BlueScan: Boosting Wi-Fi Scanning Efficiency Using Bluetooth Radio

Juheon Yi†, Weiping Sun†, Jonghoe Koo†, Seongho Byeon†, Jaehyuk Choi‡, and Sunghyun Choi†
†Department of ECE and INMC, Seoul National University, Seoul, Korea
‡Department of Software, Gachon University, Seongnam, Korea
Email: {jhyi16, weiping, jhkoo, shbyeon}@mwnl.snu.ac.kr, jchoi@gachon.ac.kr, schoi@snu.ac.kr

***Abstract*—The increasing demand for ubiquitous wireless connectivity has led to the widespread of Wi-Fi networks. However, due to the limited coverage of Wi-Fi networks, mobile stations (STAs) need to frequently search for neighboring Wi-Fi access points (APs). Inevitable inefficiency occurs during the Wi-Fi scanning since the STA typically does not have any prior knowledge of the neighboring APs, thus leading to unnecessary waste of time and energy. In this paper, we propose BlueScan, a scheme to boost Wi-Fi scanning using collocated Bluetooth radio. BlueScan enhances Wi-Fi scanning with low power Bluetooth radio by identifying the operating channels and target beacon transmission times (TBTTs) of neighboring APs, thus changing the Wi-Fi scanning from a blind search to an intelligent search. We implement a prototype of BlueScan using Ubertooth platform, and evaluate its performance through real experiments. Our results demonstrate that BlueScan reduces the scanning delay up to 77% compared to legacy Wi-Fi scanning.**

I. Introduction

Nowadays, Wi-Fi has become extremely widespread in our daily lives, providing ubiquitous wireless connectivity. Wi-Fi has now become the dominant source for mobile data traffic, and this tendency is expected to continue in the future [1]. However, due to the limited coverage of Wi-Fi networks, a mobile station (STA)[1] needs to ceaselessly search for available or better Wi-Fi access points (APs). The process of searching for neighboring Wi-Fi APs is called *Wi-Fi scanning*.

Wi-Fi scanning embeds inefficiency by nature due to the lack of prior knowledge on the existence of neighboring APs provided to STAs, thus becoming a blind search; STAs have to scan every Wi-Fi channel in each Wi-Fi scanning. Such inefficiency can give rise to unacceptable scanning delay, and even worse, excessive energy consumption of the STAs. Several studies report that Wi-Fi scanning drains a substantial portion of smartphone's battery [2–4], and its delay hinders smooth Wi-Fi handover [5].

While Wi-Fi starts to shift its focus to 5 GHz band, many studies [6–8] still emphasize the importance of Wi-Fi on 2.4 GHz band. For example, the studies in [7, 8] reveal that about 40% of Wi-Fi devices still remain 2.4 GHz only. It is also reported that even for dual-band devices supporting both 2.4 GHz and 5 GHz bands, most of them are connected to 2.4 GHz Wi-Fi networks due to the favorable condition for signal propagation provided by this band [8].

Acknowledging the aforementioned issues, there have been many efforts to advance Wi-Fi scanning, by utilizing the collocated low-power wireless personal area network (WPAN) radio, e.g., Bluetooth or ZigBee [9–12]. The main focus of the existing efforts is, in essence, to reduce energy consumption caused by unnecessary Wi-Fi scanning when there is no available Wi-Fi network nearby, by predicting the presence of neighboring Wi-Fi APs using WPAN radio before conducting the actual Wi-Fi scanning. However, they have focused primarily on determining the absence of Wi-Fi APs; there is still a lack of a method to indicate when (i.e., proper timing of the scanning) and where (i.e., the operating channels of the neighboring Wi-Fi APs) the actual Wi-Fi scanning should be conducted after judging that there are available APs nearby.

In this paper, we propose BlueScan, a scheme of **Blue**tooth-aided Wi-Fi **Scan**ning, that achieves further enhancement in WPAN-aided Wi-Fi scanning. BlueScan is a novel approach not only to predict the availability of Wi-Fi networks using WPAN radio but also to enlighten the collocated Wi-Fi with detailed knowledge of neighboring APs, such as the operating channels and the expected timing of the beacon transmissions, referred to as *target beacon transmission time* (TBTT). Fundamental challenges in realizing BlueScan with WPAN radio are: 1) WPAN radio cannot decode Wi-Fi signals; it can only conjecture that there might be an incoming Wi-Fi signal through Received Signal Strength Indicator (RSSI) observation, and 2) multiple Wi-Fi APs' operating channels can overlap fully or partially.

To tackle these challenges, we propose a beacon frame detection algorithm which exploits the periodicity of beacon frames in both time and frequency domains. Based only on RSSI observations obtained from the built-in RSSI sampling mechanism of Bluetooth radio, BlueScan first identifies the TBTTs of periodic beacon frames coming from neighboring Wi-Fi APs on a set of target channels. BlueScan then investigates the congruence of the TBTTs observed over multiple adjacent channels to pinpoint the operating channel of the corresponding beacon frame. BlueScan operates in a purely passive manner (i.e., without introducing extra traffic) without need for hardware modification of STA or AP.

It is worth noting that, although BlueScan is general enough to be implemented on various WPAN radios such as ZigBee, Bluetooth is the best choice to serve our purpose due to the proliferation of the Wi-Fi/Bluetooth combo chipsets in contemporary mobile devices (e.g., smartphone), such that the collocated Bluetooth radio, unlike ZigBee, can be exploited without any need for additional/specialized hardware. Moreover, RSSI sampling rate of Bluetooth radio (e.g., as high as

[1]In this paper, we will refer to the target device (e.g., smartphone) equipped with collocated Wi-Fi and Bluetooth radios as STA.

978-1-5386-4281-8/18/$31.00 ©2018 IEEE

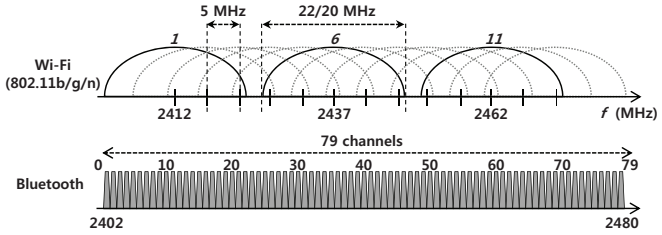


Fig. 1. Channelization of Wi-Fi and Bluetooth.

1 MHz in CC2400 [13]) is generally faster than that of ZigBee radio (e.g., as high as 7.812 kHz in CC2420 [14]), enabling more agile detection algorithm.

The major contributions of this paper are as follows:

- We enable Bluetooth radio to detect neighboring Wi-Fi APs' operating channels and TBTTs through RSSI observation, by developing a detection algorithm exploiting the periodicity of beacon frames in both time and frequency domains.
- We propose BlueScan, as a general framework, to aid and advance Wi-Fi scanning, which incorporates Wi-Fi sending probe requests only on channels where detected APs are operating (selective active scanning), or waking up at specific channel and TBTT to listen for beacon frames (scheduled passive scanning).
- We implement a prototype of BlueScan on Ubertooth platform, and verify its effectiveness via prototype-based experiments. Our results show that BlueScan reduces scanning delay up to 77% compared with legacy Wi-Fi scanning.

The rest of the paper is organized as follows. We summarize preliminary knowledge and related work in Section II. We discuss the motivation for proposing BlueScan, and explain the overall flow in Section III. The main functional blocks of BlueScan are detailed in Section IV. We verify BlueScan's feasibility and effectiveness via prototype-based experiments in Section V and conclude the paper in Section VI.

II. Preliminaries

A. Wi-Fi and Bluetooth Channelization on 2.4 GHz Band

Channelization of Wi-Fi (on 2.4 GHz band) and Bluetooth is shown in Fig. 1. Wi-Fi (802.11b/g/n) defines 13 channels, each with 22/20/20 MHz bandwidth, centered from 2412 MHz to 2472 MHz (5 MHz apart). Bluetooth defines 79 channels, each with 1 MHz bandwidth, from 2402 MHz to 2480 MHz (1 MHz apart). As a result of overlapped channels, Wi-Fi and Bluetooth can interfere with each other, which in turn means they can see each other.

B. Wi-Fi Scanning

There are two types of Wi-Fi scanning: *passive* and *active*. In passive scanning, an STA searches candidate AP(s) by conducting idle listening on each Wi-Fi channel sequentially to capture beacon frames broadcasted by neighboring APs. Beacon frame is supposed to be transmitted periodically at each TBTT, albeit actual transmission suffers random delay caused by carrier sense multiple access with collision avoidance (CSMA/CA). Active scanning involves the STA broadcasting a probe request frame and listening for probe response frame(s) replied by AP(s). Generally, passive scanning takes more time, while active scanning introduces additional traffic to the channel. Specific parameters of Wi-Fi scanning (e.g., active or passive, scanning time per channel, scanning interval) are determined by the OS and hardware.

C. Related Work

Wi-Fi only approaches: Due to the excessive energy consumption caused by frequently triggered Wi-Fi scanning, recent OSs in STAs adjust the Wi-Fi scanning interval to save energy. For instance, in iOS 8, the scanning interval is adjusted exponentially within the range of 15 s to 480 s based on the current state of the device [2]. Although such an approach may reduce energy consumption by conducting the scanning less frequently, the chance of missing available APs also increases as the scanning interval increases. This problem becomes more severe when the STA is on the move.

Cross-technology aided approaches: There have been several efforts exploiting radios collocated with Wi-Fi to enhance Wi-Fi scanning. Blue-Fi [12] proposes a Wi-Fi availability prediction scheme using Bluetooth contact patterns along with cell tower information. Footprint [3] utilizes cell tower information to trigger Wi-Fi scanning only when the location of the STA changes significantly. ZiFi [9] proposes a novel digital signal processing algorithm, called common multiple folding (CMF), to search for Wi-Fi beacon frames with arbitrary period based on the RSSI samples of ZigBee radio. WiSpot [15] utilizes multiple ZigBee radios to further reduce RSSI sampling time compared to ZiFi. Several other efforts [10, 16, 17] utilize ZiFi to enable novel applications without making progress in Wi-Fi scanning. WidthSense [11] proposes a lightweight algorithm to detect Wi-Fi signals via correlation analysis between RSSI samples obtained on two WPAN channels in a zigzag manner.

Unfortunately, none of the existing solutions is sufficient to fulfill our goals. They either require context information [3, 12], which cannot be obtained in unknown places, or ZigBee radios [9, 15], which is not available on most contemporary mobile devices. Besides, none of the previous approaches [9, 11, 15] has the ability to determine the detailed knowledge of the neighboring APs (i.e., operating channel and TBTT). More importantly, they do not provide any clues on how to leverage the information provided by collocated radio to further enhance the Wi-Fi scanning conducted afterwards.

III. BlueScan: Overview

A. Motivation

Periodicity of beacon transmissions: To characterize the signatures of beacon frames, we have captured beacon frames from 288 distinct Wi-Fi APs, and find that, 98.6% of them always broadcast beacon frames with 102.4 ms interval, which is consistent with a well-known fact that the beacon period is rarely changed from its default value [7, 17]. Besides, among the captured beacon frames including only 802.11b

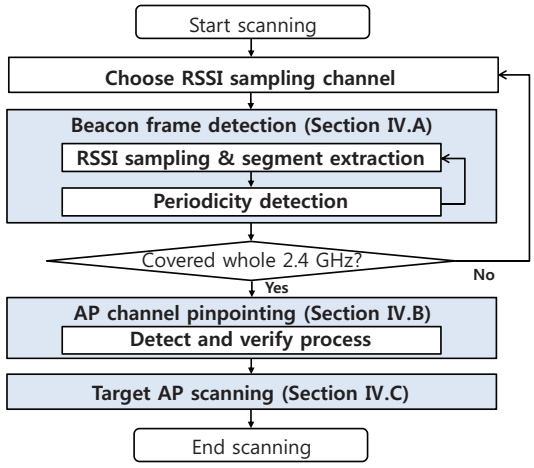


Fig. 2. Overall flow of BlueScan.

and 802.11g beacon frames,[2] 802.11b beacon frames account for 80% of the total, and their airtime is in a fixed range of [1.44, 3.2] ms, corresponding to 2.11 ms on average. The rest are 802.11g beacon frames, whose airtime is in a fixed range of [0.24, 0.53] ms and becomes 0.385 ms on average. *That is, most of the Wi-Fi APs employ fixed beacon period, and the airtime of the beacon frames is within a certain range.* These unique signatures enable WPAN radio to distinguish beacon frames from data traffic, and possibly other interferers in 2.4 GHz ISM band (e.g., microwave oven, ZigBee), and detect Wi-Fi APs only from the observed RSSI samples.

Why Bluetooth? Reasons for choosing Bluetooth to achieve our goal are threefold: 1) *Low power*. Since the power consumption of Bluetooth radio when receiving signal is lower than that of Wi-Fi radio, resolving the inefficiency of Wi-Fi scanning with Bluetooth radio can save energy. 2) *Wide availability*. Unlike ZigBee-based solutions [9, 15] where additional ZigBee radio is required to be applied to most mobile devices, Bluetooth-based approach does not require such additional hardware, since Bluetooth radio is often collocated with Wi-Fi radio as part of combo chipset in most mobile devices. 3) *Faster RSSI sampling*. Bluetooth radio typically provides higher RSSI sampling rate than ZigBee radio. For instance, CC2400 Bluetooth radio provides RSSI sampling rate up to 1 MHz [13], while CC2420 ZigBee radio can only provide RSSI sampling rate up to 7.812 kHz [14]. RSSI samples with higher sampling rate can reflect the variation of the incoming signal more closely, thus enabling more agile detection algorithm, as shown in Section V.

B. Design Philosophy

The main design philosophy of BlueScan is *"Bluetooth tells Wi-Fi when and where to listen for beacon frames of neighboring APs."* BlueScan first collects channel and TBTT information of neighboring APs using Bluetooth radio, and then utilizes the information as a guideline on when and where to conduct the actual Wi-Fi scanning, making the Wi-Fi scanning an informed search which would otherwise have become a blind search. Distinguished from existing approaches, BlueScan mainly handles the following issues: 1) accurately identify the TBTTs and operating channels of neighboring Wi-Fi APs, which have not been addressed in previous approaches, and 2) compared to ZiFi [9] which employs a fixed, long *dwell time* (i.e., the time for the WPAN radio to stay on a certain channel to collect RSSI samples), we adaptively adjust the dwell time depending on the state of the channel so that the delay encountered in beacon frame detection can be optimized without sacrificing the detection performance.

[2]For the sake of backward compatibility, beacon frames are transmitted in 802.11b/g mode at 2.4 GHz. No beacons were transmitted in 802.11n mode.

C. Overall Flow

The overall flow of BlueScan is summarized in Fig. 2. It starts with sampling RSSI with *dwell-and-switch* approach, i.e., dwelling on a certain channel, collecting RSSI samples, detecting periodic beacon frames and TBTT, and moving on to another channel. The dwell time of each channel is adapted considering the channel utilization. After conducting the detection algorithm over the entire 2.4 GHz band, AP channel pinpointing algorithm takes place to determine the actual operating channel of each AP, thus benefiting target AP scanning conducted afterwards.

IV. BlueScan: Algorithms

A. Beacon Frame Detection

The first part of BlueScan is to detect the sets of periodic beacon frames by RSSI observations. There are two challenges to achieve the goal. First, beacon frames are relatively scarce (one in every 102.4 ms) such that a long time may be required to obtain sufficient RSSI samples for detection, resulting in unacceptably high delay. Second, the periodicity of beacon frames can be distorted, since actual beacon transmission time may be delayed due to a busy medium, thus making it difficult to detect periodicity. In what follows, we address these challenges to present a robust detection algorithm that accurately detects beacon frames and identifies its TBTT.

Segment extraction: Beacon frame detection algorithm starts with obtaining RSSI samples (denoted by R) for initial dwell time (T_{min}) and extracting *segments* from them. A *segment* is a set or burst of contiguous RSSI samples above the predefined threshold $RSSI_{th}$ (dBm) and its corresponding airtime is in the range of beacon frames. In this phase, we extract segments from R and construct a time-series sequence $S = \{s(1), s(2), ..., s(l)\}$, where $s(i)$ denotes the ith segment.

To find periodic beacon frames as well as identify their TBTTs accurately, we classify each segment into type 1 or type 2 according to the channel idle time before the segment starts (T_{idle}) as follows.

$$\begin{cases} \text{if } T_{\text{idle}} \geq T_{\text{backoff}} : \text{type-1 segment}, \\ \text{if } T_{\text{idle}} < T_{\text{backoff}} : \text{type-2 segment}, \end{cases} \quad (1)$$

where T_{backoff} is the maximum backoff time for a beacon transmission. Here, T_{backoff} is derived as

$$T_{\text{backoff}} = \text{DIFS} + T_{\text{slot}} \cdot CW_{min}, \quad (2)$$

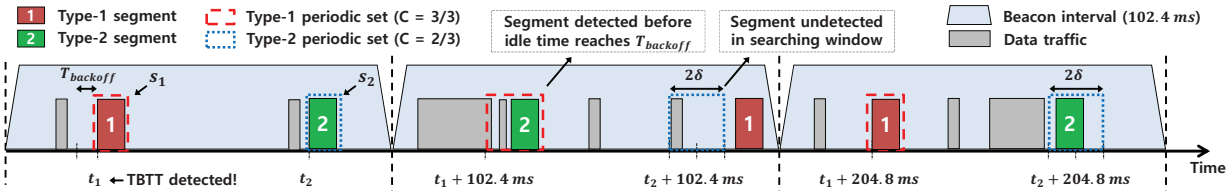


Fig. 3. Example of the periodicity searching algorithm (input RSSI samples: 3 (=N_{input}) beacon periods).

where distribute interframe space (DIFS), T_{slot}, and CW_{min} are 50 μs, 20 μs, and 31, respectively, in case of 802.11b. Note that in case of broadcast frames including beacon frames, CW is always set as CW_{min} since feedback cannot be obtained.

Consider a beacon transmission whose idle time T_{idle} is larger than T_{backoff}. We can conclude that this beacon frame transmission was initiated without backoff (because in case of backoff, T_{idle} is determined by the backoff time and thus is always smaller than T_{backoff}). More importantly, it implies that the beacon frame was transmitted exactly at its TBTT. Based on this observation, the segment type will be exploited to identify both beacon frames and TBTTs in the periodicity detection algorithm described below. For example, if a segment classified as type 1 is somehow identified as a beacon frame later, we can consider its starting point as TBTT.

Periodicity detection: Once the segments are extracted and classified, we next search 102.4 ms periodicity among the segments in S. To this end, we define segment periodicity and two types of periodic sets as follows.

Definition 1. (*Segment Periodicity*) A given set of segments, S, is said to be periodic with a period T and delay margin δ, if segment s ($\in S$) with the same length (i.e., airtime) exists "almost" every $[T-\delta, T+\delta]$ (ms) interval.

Definition 2. (*Type-1 Periodic Set*) Given a set S and a type-1 segment s_1 ($\in S$), let P_S^1 ($\subset S$) be a subset that contains s_1 and only the periodic segments such that every segment s ($\in P_S^1$) has the same length (i.e., airtime) and shows up within the range of $[t_1 + n \cdot T, t_1 + n \cdot T + \delta]$ (ms) for some integer n, where t_1 is the starting time of s_1.

Definition 3. (*Type-2 Periodic Set*) Given a set S and a type-2 segment s_2 ($\in S$), let P_S^2 ($\subset S$) be a subset that contains s_2 and only the periodic segments such that every segment s ($\in P_S^2$) has the same length (i.e., airtime) and shows up within the range of $[t_2 + n \cdot T - \delta, t_2 + n \cdot T + \delta]$ (ms) for some integer n, where t_2 is the starting time of s_2.

The basic idea is that if a type-1 segment is identified as a beacon frame, we can easily discover the other beacon frames (i.e., a periodic set P_S^1) in S. Therefore, we first search for type-1 periodic sets with which the TBTT can be obtained.

Fig. 3 illustrates our proposed periodicity detection method. Let s_1 denote a type-1 segment in S, and t_1 be its starting time. We perform the following to find subsequent periodic segments based on the hypothesis that s_1 is a beacon frame and t_1 is its TBTT. To find the nth subsequent periodic segment, with a searching window starting at $t_1 + 102.4 \cdot n$ (ms), we search for a segment s ($\in S$) whose length (i.e., airtime) is identical to s_1 until the total idle duration between $t_1 + 102.4 \cdot n$ (ms) and the starting time of s reaches T_{backoff},[3] as shown in Fig. 3. By grouping such segments, we construct a type-1 periodic set P_S^1 with a period $T = 102.4$ (ms).

Similarly, let t_2 denote the start of a type-2 segment s_2 ($\in S$). Based on the hypothesis that s_2 is a beacon frame, the interval between s_2 and its subsequent periodic segment may be shorter than 102.4 ms because of possible backoff delay which s_2 might have experienced. Therefore, starting from a type-2 segment, the nth subsequent periodic segments are searched within $[t_2 + 102.4 \cdot n - \delta, t_2 + 102.4 \cdot n + \delta]$ (ms) and added to a periodic set P_S^2, where the window size δ is chosen sufficiently large to account for the backoff delay.

As channel utilization increases, longer dwell time is needed to observe a beacon frame labeled as a type-1 segment. We adjust the dwell time depending on the channel utilization, using a threshold based classification of the periodic sets.

For each periodic set P_S, we define *confidence* to indicate how strong the periodicity of P_S is.

Definition 4. (*Confidence of a Periodic Set*) Given a periodic set P_S, its *confidence* $C(P_S)$ is defined as

$$C(P_S) = N_{set}/N_{input}, \tag{3}$$

where N_{set} denotes the number of detected segments in a periodic set P_S, and N_{input} denotes the number of beacon periods in the input RSSI samples S, i.e., maximum size of a periodic set.

A type-1 periodic set P_S^1 is labeled as 1) clear beacon frames ($Beacons$), 2) vague set ($Vague_{T1}$), meaning that we need more RSSI samples to confirm them as beacon frames, or 3) rejected, meaning that we conclude they are not beacon frames, based on the two thresholds α_1 and α_2 as follows.

$$\begin{cases} \text{if } C(P_S^1) \geq \alpha_1 : P_S^1 \text{ is a set of beacon frames,} \\ \text{if } \alpha_2 \leq C(P_S^1) \leq \alpha_1 : \text{sample more RSSI for decision,} \\ \text{if } C(P_S^1) \leq \alpha_2 : P_S^1 \text{ is not a set of beacon frames.} \end{cases} \tag{4}$$

Similarly, type-2 periodic set P_S^2 is labeled as 1) a vague set ($Vague_{T2}$), or 2) rejected, using the threshold α_1 as follows.

$$\begin{cases} \text{if } C(P_S^2) \geq \alpha_1 : \text{sample more RSSI for decision,} \\ \text{if } C(P_S^2) \leq \alpha_1 : P_S^2 \text{ is not a set of beacon frames.} \end{cases} \tag{5}$$

We only use the larger threshold (α_1) since it is more likely that data traffic may be falsely detected as a periodic segment due to large searching window δ.

[3]The rationale behind this is that the backoff counter is decreased only when the channel is sensed idle.

Algorithm 1 Beacon frame detection algorithm

Input: Target Bluetooth channel f for RSSI sampling
Output: List of periodic beacon frames B and corresponding TBTTs
Initialize: $R \leftarrow$ Sample RSSI on f for T_{min}, $DwellTime \leftarrow T_{min}$

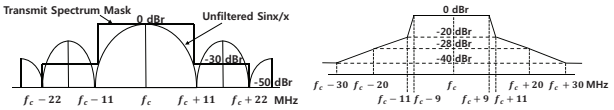
1: **while** $DwellTime < T_{max}$ **do**
2: $S = ExtractSegments(R)$
3: $(Beacons, Vague_{type1}) = DetectType1Periodicity(S)$
4: **if** $Beacons$ exist **then**
5: $B \leftarrow B \cup Beacons$
6: $R \leftarrow R - Beacons$
7: go to 3
8: **end**
9: $Vague_{type2} = DetectType2Periodicity(S)$
10: **if** $Vague_{type1}$ exists or $Vague_{type2}$ exists **then**
11: $R \leftarrow$Sample RSSI on f for additional 102.4 ms
12: $DwellTime \leftarrow DwellTime + 102.4\ ms$
13: go to 2
14: **else**
15: **break**
16: **end**
17: **end**

Based on the above classification of periodic sets, we determine whether more RSSI samples are required or not; if there exist any vague set(s), we obtain RSSI samples for additional beacon period (102.4 ms), append them to the input RSSI samples, and repeat the segment extraction and periodicity detection process until the dwell time reaches an upper bound (T_{max}). Otherwise, we finish RSSI sampling and move on to the next channel.

Overall algorithm flow: The run time algorithm of beacon frame detection algorithm on a single channel is illustrated in Algorithm 1. The algorithm starts by sampling RSSI on target channel for initial dwell time. The algorithm continues until dwell time reaches the maximum bound (line 1). From the RSSI samples, the segment extraction is performed (line 2), and then type-1 periodic sets and/or vague sets are obtained to examine the confidence of their periodicity (line 3). If the set is identified as a beacon frame set (based on (4)), it is removed from the original RSSI samples (by setting corresponding RSSI samples to the noise floor value) and type-1 periodic set detection is repeated until no more beacon frames are detected (lines 4–7). After all the beacon frames are detected and removed, type-2 periodic set is searched (line 9). If vague type-1 periodic set or type-2 periodic set exists, RSSI samples are collected for additional 102.4 ms (lines 10–13). Otherwise, the algorithm terminates and outputs the detected beacon frames and their corresponding TBTTs (line 15).

B. AP Channel Pinpointing

The main challenge in pinpointing the AP's operating channel (i.e., center frequency) is that every 2.4 GHz Wi-Fi channel overlaps with two to four adjacent channels. That is, the beacon frames of an AP are detected over multiple overlapping channels. Therefore, it is impossible to correctly identify the operating channel of the AP based on the beacon frame information obtained only on a single channel.



(a) IEEE 802.11b (b) IEEE 802.11g

Fig. 4. Transmit spectrum mask of IEEE 802.11.

The basic idea for AP channel pinpointing is that while the beacon frames from a single AP can be detected on its neighboring channels,[4] their TBTTs will be congruent to each other with respect to modulo 102.4 (ms), i.e., they have the same TBTT%102.4 value. That is, if we detect beacon frames in the whole 2.4 GHz band, we can pinpoint the channels of the APs by grouping the ones with the same TBTT%102.4.

The transmit spectrum mask of IEEE 802.11b/g shown in Fig. 4 gives a clue that an AP may be observable on two neighboring channels per each side. We observe in Section V that an AP on Wi-Fi channel i is observable on channels $(i-1)$ and $(i+1)$ with almost the same RSSI, while typically it is not observable on channels $(i-2)$ and $(i+2)$.

Let f_i (MHz) be the frequency of the Bluetooth channel that overlaps with the center frequency of Wi-Fi channel i, i.e.,

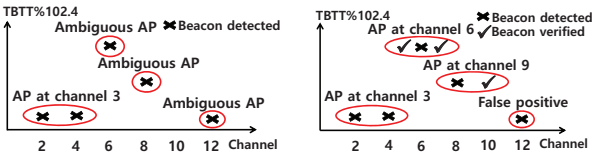
$$f_i = 2407 + 5 \cdot i. \tag{6}$$

The simplest approach in AP channel pinpointing is to detect beacon frames on all 13 f_i's[5]. However, it may incur high delay. To minimize the delay, we adopt *detect-and-verify process*. Since the APs on odd numbered Wi-Fi channels are observable on its neighboring even numbered channels, detecting beacon frames only on even numbered Wi-Fi channels is enough to cover the whole 2.4 GHz band. Therefore, we first search for beacon frames on even numbered channels (*detect*), and additionally search for beacon frames on necessary odd numbered channels to pinpoint the channels of the APs (*verify*). We verify in Section V that this approach can significantly reduce scanning delay without degrading performance.

Fig. 5 illustrates a scenario that can occur in the detect-and-verify process. Ideally, an AP operating on an odd numbered channel (e.g., channel 3) will be observed on two neighboring even numbered channels (e.g., channels 2 and 4). Therefore, when beacon frames with the same TBTT%102.4 are detected on two consecutive even numbered channels, we determine that the AP is on the odd numbered channel in the middle. For a set of beacon frames discovered on only one even numbered channel (lets say w), there are three possibilities: 1) the AP is on that even numbered channel (i.e., w), 2) the AP is on a neighboring odd numbered channel (i.e., $w-1$ or $w+1$), but beacon frame detection algorithm on the other neighboring even numbered channel failed to detect its beacon frames, or 3) it is a false positive. To correctly distinguish these three cases, we first search for beacon frames in RSSI samples already obtained during beacon frame detection algorithm on

[4]We refer to this as the AP is observable in neighboring channels.

[5]As beacon frames are transmitted only in 11b/g mode (20 MHz bandwidth), we refer to the channel as the 13 20 MHz channels in 2.4 GHz band.



(a) Before channel pinpointing (b) After channel pinpointing

Fig. 5. Channel pinpointing process.

neighboring even numbered channels. When searching for the beacon frames, we apply the lower confidence threshold (α_2) to account for the possibility that beacon frames were missed due to high threshold. If beacon frames are not detected, additional RSSI samples are obtained on neighboring odd numbered channels. If beacon frames are detected on the odd numbered channels, we determine that the channel of the AP is the middle. Otherwise, we consider them as a false positive.

C. Target AP Scanning

After detecting the TBTTs and operating channels of neighboring APs, BlueScan can choose between two efficient Wi-Fi scanning methods.[6] The first option is *selective active scanning*, where Wi-Fi triggers active scanning only on the channels where the APs are detected. The second option is *scheduled passive scanning*, where Wi-Fi triggers passive scanning on the channel where the AP operates at the exact TBTT to listen for the beacon frame.

The choice between the two scanning methods depends on the application. For applications that require seamless Wi-Fi connectivity (e.g., video streaming), selective active scanning can be used to minimize scanning delay and enable smooth handover. On the other hand, scheduled passive scanning may be adequate for applications sensitive to energy consumption (e.g., Wi-Fi-based localization). While scheduled passive scanning may incur larger delay than selective active scanning, it achieves the minimum Rx time needed for an STA to search for the AP (i.e., airtime of beacon frame). Additionally, non-intrusive scanning of STAs with scheduled passive scanning can be friendly to the network. Note that probe requests from aggressive active scanning of the STAs in dense environments are known to degrade the overall network throughput [2].

V. Performance Evaluation

A. Implementation

We implement BlueScan on an Ubertooth device [18], which incorporates a Bluetooth transceiver (CC2400 [13]). The Ubertooth device is connected to a laptop equipped with 802.11n AR9380 NIC via USB interface. We did not implement BlueScan on a smartphone since we are not allowed to read RSSI samples from the Bluetooth interface of a smartphone, which is proprietary. The overall structure of BlueScan is described in Fig. 6. Main handler code (Python 2.7.3) executed as the main process uses Python subprocess module, which enables C codes to run as its subprocesses,

[6]Depending on the application, multiple Wi-Fi scanning can be triggered using the AP list created by Bluetooth radio.

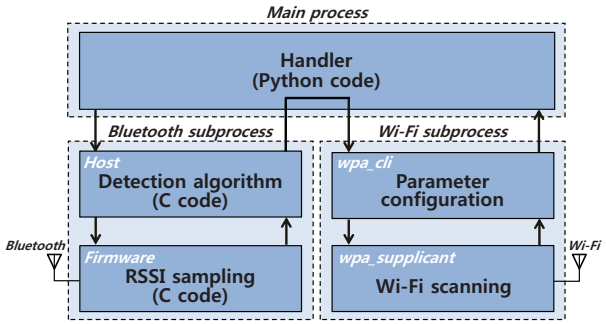


Fig. 6. Overall structure of BlueScan.

TABLE 1: Parameters for beacon frame detection algorithm

Parameter	Description	Value
$RSSI_{th}$	RSSI threshold in extracting segments	-85 dBm[7]
T_1	11b compliant beacon frame airtime minimum	1 ms
T_2	11b compliant beacon frame airtime maximum[8]	3 ms
α_1	confidence threshold for clear periodicity	0.75
α_2	confidence threshold for vague periodicity	0.5
δ	delay margin for searching type-2 periodicity	2 ms
T_{min}	minimum dwell time per channel	2×102.4 ms
T_{max}	maximum dwell time per channel	8×102.4 ms

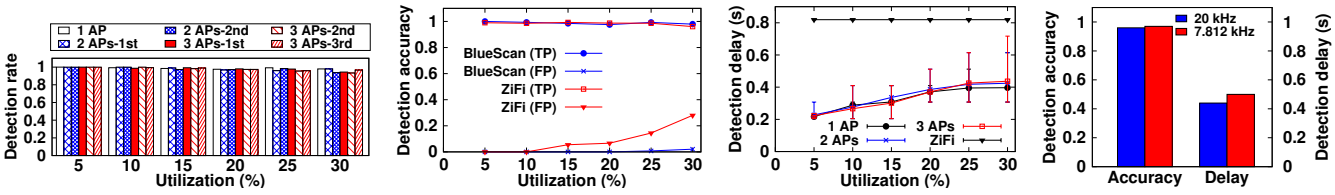
to manage the interaction between Ubertooth and Wi-Fi. We refer to the codes that control the operations of Ubertooth and Wi-Fi as Bluetooth and Wi-Fi subprocesses, respectively.

Before Wi-Fi scanning, the main process first opens the Bluetooth subprocess, where the firmware in Ubertooth samples RSSI every 50 μs. When 50 RSSI samples are collected, they are transferred to the host C code, where the detection algorithm is implemented, via USB. Every USB packet is marked with the firmware clock value at the time of creation. We utilize this value to estimate the timestamp of RSSI samples. For a USB packet with packet creation time T, the timestamp of the ith RSSI sample is marked as $T - 50 \cdot i$ (μs). After the host code processes the RSSI samples and detects neighboring APs, the list of APs is sent to the main process.

Upon Wi-Fi scanning request, the main process opens the Wi-Fi subprocess, implemented by modifying the source codes of *wpa_supplicant* and *wpa_cli*, or *wpa command line interface*, which delivers scanning requests (*selective active* or *scheduled passive*) to wpa_supplicant. Selective active scanning is triggered by configuring the scan frequencies in struct *wpa_supplicant*. Scheduled passive scanning is triggered by $wpa\_supplicant\_req\_scan()$ function, where scanning event can be scheduled in units of μs. To account for any delay between the scanning request and the actual scanning, scheduled passive scanning is triggered 10 ms earlier than TBTT, with scanning time 15 ms. Specific scanning parameters are configured in the device driver *ath9k* [19] and *mac80211* in backport 4.2.6-1.

[7]We set the RSSI threshold above noise floor to filter out the APs with weak signal strength. Not only the APs with low RSSI are of no worth detecting, intermittently detected beacon frames from such APs unnecessarily degrade the performance of beacon frame detection algorithm.

[8]We only evaluate our algorithm for 11b beacon frames. For 11g beacon frames, T_1 and T_2 should be modified accordingly. Though we expect similar performance for 11g beacon frames, further investigation is needed.



(a) Accuracy (b) Accuracy compared with ZiFi (c) Delay (d) Impact of sampling rate

Fig. 7. Performance of beacon frame detection algorithm.

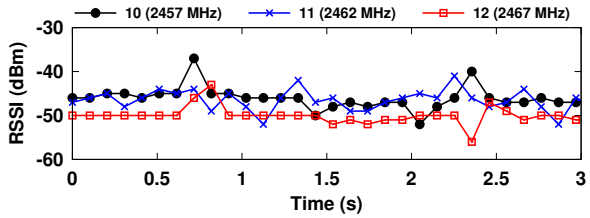


Fig. 8. RSSI of beacon frames (at channel 11) measured in different channels.

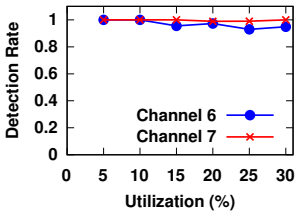


Fig. 9. Channel pinpointing accuracy.

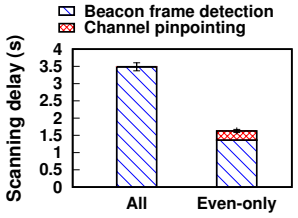


Fig. 10. Scanning delay of two channel pinpointing approaches.

As a comparison scheme, we implement the functional components of ZiFi [9] in host C code. We choose ZiFi as comparison scheme as it is the most representative work and employs the most similar approach to BlueScan in detecting Wi-Fi APs (searching for periodic beacon frames). Note that ZiFi does not handle AP channel pinpointing; it can only be compared with BlueScan in terms of beacon frame detection.

B. Performance of Beacon Frame Detection

Topology description: We deploy one to three off-the-shelf Wi-Fi APs (each transmitting 2 ms 802.11b beacon frame) with varying number of STAs (zero to three) to generate data traffic according to [7], stating that the typical channel utilization of a 2.4 GHz channel is around 20%, mostly from video streaming, file sharing, and web browsing. Similarly, [9, 16] observe that 2.4 GHz channel utilization is generally under 30%. Parameter setting for beacon frame detection algorithm is summarized in Table 1.

Results: Fig. 7(a) shows the detection rate of individual APs depending on the channel utilization and the number of APs on the same channel. Detection rate is the ratio of the events where the AP is successfully detected to the total number of detection events. The number of APs does not affect the performance significantly, because BlueScan detects beacon frames, removes them from the input RSSI samples, and repeats the process until no more APs are detected. With higher channel utilization, detection rate degrades due to periodicity distortion caused by backoff in beacon transmission.

Fig. 7(b) compares the performance of BlueScan and ZiFi in case where one AP is deployed. While both schemes achieve detection rate close to one, they show difference in terms of false positive (FP) rates. Since ZiFi detects beacon frames using only the periodicity signature, FP rate increases with channel utilization, as periodicity of data traffic increases. In contrast, BlueScan achieves FP rate close zero even with high channel utilization, since BlueScan considers both the periodicity of the beacon frames and the backoff operation.

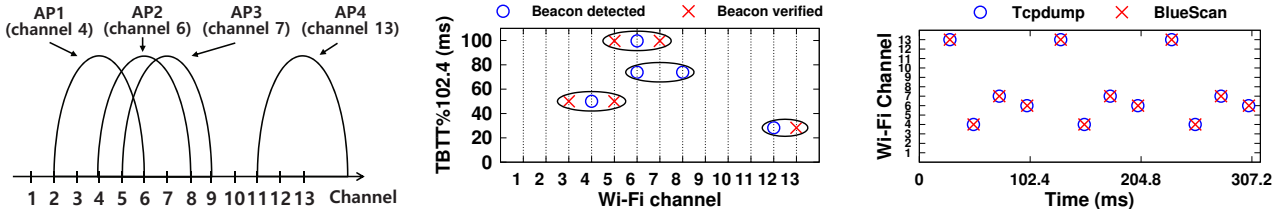
Fig. 7(c) shows the detection delay depending on channel utilization. Detection delay is defined as the total time needed to discover all the APs in the same channel. Error bar indicates the 90% percentile of delays. Compared to ZiFi [9] which employs a long, fixed RSSI sampling time (8 beacon periods), BlueScan efficiently adapts RSSI sampling time depending on channel utilization. When channel utilization is around 5 % (almost no traffic other than beacon frames), beacon frame detection algorithm mostly terminates in two beacon periods. As the channel utilization increases, however, detection delay increases since longer dwell time is needed to observe the beacon frame classified as a type-1 segment. The variation in detection delay increases with higher channel utilization since the detection algorithm highly depends on the traffic pattern.

Fig. 7(d) compares the performance of BlueScan with different RSSI sampling rates: 20 kHz and 7.812 kHz (maximum RSSI sampling rate of ZigBee radio). While the detection accuracy remains almost the same, faster RSSI sampling enables delicate segment extraction leading to increased efficiency in periodicity detection, thus reducing the detection delay. This indicates that Bluetooth radio with higher RSSI sampling rate enables a more agile detection algorithm than ZigBee radio.

C. Performance of Channel Pinpointing

RSSI of beacon frames detected on neighboring channels: Fig. 8 shows the RSSI of beacon frames from AP on channel 11 detected on the same and neighboring channels. Each point in the figure represents the average RSSI of each received beacon frame. We see that beacon frames are observable on channels 10 and 12 with almost the same RSSI, while they are not detectable on channels 9 and 13. Therefore, as mentioned in Section IV, detecting beacon frames on even numbered Wi-Fi channels is enough to cover the entire 2.4 GHz band.

Channel pinpointing accuracy: Fig. 9 shows the detection rate of two APs when they are partially overlapped on channel 6 and 7. Detection rate is the ratio of the events where the channel of the AP is pinpointed accurately to the total number



(a) Wi-Fi AP deployment (b) Result of *detect-and-verify process* (c) Detected TBTTs of APs

Fig. 11. Exemplary detection result of BlueScan.

of detection events. Although the detection rate for both APs remain high even with high channel utilization, there exists a slight gap between the two. The reason is because we detect beacon frames only on even numbered channels. For the AP on channel 7, there is a high chance that its channel is pinpointed because it is observable on both neighboring even numbered channels (i.e., 6 and 8). However, since the AP on channel 6 is observable only in the same channel, its channel cannot be pinpointed at all if beacon frames are missed on channel 6.

Channel pinpointing delay: Fig. 10 shows the average of 10 scanning delays needed to correctly pinpoint the channels of all three APs deployed on channels 2, 7, and 13, with error bar indicating the standard deviation. 'All' refers to a simple approach, where Bluetooth radio detects beacon frames on all 13 Wi-Fi channels and pinpoints APs' channels. 'Even only' refers to the proposed *detect-and-verify process*, where beacon frames are searched only on six even numbered Wi-Fi channels and necessary odd numbered channels for channel pinpointing. We observe that the proposed detect-and-verify process achieves about 54% less scanning delay.

D. Real World Experiment

Fig. 11 depicts a result of BlueScan conducted in a real-world environment where four APs are deployed as shown in Fig. 11(a). Fig. 11(b) illustrates how detect-and-verify process detects the TBTTs and channels of the APs. Fig. 11(c) shows that TBTT values estimated by BlueScan are highly accurate (with a maximum error below 300 μs) compared with the ground truth values, which is estimated by processing the timestamp of captured beacon frames using tcpdump.

We further evaluate BlueScan in various real world environments. Table 2 summarizes the result. 'Number of APs' denote the number of APs whose signal strength is above -80 dBm and stable to be detected by Ubertooth. Wi-Fi scanning delay is the total scanning time logged at wpa_supplicant. Bluetooth scanning delay is measured as the total time spent for the Bluetooth radio to detect neighboring APs, logged at the host code of Ubertooth. Legacy Wi-Fi scanning actively scans[9] all 13 channels in 2.4 GHz, where two probe requests are sent per each channel. Scanning time per channel is set to 40 ms, which is a common value [5]. For selective active scanning, scanning delay is proportional to the distinct number of channels where the APs exist, whereas in case of scheduled passive scanning, scanning delay is proportional to the number of APs. On average, selective active scanning and scheduled passive scanning reduces the scanning delay by 77% and 53% compared with legacy Wi-Fi scanning, respectively. We note that Wi-Fi remains in sleep mode for majority of the time in scheduled passive scanning. The time Wi-Fi is in Rx mode is reduced by 88% compared with legacy scanning, indicating a further energy saving. We also note that current implementation of scheduled passive scanning is an initial version where the scanning order of APs is not optimized. By considering the relative timing of TBTTs, the scanning delay can be further reduced with optimized scanning order.

TABLE 2: Real world experiment result

	Place	Resident	Library	Fitness center
	Number of APs	3	5	5
Legacy Wi-Fi scanning	Wi-Fi scanning delay	538.8 ms	574.8 ms	570.5 ms
	Probe requests	26	26	26
	Beacon frames	16	26	15
	Probe responses	9	15	21
	Bluetooth scanning delay	2.5 s	2.5 s	2.9 s
Selective active scanning	Wi-Fi scanning delay	130.9 ms	133.1 ms	129.4 ms
	Probe requests	6	6	6
	Beacon frames	3	7	7
	Probe responses	5	15	8
Scheduled passive scanning	Wi-Fi scanning delay	188.7 ms	305.1 ms	301.7 ms
	Wi-Fi Rx turn on time	44.7 ms	80.4 ms	80.3 ms
	Beacon frames	3	5	5

E. Energy Consumption Evaluation

Since we have not implemented BlueScan using smartphone, the expected energy consumption of the smartphone version of BlueScan is evaluated via energy consumption modeling based on the measurement results using a real smartphone device. We first measure Tx/Rx/idle[10] power consumption of Wi-Fi interface in Galaxy S5 smartphone (SM-G900) using Monsoon power monitor following the power measurement method in [20]. Then we measure the average power consumption of the smartphone with and without Bluetooth device discovery operation[11] and calculate the difference between the two values to extract power consumed solely by the Bluetooth interface. Table 3 summarizes the measured power consumption of Wi-Fi and Bluetooth interfaces.

The expected energy consumption of legacy Wi-Fi scanning, E_l, can be modeled as

$$E_l = N_{ch} \cdot (N_{req} \cdot \ T_{req} \cdot P_{w,Tx} + T_{sc} \cdot P_{w,idl}) \\ +(N_{resp} \cdot \ T_{resp} + N_{bc} \cdot \ T_{bc}) \cdot P_{w,Rx}, \quad (7)$$

[9] After analyzing various Android source codes, we verified that active scanning is the default for most Android smartphones.

[10] We assume the energy consumption of Wi-Fi in sleep mode is negligible.

[11] Here Bluetooth device discovery is set to use 100% duty cycle.

TABLE 3: Wi-Fi and Bluetooth power of SM-G900

Wi-Fi			Bluetooth
Tx	Rx	Idle	Rx
1768 mW	442 mW	350 mW	136 mW

where N_{ch} is 13, i.e., the number of Wi-Fi channels in 2.4 GHz band, N_{req} and T_{req} are the number of probe request frames sent per channel and their airtime, respectively. T_{sc} is the idle listening time of Wi-Fi during the scanning process. N_{resp} and N_{bc} denote the total numbers of probe response frames and beacon frames received on the entire Wi-Fi channels, respectively, and T_{resp} and T_{bc} are the airtime of those packets, respectively. $P_{w,Tx}$, $P_{w,Rx}$, and $P_{w,idl}$ are the Tx, Rx, and idle power consumption of Wi-Fi interface, respectively.

Similarly, the expected energy consumption of BlueScan with selective active scanning, $E_{p,a}$, becomes

$$E_{p,a} = T_{bt} \cdot P_{bt,Rx} + N_{ch,d} \cdot (N_{req} \cdot \ T_{req} \cdot P_{w,Tx} + T_{sc} \cdot P_{w,idl}) + (N_{resp} \cdot \ T_{resp} + N_{bc} \cdot \ T_{bc}) \cdot P_{w,Rx}, \quad (8)$$

where T_{bt} and $P_{bt,Rx}$ denote total detection delay of Bluetooth and Bluetooth Rx power, respectively. $N_{ch,d}$ denotes the number of distinct channels of Wi-Fi APs detected by BlueScan.

In the case of BlueScan with scheduled passive scanning, BlueScan only tries to listen for beacon frames. Accordingly, energy consumption of BlueScan with scheduled passive scanning, $E_{p,p}$, is expressed as

$$E_{p,p} = T_{bt} \cdot P_{bt,Rx} + N_{ap} \cdot T_{bc} \cdot P_{w,Rx}, \quad (9)$$

where N_{ap} is the number of Wi-Fi APs detected by BlueScan.

Fig. 12 shows the estimated energy consumption for legacy Wi-Fi scanning and BlueScan in the resident environment shown in Table 2, where the estimated values are obtained by substituting the results of power consumption measurement shown in Table 3 into (7) to (9). The x-axis indicates the number of full channel scanning events in case of legacy Wi-Fi scanning, and the number of scanning events (selective active or scheduled passive) in case of BlueScan. Fig. 12 demonstrates that once BlueScan creates the list of neighboring APs, substantial amount of energy can be saved in subsequent Wi-Fi scanning events compared with legacy Wi-Fi scanning. When the x-axis value is one, BlueScan consumes slightly more energy than the legacy scheme. This is because the energy consumption added by Bluetooth operation (as much as 0.024 mAh) is greater than that saved in Wi-Fi operation (76 % by selective active scanning and 93 % by scheduled passive scanning compared with the legacy Wi-Fi scanning).

Note that the measured power consumption of Bluetooth includes the energy consumed by other operations besides RSSI sampling (e.g., packet detection) such that the energy consumption of BlueScan in real implementation will be less than the modeling results. Parameter optimization of BlueScan (i.e., number of channels where RSSI sampling is conducted and dwell time) can also decrease the energy consumption, which will be included in our future work. Additionally, if a smartphone uses further energy-optimized Bluetooth Low Energy (BLE) chipset, BlueScan can use BLE radio to further reduce energy consumption.

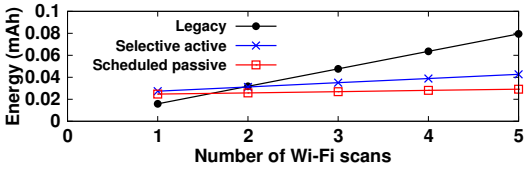


Fig. 12. Comparison of estimated energy consumption

VI. Conclusion

In this paper, we proposed BlueScan, a Bluetooth-aided Wi-Fi scanning scheme, which enhances the efficiency of Wi-Fi scanning by using the collocated Bluetooth radio to pinpoint the channels and TBTTs of neighboring APs. We demonstrate the effectiveness of BlueScan via real prototype implementation and experiments on Ubertooth platform. Our results have shown that BlueScan reduces scanning delay by up to 77% on average compared with legacy Wi-Fi scanning. BlueScan can be a novel cross-technology aided approach to support several Wi-Fi operations such as seamless handover. As future work, we plan to optimize the parameters in our algorithm, and further evaluate BlueScan for Wi-Fi handover.

VII. Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) [No.2017-0-01507, Developing ultrasound communication for an extensible 2nd-screen service to make set-tops VODs play by mobiles]

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021," Feb. 2017. [Online]. Available: http://www.cisco.com/go/vni
[2] Hu, Xueheng *et al.*, "Is there WiFi yet?: How aggressive probe requests deteriorate energy and throughput," in *Proc. ACM IMC*, 2015.
[3] H. Wu, K. Tan, J. Liu, and Y. Zhang, "Footprint: cellular assisted Wi-Fi ap discovery on mobile phones for energy saving," in *Proc. 4th ACM International Workshop on Experimental Evaluation and Characterization*, 2009.
[4] A. Anand, C. Manikopoulos, Q. Jones, and C. Borcea, "A quantitative analysis of power consumption for location-aware applications on smart phones," in *Proc. IEEE International Symposium on Industrial Electronics*, 2007.
[5] X. Chen and D. Qiao, "HaND: Fast handoff with null dwell time for IEEE 802.11 networks," in *Proc. IEEE INFOCOM*, 2010.
[6] A. Farshad, M. k. Marina, and F. Garcia, "Urban wifi characterization via mobile crowdsensing," in *Proc. IEEE NOMS*, 2014.
[7] Biswas, Sanjit *et al.*, "Large-scale measurements of wireless network behavior," in *Proc. ACM SIGCOMM*, 2015.
[8] Sui, Kaixin *et al.*, "Characterizing and improving wifi latency in large-scale operational networks," in *Proc. ACM MobiSys*, 2016.
[9] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma, "ZiFi: wireless LAN discovery via ZigBee interference signatures," in *Proc. ACM MobiCom*, 2010.
[10] Gao, Yuhang *et al.*, "ZiFind: exploiting cross-technology interference signatures for energy-efficient indoor localization," in *Proc. IEEE INFOCOM*, 2013.
[11] J. Choi, "WidthSense: Wi-Fi Discovery via Distance-Based Correlation Analysis," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 422–425, 2017.
[12] G. Ananthanarayanan and I. Stoica, "Blue-fi: enhancing Wi-Fi performance using bluetooth signals," in *Proc. ACM MobiSys*, 2009.
[13] Texas Instruments, "CC2400 2.4 GHz low-power RF transceiver," 2007.
[14] ——, "CC2420: 2.4 GHz IEEE 802.15. 4/ZigBee-ready RF transceiver," 2007.
[15] J. Ansari, A. Tobias, and M. Petri, "WiSpot: fast and reliable detection of Wi-Fi networks using IEEE 802.15. 4 radios," in *Proc. ACM MobiWac*, 2011.
[16] S. M. Kim and T. He, "FreeBee: Cross-technology Communication via Free Side-channel," in *Proc. ACM MobiCom*, 2015.
[17] T. Hao *et al.*, "Wizsync: Exploiting Wi-Fi infrastructure for clock synchronization in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 6, pp. 1379–1392, 2014.
[18] Ubertooth-one, http://ubertooth.sourceforge.net.
[19] Ath9k: Atheros Wireless Driver, http://wireless.kernel.org/en/users/Drivers/ath9k/.
[20] Koo, Jonghoe *et al.*, "PIMM: packet interval-based power modeling of multiple network interface-activated smartphones," in *Proc. ACM e-Energy*, 2015.