac{N}{P}-k} \quad (8)$$

where $X = X_1 + X_2$ and the CDF of X can be derived from the CDFs of X_1 and X_2 given in Eq. (5) and (6), respectively. The beacon backoff time ranges from 0 to a constant Δ. Thus the expected FN rate is given by:

$$\mathcal{FN} = \sum_{\delta=0}^{\Delta} Prob(X = \delta) \cdot f_{FN}(\delta) \quad (9)$$

We note that the constants Eq. (9) (Δ and CW) are PHY-specific parameters defined in 802.11 standard.

6. EXPERIMENTATION

6.1 Experimental Setup and Methodology

We implemented ZiFi on two platforms: ASUS Linux netbook integrating a TelosB mote through the USB interface, and Nokia N73 smartphone integrating a ZigBee card through the miniSD interface. The CMF algorithm is implemented in Mablab on netbook and in C++ on Nokia N73. The RSS sampler of ZiFi is implemented in ZigBee module on both platforms and all other components run on netbook or Nokia N73. The RSS sampling interface on Nokia N73 only allows one RSS sample to be transferred from the miniSD card at a time. Our measurement showed that the resulted RSS sampling rate cannot meet the requirement of ZiFi. However, the ZigBee module developers at Nokia confirmed that this problem is not due to the limitation of ZigBee chip and can be resolved by providing a batch RSS sampling interface in a future release of Software Development Kit (SDK). Our experiments on Nokia N73 are carried out using preloaded RSS samples.

Our experimental testbed consists of four 802.11g APs, four Linux-based 802.11 netbooks, two TelosB motes equipped with CC2420 radios, and a Nokia N73 smartphone. To evaluate the performance of detecting APs from different vendors, we intentionally used four wireless routers (Linksys, Belkin, TP-LINK, and D-Link) from different vendors as APs. ASUS Eee netbooks equipped with Intel Athero 928x NICs are used as clients. Our evaluation focuses on the following performance metrics: 1) *AP detection accuracy characterized by FP and FN rates.* FPs in AP detection can falsely turn on WiFi interfaces leading to energy waste while FNs cause WiFi clients to miss the network connectivity opportunities that are available. We evaluate FP and FN rates under different settings of WiFi (e.g., channel rates), traffic workload, and parameters of ZiFi. 2) *Computation and energy overhead.* Mobile devices have tight budgets on energy and CPU resources. Our evaluation shows that ZiFi incurs little overhead while significantly reducing the idle time of NICs and hence leads to overall system energy conservation.

The performance of ZiFi depends on both the characteristics WiFi APs (e.g., modulation rate and transmit power) and user traffic (e.g., workload). In our experiments, the user traffic is generated from a high-fidelity Internet traffic generator called D-ITG [4] that runs on our APs. D-ITG has several advantages over the existing traffic generators such as the capability of generating multiple simultaneous flows from different protocols. Empirical results showed that D-ITG can reproduce realistic traffic patterns under a wide range of network settings [4]. The use of D-ITG allows us to evaluate ZiFi in comprehensive WiFi and traffic settings, which would be impossible for using particular operational WiFi deployment. We note that several large-scale WiFi WiFi traces (e.g., the SIGCOMM [9] and OSDI [3] traces) are publicly available. However, they are collected under particular network settings.

6.2 Detection Accuracy

We evaluate the detection accuracy of ZiFi using two WiFi nodes. A Linksys WRT54G2 router is used as AP and an ASUS Linux netbook serves as the client. ZiFi is executed on another netbook and a TelosB mote that are connected via USB. The traffic generated by client contains one TCP stream and one UDP stream. The length of frames is uniformly distributed, from 5 to 1400 bytes for TCP and 50 to 1400 bytes for UDP. We vary the average modulation rate to control the channel utilization. The distance between AP and ZiFi node is 3 meters. In Section 6.3, we evaluate the detection range of ZiFi by varying the distance between AP and ZiFi node. The network traffic is logged as ground truth for micro-scale analysis of ZiFi performance. The experiments were conducted in a residential environment. The AP uses channel 1 and both AP ad client transmit at the default power level. The length of AP beacon period is configurable at a step of 1.024 *ms*. We varied the period length and observed no obvious performance variation of ZiFi. We used a fixed period of $96 \times 1.024 = 98.304$ *ms* throughout all experiments. However, as this setting is unknown to ZiFi, the CMF algorithm of ZiFi searches for the beacon period within the range of $(60 \sim 120) \times 1.024$ *ms*.

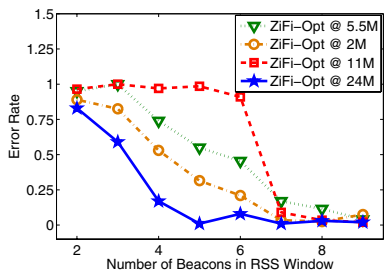


Figure 10: Detection error rate vs. number of beacon periods.

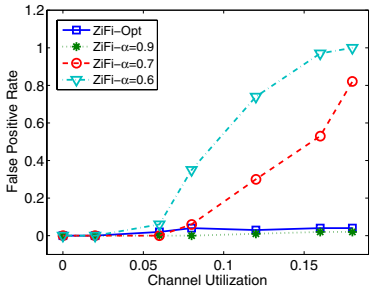


Figure 11: Detection error rate vs. ZigBee channel utilization.

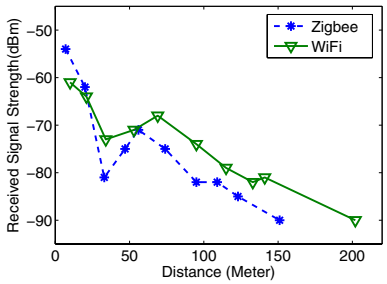


Figure 12: The RSS sensed by a TelosB mote and an 802.11 netbook.

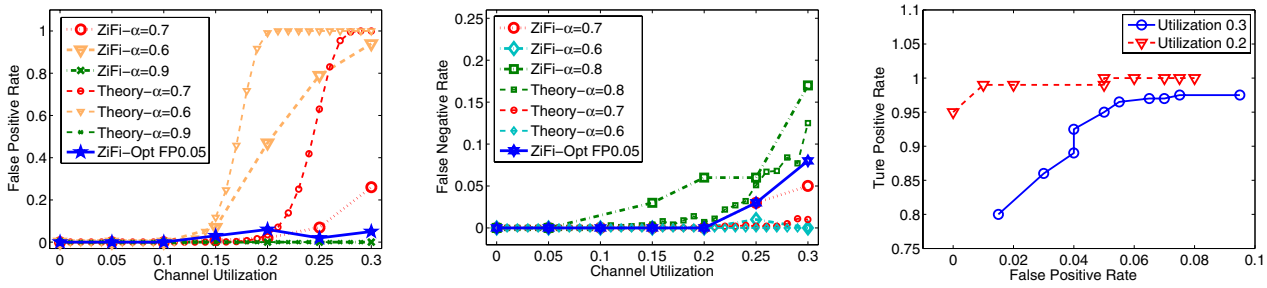
Comparison with other signal processing approaches. Autocorrelation and FFT are two signal processing algorithms widely used to detect periodic signals. We now compare the performance of them against that of ZiFi. Fig. 9 shows the results when they are applied to 10000 RSS samples (i.e., 1.22 second) of the traffic. The AP modulation rate is set to 2 Mbps. When the data rate is only 30 Kbps, Autocorrelation clearly detects a peak in its results due to the beacons. However, its performance drops sharply when data rate increases. In the result of 180 Kbps, the beacon peak can hardly be distinguished from the peaks due to data traffic. When the data rate is 100 Kbps, FFT identifies the 10 Hz beacon signal in the first peak and the harmonics of the signal in the following peaks. However, it fails to identify the beacon signal when the data rate increases to 260 Kbps. Therefore, these results show that Autocorrelation and FFT cannot reliably identify the existence of WiFi networks. In contrast, ZiFi successfully detects beacons under all settings. Fig. 9 (c) shows the folding results of ZiFi for 400 Kbps where the beacon peaks can be clearly identified.

Impact of RSS window size. The size of RSS window used by ZiFi is a critical design parameter as it directly determines the detection delay and overhead. Fig. 10 shows the detection error rate of ZiFi when the RSS window contains different numbers of beacon periods. The AP transmits at each of four different modulation rates in turn while the channel utilization ratio is always tuned to 30%. Each data point is the average of 5 runs. For each run, ZiFi carries out the detection for 40 times and the error rate is computed as the probability of failing to detect a beacon or falsely detecting a non-beacon signal, i.e., the sum of FN and FP rates. Since the in-air time of data frames transmitted in 11 Mbps is very close to that of beacon frames, most RSSI samples of data frames are not removed by the RSS shaper, which causes significant noise in the folding results. However, even in this worst case, the error rate of ZiFi quickly decreases to near-zero when the RSS window contains more than only 7 beacons. For instance, when 8 beacon periods of RSS samples are used (which corresponds to a total delay of $8 \times 96 \times 1.024 = 786.4$ *ms*), ZiFi's average error rate under four channel rates is only 4.8%.

Impact of ZigBee interference. The RSS samples gathered by ZiFi may contain the transmissions of other devices operating in the open radio spectrum, which thus introduce noise in WiFi beacon detection. In particular, the RSS samples can be easily contaminated by the traffic of peer ZigBee nodes. Such noise could be eliminated if the ZigBee radio of ZiFi is able to decode these frames. However, this is often impossible because the wireless interference range is typically much larger than the communication range. Moreover, the frames transmitted by radios on overlapping channels can be sensed by RSSI but cannot be decoded [28]. We now evaluate the impact of such interference from peer ZigBee nodes. In the experiment, a pair of TelosB motes transmit on an overlapping channel. The frame sizes are uniformly distributed between 14 and 74 bytes. The in-air time of these frames has a significant overlap with that of WiFi traffic. The data rate of transmission is varied from 1.35 to 27.1 Kbps to obtain different channel utilization ratios. Three ZiFi variants are used as baselines for comparison. 'ZiFi-α=x' refers to the implementation of ZiFi where the detection threshold is manually set to be $\alpha \cdot N/P$ where $\alpha \in (0, 1]$, N and P are the total number of RSS samples and the real beacon period, respectively. 'ZiFi-opt' refers to the default implementation of ZiFi that computes the threshold based on an 0.05 FP upper bound. Fig. 11 shows that ZiFi-opt yields near-zero false positives. In contrast, two ZiFi variants falsely classify more ZigBee signals as WiFi beacons when the channel workload is higher. The main reason is that ZigBee traffic contains more periodic signals under heavier traffic load, which results in many folding peaks. ZiFi-opt automatically chooses high detection thresholds to avoid such false positives and yield similarly performance as 'ZiFi-α=0.9' which has a manually set high threshold.

Impact of traffic workload. Our objective of evaluation in this experiment is three-fold. First, we test the FP and FN rates of ZiFi under various settings of channel utilization ratio and bit rate. Second, we compare the experimental results with analytical result presented in Section 5. Third, we plot the receiver operating characteristic (ROC) [27] curve of ZiFi. ROC characterizes how true positive (TP) rate varies with FP rate and is widely adopted for evaluating the capability of detection systems.

The maximum channel utilization ratio in our evaluation is set to 30%. We note that real-world WiFi deployments usually have low channel utilization ratio. Our analysis of over 10^4 seconds of traces collected at OSDI 2006 [3] and SIGCOMM 2008 [9] show that the channel utilization ratios (computed per second) have a mean of 7.58% and 0.81%, and median 7.2% and 0.3%, respectively. This result is also consistent with the recent finding [16] that significant white space exists in real WiFi traffic. Fig. 13(a) and (b) show the FP and FN rates under different channel utilization ratios. It can be seen that ZiFi variants with fixed detection thresholds yield poor FP or FN rate. For instance, although a low threshold (e.g., 0.6) has a near-zero FN rate, it leads to ex-



(a) FP rate vs. channel utilization ratio. (b) FN rate vs. channel utilization ratio. (c) True positive rate vs. FP rate.

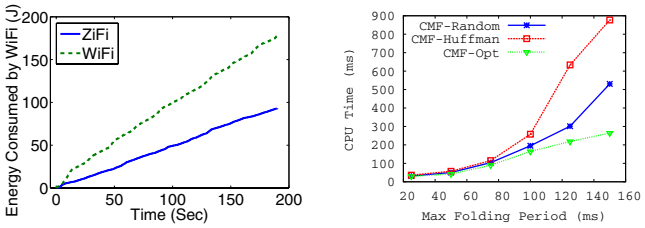
Figure 13: Detection performance of ZiFi.

tremely high FP rates when channel is heavily loaded. This is because the folding peaks of noise (data frames) become higher and many of them exceed the low threshold. However, when the threshold is set to 0.9, although the FP rate is low, many beacons are missed. In contrast, ZiFi-opt can achieve both satisfactory FP and FN rates by automatically adjusting the threshold based on channel workload. Under all settings, ZiFi-opt has a FP rate lower than the preset upper bound 0.05 while achieving low FN rates. It can be seen from Fig. 13(b) that the theoretical prediction matches the experimental FN under all settings. However, Fig. 13(a) shows a considerable gap between theoretical and experimental FP rates when the channel utilization is high. This is due to two reasons. First, ZiFi implements an RSS shaper that can remove some noise, e.g., the RSS samples that are likely data traffic (see Section 4.3). However, the impact of RSS shaper is not modeled in our FP analysis. Second, our FP analysis is based on a uniform channel utilization model where the probability at which any slot is busy is constant. However, the data traffic under this model yields better periodicity than reality because the burstiness [16] of real WiFi traffic is not considered. As a result, the theoretical FP rate is a pessimistic estimation of real FP rate.

Fig. 13(c) plots the ROC curves of ZiFi. We vary the FP upper bound from 0.01 to 0.46 at a step of 0.05, and calculate the true positive (TP) and FP rates for each setting. It can be seen that ZiFi achieves a good TP rate if the allowable FP rate is above 4%. We can also see that ZiFi has a desirable configurability and allows a user to achieve trade-offs between FP rate and TP rate. For instance, one may set a higher FP bound to maximize the opportunity of finding WiFi networks while setting a lower FP bound to reduce the number of NIC wake-ups for energy conservation.

6.3 Detection Range, Energy Consumption, and Overhead

We now evaluate the performance of ZiFi in a mobile environment. We deploy three APs, which work on channel 1, 6, and 11, respectively. In all experiments (except the one to measure detection range), a person carrying a ZiFi or WiFi node moves between four preselected locations in a room of 20X7 m^2. Three of four locations have WiFi coverage from at least one of the APs while the last location has no coverage. After arriving at each location, it stays for 3 minutes before moving to the next location. This setup is to mimic the intermittent WiFi coverage experienced by mobile users. The WiFi node switches its NIC on for 5 seconds once every 20 seconds to detect WiFi coverage. On the ZiFi node, ZigBee radio is first switched on, which decides whether to



(a) Energy consumption (b) CPU Overhead

Figure 14: CPU overhead of CMF on N73 smpartphone, and energy consumption of ZiFi and WiFi nodes.

wake up the WiFi NIC based on the detection results. The energy consumption of the whole system is measured from inquiring the current readings from the ACPI interface.

We first evaluate the distance within which ZiFi can reliably detect APs. Typical ZigBee radios such as CC2420 [2] have a reception sensitivity below -90 dBm, which is similar to WiFi radios. To verify this, we measured the average RSS of a WiFi AP on both a ZigBee node and a WiFi node. Fig. 12 shows that the RSS of ZigBee and WiFi nodes are very similar. Due to a concrete wall, the RSS drops sharply around 40 meters from the AP. Moreover, we found that the WiFi node can successfully associate with the AP until they are more than 140 meters away. The overall result shows that ZiFi has a similar detection range as WiFi node.

We then measure the energy consumption of both ZiFi and WiFi nodes. Due to the constraints of experimental space, the transmission range of APs is reduced by wrapping aluminum foil around the antennas. We only measure the energy consumption when there is no WiFi coverage. Once an AP is discovered, both WiFi and ZiFi nodes consume similar power. Fig. 14(a) shows that ZiFi is significantly more energy-efficient than WiFi node. This is because only ZigBee radio is active when the network coverage is absent, while the WiFi node periodically wakes up NIC and scans for APs leading to significant energy waste.

Finally, we measure the CPU overhead of ZiFi on Nokia N73 smartphone. Total 80K RSS samples are searched for the beacon period whose value lies in set $\{25, \cdots, 25 \cdot k\}$ ms. We vary k from 1 to 6 to evaluate the CPU time of running CMF with different sizes of period sets. The result is the average of 20 runs. For performance comparisons, we implemented two baseline algorithms called CMF-Random and CMF-Huffman. Both algorithms build the tree in a bottom-up fashion. Initially, a single-node tree is created for each period and every two trees from the current tree set are then

merged into one tree by adding a new root. This process continues iteratively until a single tree remains. CMF-Random chooses two trees randomly from the current trees to merge while CMF-Huffman chooses the two trees whose roots have the minimum periods. Fig. 14(b) shows the CPU time of folding on the trees found, where X-axis is the maximum value of the period set. We can see that the CPU usage for both baseline algorithms increase sharply when the maximum period exceeds 100 *ms*. In contrast, CMF-opt has significantly lower overhead. The maximum CPU time of running CMF-opt under all settings is around 200 *ms*.

7. CONCLUSION

We developed a system called *ZiFi* that utilizes ZigBee radio to identify the existence of WiFi networks by detecting interference signatures generated by WiFi beacons. A new DSP algorithm called Common Multiple Folding (CMF) is developed to amplify signals with unknown periods in WiFi interference samples. ZiFi also adopts a constant false alarm rate (CFAR) detector that can minimize the false negative (FN) rate of WiFi AP detection while satisfying the user-specified upper bound on false positive (FP) rate. Our evaluation results on two platforms, Linux netbook connected to a TelosB mote through the USB interface, and Nokia N73 smartphone that integrates a ZigBee card through the miniSD interface, showed that ZiFi can detect WiFi APs with high accuracy, short delay, and little overhead.

8. ACKNOWLEDGEMENT

This research was supported in part by the US National Science Foundation under grant CNS 0916576, the State Key Program of National Natural Science of China under grant 60933011, the National Natural Science Foundation of China under grant 60873241, and the National High Technology Research and Development Program of China (863 Program) under grant 2008AA01Z217. We thank Dr. Canfeng Chen at Nokia Research Center (China) for providing the miniSD ZigBee module of Nokia N73 and offering assistance to our experiments.

9. REFERENCES

[1] Texas Instruments Inc., BRF6300 BlueLink 5.0.

[2] Texas Instruments Inc., CC2420: Single-Chip 2.4 GHz IEEE 802.15.4 Compliant RF Transceiver.

[3] Crawdad: A community resource for archiving wireless data at dartmouth. `http://crawdad.cs.dartmouth.edu/`.

[4] D-itg distributed internet traffic generator. `http://www.grid.unina.it/software/ITG/`.

[5] Hobbes & co., ltd., wl-f601pro digital wifi detector. `http://www.hobbes-europe.com/product.php5?products_id=79`.

[6] Metageek, llc, wi-spy spectrum analyzer. `http://www.metageek.net/`.

[7] Pantech & curitel p1 cell phone. `http://www.curitel.com`.

[8] Rf micro devices inc. bluetooth transceiver rf2968. `http://www.rfmd.com/`.

[9] Sigcomm 2008 traces. `http://www.cs.umd.edu/projects/wifidelity/sigcomm08_traces/`.

[10] Zigbee alliance, products & certification overview. `http://www.zigbee.org/Products/Overview.aspx`.

[11] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: Energy management for voip over wi-fi smartphones. In *MobiSys*, 2007.

[12] Y. Agarwal, C. Schurgers, and R. Gupta. Dynamic power management using on demand paging for networked embedded systems. In *ASP-DAC*, 2005.

[13] G. Ananthanarayanan and I. Stoica. Blue-Fi: enhancing Wi-Fi performance using bluetooth signals. In *MobiSys*, 2009.

[14] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden. A measurement study of vehicular internet access using in situ wi-fi networks. In *MobiCom*, 2006.

[15] K. Chebrolu and A. Dhekne. Esense: communication through energy sensing. In *MobiCom*, 2009.

[16] S. Geirhofer, L. Tong, and B. Sadler. Dynamic spectrum access in wlan channels: Empirical model and its stochastic analysis. In *TAPAS*, 2006.

[17] N. Mishra, K. Chebrolu, B. Raman, and A. Pathak. Wake-on-WLAN. In *WWW*, 2006.

[18] N. Mishra, D. Golcha, A. Bhadauria, B. Raman, and K. Chebrolu. S-WOW: signature based Wake-on-WLAN. In *COMSWARE*, 2007.

[19] A. J. Nicholson and B. D. Noble. Breadcrumbs: forecasting mobile connectivity. In *MobiCom*, 2008.

[20] T. Pering, Y. Agarwal, R. Gupta, and R. Want. CoolSpots : reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *MobiSys*, 2006.

[21] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *IPSN*, 2005.

[22] A. Rahmati and L. Zhong. Context-for-wireless. In *MobiSys*, 2007.

[23] E. R.V.E.LOVELACE, J.M.SUTTON. Digital search methods for pulsars. In *Nature.Vol.222*, 1969.

[24] E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *MobiCom*, 2002.

[25] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins. Turducken: hierarchical power management for mobile devices. In *MobiSys*, 2005.

[26] D. H. STAELIN. Fast folding algorithm for detection of periodic pulse trains. In *Proc. IEEE, Vol. 57, p. 724 - 725*, 1969.

[27] P. Varshney. *Distributed Detection and Data Fusion*. Spinger-Verlag, New York, NY, 1996.

[28] G. Xing, M. Sha, J. Huang, G. Zhou, X. Wang, and S. Liu. Multi-channel interference measurement and modeling in low-power wireless networks. In *the 30th IEEE Real-Time Systems Symposium (RTSS)*, 2009.

[29] X. Zhang, X. Wang, Y. Ren, C. Chen, and J. Ma. A novel compatible hardware expansion method based on general memory interface. In *CMC*, 2009.

[30] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma. Zifi: Wireless lan discovery via zigbee interference signatures. Technical Report MSU-CSE-10-18, Computer Science and Engineering, Michigan State University, East Lansing, Michigan, 2010.