C-SCAN: Wi-Fi Scan Offloading via Collocated Low-Power Radios

Jonghwan Chung, Junhyun Park, *Member, IEEE*, Chong-Kwon Kim, and Jaehyuk Choi, *Member, IEEE*

***Abstract*—Wi-Fi channel scanning—the task of searching for available channels at a given location—is a fundamental feature to maintain always-available and high-quality wireless connectivity in today's mobile devices. However, it is a challenging task to design an intelligent scanning algorithm that can discover available access points (APs) in a short time period, because the scanning station has no prior knowledge on the APs in its vicinity. Therefore, traditional scanning algorithms seek to discover available APs by scanning the full set of channels, including channels where no APs exist. This often induces unnecessary scanning latency and/or energy consumption. In this paper, we present a novel scheme, called *C-SCAN*, that exploits a low-power wireless personal area network interface, such as Bluetooth or ZigBee integrated into the device to offload Wi-Fi scanning overhead by suppressing unnecessary scanning of AP-free Wi-Fi channels. To this end, *C-SCAN* inspects channel information with a low-power Bluetooth radio and identifies which Wi-Fi channels are in use, prior to the actual channel scanning with a Wi-Fi interface. By excluding the channels determined to be empty, the Wi-Fi scanning manager can perform scanning only on available Wi-Fi channels. Thereby, a significant performance gain in terms of delay and energy is obtained. We implement a prototype of *C-SCAN* using a Bluetooth-compliant wireless transceiver and demonstrate its efficiency. Experimental results show that *C-SCAN* achieves high detection accuracy with low latency, even in dense Wi-Fi environments.**

***Index Terms*—Bluetooth, energy efficient communication and networking for IoT, heterogeneous multi-interface radios, smartphones, Wi-Fi scanning.**

I. Introduction

WITH the increasing demand for high-speed and cost-effective Internet access, Wi-Fi has become the predominant wireless technology in modern mobile devices. According to Cisco's latest forecast [1], mobile data traffic has grown 18 times over the past five years and 60% of the total mobile data traffic in 2016 was offloaded onto the fixed network through Wi-Fi and/or femtocell. In fact, Wi-Fi connectivity has become a crucial requirement for wireless consumers to use data-hungry applications without worrying about their cellular data usage.

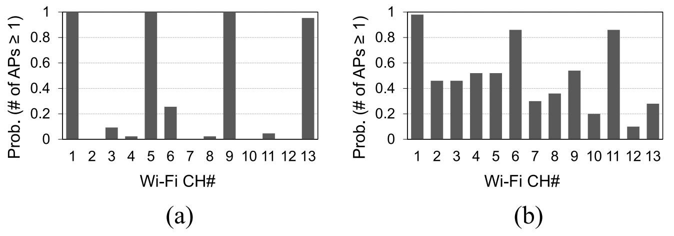


Fig. 1. Channel assignment status of Wi-Fi deployments measured in (a) 43 subway stations in which most APs are centrally managed, and in (b) local cafes, restaurants, and houses in which APs are managed individually.

The first step in delivering the benefits of using a Wi-Fi connection on mobile devices is to discover available Wi-Fi access points (APs) in the vicinity by *Wi-Fi scanning*. Wi-Fi scanning is the task of searching for available APs and their operation channels at a given location. It is a fundamental feature to provide adequate quality of experience requirements for various mobile applications, such as virtual reality and mobile media, where the demand for ensuring low-delay and seamless connectivity is important. To meet these requirements, however, frequent Wi-Fi scanning should be performed owing to user mobility and the short transmission range of Wi-Fi networks [2], which will lead to excessive battery drain.

An ideal scanning algorithm seeks to discover the maximum number of APs in the shortest period of time [3]. However, it is not an easy task to design such an optimal Wi-Fi scanning algorithm because the scanning station has no prior knowledge on the channel information of neighboring APs. Fig. 1 shows the empirical probability distribution that at least one AP is deployed in a given channel, measured in various places such as subway stations, cafes, restaurants, and houses. From the result, we can observe high variations in the AP deployments across channels. In order to obtain the AP and channel information, traditional scanning algorithms scan the full set of channels, including channels where no APs exist, spending a certain amount of time on each of them. As a result, this approach often leads to high scanning latency and/or energy consumption due to unnecessary scanning on the empty channels.

Manuscript received November 1, 2017; revised January 27, 2018 and February 19, 2018; accepted February 21, 2018. Date of publication March 1, 2018; date of current version April 10, 2018. This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2017R1D1A1B03034915, in part by the Korean Government (MSIP) under NRF Grant 2016R1A5A1012966 and NRF Grant NRF-2015R1A2A1A01007400, and in part by the Institute for Industrial Systems Innovation of Seoul National University. *(Jonghwan Chung and Junhyun Park are co-first authors.) (Corresponding author: Jaehyuk Choi.)*

J. Chung, J. Park, and C.-K. Kim are with the Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea (e-mail: chungjohn@snu.ac.kr; jhpark041@snu.ac.kr; ckim@snu.ac.kr).

J. Choi is with the Department of Software, Gachon University, Seongnam 13120, South Korea (e-mail: jchoi@gachon.ac.kr).

Digital Object Identifier 10.1109/JIOT.2018.2811240

In this paper, we present a novel approach for Wi-Fi channel scanning on mobile devices, called *C-SCAN*, that discovers available channels of nearby Wi-Fi APs in a fast and energy-efficient way. Unlike traditional approaches using a Wi-Fi interface, *C-SCAN* uses a low-power collocated wireless personal area network (WPAN) radio, particularly Bluetooth or BLE, coexisting in the scanning device for Wi-Fi channel scanning. *C-SCAN* exploits the fact that a Bluetooth radio can sense Wi-Fi signals in a frequency band (i.e., a set of consecutive Bluetooth channels) that overlaps with the Wi-Fi channel. By using the Bluetooth radio, *C-SCAN* identifies which Wi-Fi channels are used or not prior to the actual channel scanning with a Wi-Fi interface. By excluding the channels determined to be empty, Wi-Fi scanning manager then can perform *selective* scanning only on available Wi-Fi channels, thereby obtain significant performance gain in terms of delay and energy.

Although there have been many efforts [4]–[9] to assist Wi-Fi discovery by utilizing secondary coexisting WPAN radios, such as Bluetooth or ZigBee, none of them has the ability to provide channel information of available APs, in particular, their operating channels. As a result, they have to scan the full set of channels including the channels where no APs exist even after the availability of Wi-Fi systems is detected, thus spending a significant amount of time. Unlike the existing solutions, our proposed scheme *C-SCAN* is not only to predict the availability of Wi-Fi networks but also to pinpoint the available Wi-Fi channels by using collocated Bluetooth radios.

However, there are several challenges to achieve this goal because of the following factors.

1) *Large Search Space:* To scan the entire *wideband* Wi-Fi channel using *narrowband* Bluetooth radio, the entire search space increases (i.e., from 13 to 79 channels), which may induce significant scanning latency.
2) *Measurement Accuracy:* A practical challenge in the design of *C-SCAN* is the Bluetooth radio's channel sensing accuracy. Since the Bluetooth radio cannot decode Wi-Fi frames, we should rely only on the RSSI measurement. However, RSSI values are very noisy and are easily changed by multipath effects [10].
3) *Computational Cost:* The solution should be lightweight to be easily implemented and operated in resource-constrained devices, such as IoT devices, without considerable computational cost or high energy consumption [9].

To tackle these challenges, we present a lightweight RSSI sampling method for *C-SCAN* to minimize the number of sampling channels. In addition, we employ a min–max-based sample normalization and similarity analysis to investigate the correlation between RSSI samples, thus, making *C-SCAN* robust to RSSI variations. Further, we develop a simple binary string matching and scoring algorithm to make an accurate decision on the presence of APs on target channels.

Contributions: The main contributions of this paper can be summarized as follows.

1) Introduction of a new energy-efficient approach, called *C-SCAN*, that harnesses a low-power Bluetooth interface collocated in a mobile device to identify available Wi-Fi channels. Unlike traditional schemes using Wi-Fi interfaces, our scheme uses a Bluetooth radio, which is readily available in most modern mobile devices and enables low-delay and energy-efficient Wi-Fi scanning (Section IV).
2) Design of an intelligent channel identification algorithm that pinpoints the Wi-Fi channel numbers. Inspired by the observation obtained from a rigorous measurement study, we design the baseline to identify a target Wi-Fi channel based on the RSSI values measured over several narrowband Bluetooth channels (Section IV). In addition, by using min–max normalization and circular sampling methods, we enhance the baseline algorithm to minimize the number of sampling channels, thereby improving the scanning performance in terms of delay and energy consumption (Section V).
3) Implementation and evaluation of a prototype of *C-SCAN* algorithm. We demonstrate the feasibility and effectiveness of our approach by implementing the proposed *C-SCAN* using Ubertooth [11], an open source Bluetooth-compliant board, on an Android-based platform. The extensive experiment-based evaluation demonstrates that *C-SCAN* can accurately detect channels with high accuracy in realistic wireless environments (Sections VI and VII).

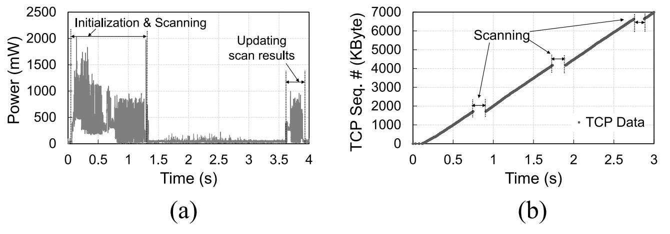


Fig. 2. Two performance issues in Wi-Fi scanning. (a) Energy trace of active Wi-Fi scanning. (b) TCP sequence number diagrams for periodic active Wi-Fi scanning (scanning interval is set to 1 s).

Paper Organization: The remainder of this paper is organized as follows. We summarize related research work in Section II. Section III describes the system overview of our approach. We explain the baseline design of *C-SCAN* in Section IV. In Section V, our baseline algorithm is optimized. Further, we discuss the implementation of our algorithm in Section VI. Section VII presents the evaluation results, and Section VIII concludes this paper and presents our future directions.

II. Related Work

Many articles and researches have reported lack of efficiency in Wi-Fi operations, which induce significant performance overhead and high energy consumption [12]–[18]. Our measurement results shown in Fig. 2(a) confirmed that the energy trace of Wi-Fi soars as active scanning is performed. Even when connected to an AP, periodic channel scanning is triggered in the range of 1, 5, and 10 min to ensure stable link quality [19]. This is not only a main culprit of energy consumption but also acts as an interruption to data transmission, directly affecting communication performance.

Most of traditional solutions consider a single type of interface; they use active (i.e., in use) Wi-Fi interfaces for probing Wi-Fi channels. Thus, while scanning candidate channels, Wi-Fi throughput degrades due to the halt of data transmission and overheads incurred by Wi-Fi scanning. From a measurement study, we observed periodic outage intervals in received TCP sequence numbers during Wi-Fi scanning, as depicted in Fig. 2(b). In the same context, Hu *et al.* [20] found that increasing the number of probe packets to coarsely discover neighboring APs causes significant overheads that hinder channel utilization in a static environment. Likewise, various methods have been studied to eliminate the overhead or to reduce unnecessary scanning trials.

A. Wi-Fi Channel Discovery

In 802.11 channel discovery process, related works fall into the following broad categories: 1) efficient use of Wi-Fi radios and 2) offloading[1] Wi-Fi scan procedure in an attempt to optimize performance and device power consumption.

The first category usually takes the approach of adjusting the Wi-Fi's scanning parameters. For an instance, the scanning intervals are adaptively set based on measured AP interarrival time, AP density, and user velocity [2], [21]. Velayos and Karlsson [22] theoretically explored the optimal values for MinCT and MaxCT. Recently, Wu *et al.* [23] and Xu *et al.* [24] suggested higher values for those timers. Furthermore, a method has been developed to set adaptive timer values other than fixed ones [3]. However, since scanning parameter-based methods do not fit well for all environments and users, the concept of selective scanning has become popular [25], [26]. In this method, Wi-Fi scans only a subset of channels with high probability of finding an AP based on experiences from previous scanning results, therefore, reducing the active duration of radio. Another strategy reported by Eriksson *et al.* [27], consists in a precomputed and stored probability of an AP operating in a channel to determine the scanning sequence. However in selective scanning, if any assumption for APs' presence is found to be wrong, it leads to full scan failure incurring additional costs. Additionally, efficient AP discovery methods aided by context information (e.g., sensory data [2], GPS [28]–[30], and cellular signals [31]) were introduced.

The second category takes an approach that offloads parts of scan procedure to avoid the inherently expensive nature of Wi-Fi. There are prior works which offload partial upper-layer protocol to hardware; TCP/IP stack [32], [33] and ARP and ICMP [34] were of concern to boost performance and save energy, respectively. Li *et al.* [35] identified that the main processor typically consumes 1–2 time more energy than the Wi-Fi radio during the scan procedure, and achieved power gain by offloading the tasks of the processor. These designs require a secondary processor to perform the offloaded tasks.

[1]In this paper, the term *offloading* refers to a computation offloading, i.e., reducing the burden of the standard Wi-Fi scanning's task, which distinguishes itself from data traffic offloading in cellular networks.

B. Multi-Radio Cooperation Technology

There have been many efforts to utilize cooperation techniques between multiple same-type radios. In multiple-input multiple-output system, diverse cooperative multiple antenna techniques were proposed for maximizing a spectrum gain [36]–[41]. The capability of spatial collaboration with the distributed antennas investigated in [36]–[38]. To leverage the spatial diversity, various network coding scheme for cooperative communication have been introduced: space-frequency code [39], space-time code [40], and rateless code [41]. In cognitive ratio network, relaying strategies were adapted for cooperation with other nodes in the vicinity [42]–[45].

Different from cooperation with the same type of radio, several approaches utilizing secondary coexisting radios, such as Bluetooth and ZigBee, have been proposed [4]–[9]. The idea of using a secondary low-power radio for saving device power consumption was first proposed in [5]. Wake-on-WLAN [46], S-WOW [47], and Essense [8] allow Zigbee and Wi-Fi radios to communicate with special codes and obtain channel information. To support direct communication between two different protocols, they require substantial amount of software and/or hardware modifications. Zi-Fi [4], Wake-On-Wireless [7], and Turducken [48] promote a hands-off listening approach to obtain networks information. Zi-Fi is designed to search for beacon frames solely by analyzing statistics and periodicity of RSSI samples. However, it is challenging to grab time periodicity of beacon frames in a short period of time and furthermore, energy signature created by beacons is susceptible to noise, making it a harder task. Choi [9] proposed a lightweight algorithm so called WidthSense to detect Wi-Fi signals via correlation analysis between the RSSI measurements obtained on two Bluetooth channels.

However, none of these existing approaches using collocated low-power WPAN radios [5]–[9] provides the detailed channel information of available APs in the vicinity, especially operating Wi-Fi channels. Our proposed scheme *C-SCAN* is also one of the kind that exploits secondary coexisting radio. Unlike these solutions, *C-SCAN* is not only to discover available Wi-Fi networks but also to provide the details of available Wi-Fi channels. Our approach may be viewed as advancing the solution discovering the availability of Wi-Fi by leveraging coexisting WPAN radios. It offloads scanning procedure of Wi-Fi and reliably detects channels with multiple active APs.

Furthermore, we expect that our approach can be jointly combined with solutions of low-power Wi-Fi, such as *E-MiLi* [49] and *Sampleless Wi-Fi* [50], to optimize Wi-Fi performance in terms of energy consumption. One example includes cooperation with *Sampleless Wi-Fi* [50]. By saving energy during channel-sensing (with *C-SCAN*) and energy during transmission (with *Sampleless Wi-Fi*), a significant reduction of Wi-Fi interface power is expected.

III. C-SCAN Overview

In this section, we outline the problem of Wi-Fi channel scanning and present an overview of our approach toward the problem.

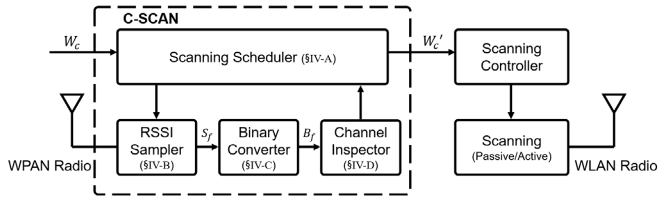


Fig. 3. *C-SCAN*'s high-level architecture.

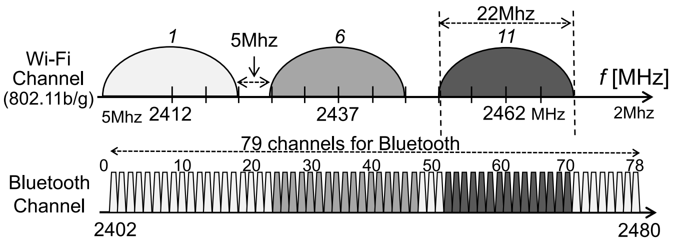


Fig. 4. IEEE 802.11b/g/n and Bluetooth frequency channels in the 2.4 GHz ISM band. Each 802.11 Wi-Fi channel (22 MHz wide) is composed of 20 Bluetooth channels (each 1 MHz wide) overlapping with a Wi-Fi channel.

A. Problem Statement and Overview

Fig. 3 illustrates the high-level overview of our approach. *C-SCAN* aims to identify available APs on target Wi-Fi channels by using a low-power Bluetooth interface—or an 802.15 WPAN-compliant radio, such as a CC2420 RF transceiver for ZigBee [51] coexisting in a device. We consider the problem of Wi-Fi channel scanning in modern mobile devices.

Basically, the target Wi-Fi channels are provided by the original Wi-Fi management module of the system, e.g., WiFiManager in the case of Android smartphones [52]. Let W_C denote a set of target channels. For a given W_C, *C-SCAN* inspects the presence of Wi-Fi signals based on the RSSI values obtained from the Bluetooth radio over the frequency range of target Wi-Fi channels. It first selects a Wi-Fi channel $w_i \in W_C$ and then tests the presence of an AP on w_i. When the operation of *C-SCAN* is complete, the scanning results, i.e., a set of discovered channels, denoted by $W_{\text{available}}$, are delivered to the Wi-Fi module. Then, the Wi-Fi scanning manager excludes the channels determined to be empty from W_C and constructs a set of new target channels, $W_C' \leftarrow W_C \cap W_{\text{available}}$. Thus, it can perform scanning only on available Wi-Fi channels.

Wi-Fi and Bluetooth are available on most modern mobile devices as their connectivity are becoming more common. Even entry-level phones have Bluetooth connectivity. Hence, we envision that the proposed method can be applied to various types of Wi-Fi enabled systems, including smartphones, wearables, and IoT devices.

B. Background and Notations

The 2.4 GHz ISM band is by far the most crowded unlicensed ISM band, occupied by various wireless technologies, such as IEEE 802.11b/g/n Wi-Fi, ZigBee, and Bluetooth [53]. Fig. 4 depicts the channel map allocated for Wi-Fi and Bluetooth in the 2.4 GHz ISM band. The IEEE 802.11b/g/n defines 13 channels within this band, numbered 1–13. Each possesses a bandwidth of 20/22 MHz and they are spaced 5 MHz apart. The Bluetooth protocol divides the band into 79 channels (each 1 MHz wide), operating from 2.402 to 2.4835 GHz.

In the rest of this paper, we use the notation w_i to represent a Wi-Fi channel i, and c_k to denote a Bluetooth channel k. For a Wi-Fi channel w and Bluetooth channel c, let $f_C(w)$ and $f_C(c)$ denote their center frequencies, respectively. Let $\mathcal{W}$ denote a set of entire Wi-Fi channels, i.e., $\mathcal{W} = \{w_1, w_2, \ldots, w_{13}\}$ and $\mathcal{C}$ represent a set of Bluetooth channels $\mathcal{C} = \{c_0, c_1, \ldots, c_{78}\}$. To describe the relation between Wi-Fi and Bluetooth channels, we define a *binary relation* R_{overlap} on a Cartesian product $\mathcal{W} \times \mathcal{C}$ as $R_{\text{overlap}} = \{(w, c) | \text{Wi-Fi channel } w \in \mathcal{W} \text{ is overlapping with Bluetooth channel } c \in \mathcal{C}\}$.

Definition 1 (Overlapping Relation): Given a Wi-Fi channel w, we define the set of all overlapping Bluetooth channels as $\mathcal{O}_c(w) = \{c | (w, c) \in R_{\text{overlap}}\}$, e.g., $\mathcal{O}_c(w_1) = \{c_0, c_1, c_2, \ldots, c_{19}\}$, as shown in Fig. 4.

Definition 2 (Channel Relation): Given a Wi-Fi channel $w_i \in \mathcal{W}$, we also define a mapping function $g_c(w_i)$ that converts w_i to the Bluetooth channel c_k whose center frequency $f_C(c_k)$ is identical to $f_C(w_i)$. We can easily derive $k = 5i + 5$ as shown in Fig. 4. For example, $g_c(w_6) = c_{35}$.

C. Channel Identification Using Bluetooth Radio

The effectiveness of our approach hinges on accurate detection of Wi-Fi signals and their frequency bands, i.e., Wi-Fi channels, via a Bluetooth radio interface.

1) Pilot Experiment: To study the feasibility, we conducted a pilot experiment that uses Ubertooth, an open source Bluetooth sniffer [11], to measure the RSSI values of wideband Wi-Fi signals over narrowband Bluetooth channels. For the measurement, we deployed one AP and one Wi-Fi client on channel 6, w_6, in a large underground parking lot where no radio interference signal in the 2.4 GHz band was present. We set the AP as a sender and generated UDP traffic using `iPerf3` with a constant packet generation rate of 5 Mb/s. For the frequency range of Wi-Fi channel w_6, i.e., $\mathcal{O}_{\text{map}}(w_6)$, the RSSI values are measured by the Bluetooth-compliant CC2400 wireless transceiver in the Ubertooth when the Wi-Fi signal is present. Hence, we designed the experiment to collect RSSI values on the Bluetooth channels $[c_{20}, c_{50}]$ (as depicted in Fig. 4) at four different distances (i.e., 1, 5, 15, and 35 m) from the AP. For data processing, we used a carrier sense (CS) threshold of -80 dBm to filter out low RSSI values.

2) Results and Observations: Fig. 5 plots the average RSSI for Wi-Fi frames sensed by a Bluetooth interface and a Wi-Fi interface. Fig. 6 plots the distributions of the RSSI values collected from the channels $\forall c \in [c_{20}, c_{50}]$. From Figs. 5 and 6, we make the following observations.

O1: Although a high degree of variation is observed in the RSSI values, it is clear that Bluetooth radios can sense Wi-Fi signals and their strength in the range of overlapping frequencies, i.e., $\mathcal{O}_c(w_6)$, in spite of different characteristics, such as transmission range and sensitivity. Further, as expected, we observe a wide "flat" section

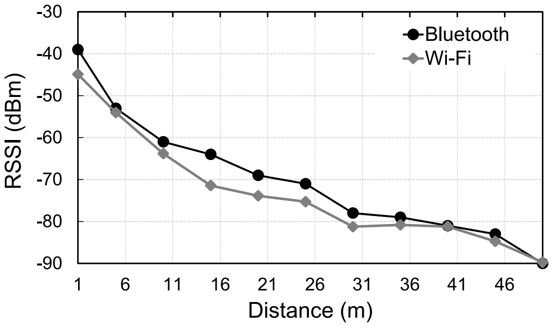


Fig. 5. Average RSSI for Wi-Fi frames sent from an AP at different distances, each sensed by a Bluetooth interface and a Wi-Fi interface.

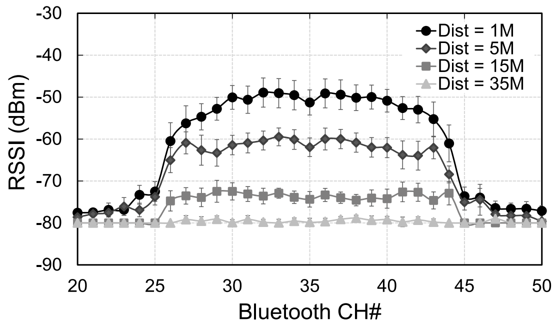


Fig. 6. Measurement results of RSSI values of Wi-Fi signals on channel 6 using Ubertooth (i.e., an open source Bluetooth sniffer) over narrowband 20–50 Bluetooth channels at different distances.

in the measured RSSI values, which correctly characterizes the coherence bandwidth[2] of Wi-Fi signals with the center frequency of channel w_6.

O2: The observed coherence bandwidth of Wi-Fi signals is larger than 10 MHz but smaller than 20 MHz. Therefore, the RSSI values measured on the three adjacent channels $w_{i-1(=5)}$, $w_{i(=6)}$, and $w_{i+1(=7)}$ spaced 5 MHz apart, have approximately identical distributions regardless of the distances. Meanwhile, although RSSI values greater than the CS threshold are observed on the channels 10 MHz apart from $f_C(w_6)$, their RSSI distributions have smaller values.

The above-mentioned observations inspire us to design an algorithm to pinpoint the operating channel number of Wi-Fi signals via a Bluetooth radio. Consider a Wi-Fi system (e.g., AP) operating on channel w. Clearly, in the range of Bluetooth channels $\mathcal{O}_c(w)$, a Bluetooth radio can sense its transmissions (e.g., data and beacon frames) and construct time-series of RSSI samples. Then, the RSSI sequences measured on two channels $c_1, c_2 \in \mathcal{O}_c(w)$ $(c_1 \neq c_2)$ will be strongly correlated (here, the degree of correlation between two time series will vary depending on the target channels). Therefore, by inspecting the RSSI sequences obtained on $\mathcal{O}_c(w)$ with a proper algorithm, we can identify the presence of any Wi-Fi system(s) on channel w.

Based on this motivation, we design our solution and explain the details in what follows.

[2]Coherence bandwidth is the range of frequencies over which the channel can be considered flat [54].

IV. C-Scan Design

In this section, we explain the details of *C-SCAN* based on its system architecture shown in Fig. 3.

A. Scanning Scheduler

Scanning scheduler is the main component of *C-SCAN* that manages flow control between components. Given a set of candidate Wi-Fi channels $W_C \subset \mathcal{W}$ received from the original scanning service[3] (e.g., WifiManager), *scanning scheduler* identifies the set of available channels $W_{\text{available}} \subset W_C$. To this end, it sets the sequence of target frequencies and executes *RSSI sampler* accordingly.

In *C-SCAN*'s baseline algorithm, all channels in W_C are scanned one by one to determine the presence of available APs. For example, for $W_C = \{w_4, w_6, w_1\}$ the scan is performed on the WPAN frequency band corresponding to w_4, and the process proceeds to channel w_6 and w_1 in order. Therefore, the algorithm is executed $|W_C|$ times.

When the scanning and inspection over target channels are complete, *scanning scheduler* generates a set of available channels $W_{\text{available}}$, a subset of W_C. The result is passed to the Wi-Fi scanning manager, as illustrated in Fig. 3. Hence, the manager performs scanning *selectively* with the set of new candidate channels $W'_C \leftarrow W_C \cap W_{\text{available}}$.

B. RSSI Sampler

RSSI sampler is invoked by *scanning scheduler* with a target Wi-Fi channel w, and it performs RSSI measurements on n scanning points $(f_1, f_2, \ldots, f_n) \in \mathcal{O}_c(w)$ via the collocated Bluetooth radio.

Fig. 7 shows the RSSI sampling mechanism adopted by *C-SCAN*. *RSSI sampler* measures RSSI values on a channel $f \in (f_1, f_2, \ldots, f_n)$ every τ and changes the channel from one to the next one in a circular fashion over the n scanning points. The parameter τ is the sampling period or slot time, and is set to 160 *us* in our implementation. Let $f(t)$ denote the sampling channel of *RSSI sampler* at time slot t, which is given by

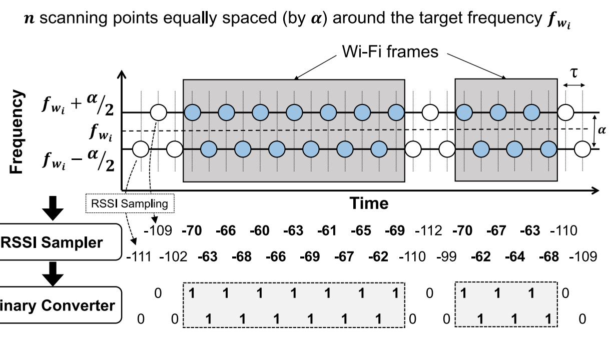
$$f(t) = f_{(t \text{ modulo } n)+1}$$

where $t = 0, 1, 2, \ldots$ and $f(t) \in (f_1, f_2, \ldots, f_n)$. A n-tuple of RSSI values, denoted by $(S_{f_1}, S_{f_2}, \ldots, S_{f_n})$, is generated and each element is updated every slot time τ. Here, S_{f_i} indicates an RSSI value measured on f_i. Then, the tuple is constantly fed to *C-SCAN*'s *channel inspector* for further examination.

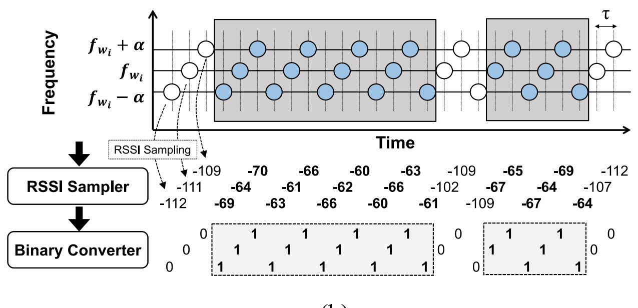
The parameter n should be at least two, whereas the minimum distance between each scanning point should be wider than 2 MHz to ensure Wi-Fi signal detection. Hence, unless the channel is overloaded, signals from narrowband protocols such as ZigBee or Bluetooth cannot affect more than two scanning points simultaneously. We design the *RSSI sampler* so that its scanning points are distributed around the center frequency of the target Wi-Fi channel with a constant interval, α, as depicted in Fig. 7.

In the baseline algorithm, the number of scanning points is set to the minimum value 2 for simplicity and the distance

[3]Due to the space limit, we omit the details on the scan procedure in the IEEE 802.11 standard. The interested reader can refer to [3] for further details.



(a)



(b)

Fig. 7. Basic operations of *C-SCAN* algorithm with two different numbers of scanning points (a) $n = 2$, and (b) $n = 3$, on target Wi-Fi channel $w_i \in W_C$. Series of raw RSSI data are collected by *RSSI sampler* in a circular fashion and processed by *binary converter*. Each binary digit, either 0 or 1, implies the presence of a signal at a corresponding scanning point. A binary string is a simplified form that summarizes the channel status.

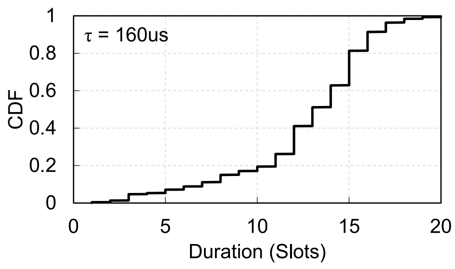
between them ($|f_2 - f_1|$) is set to 10 MHz. To be specific, when a target Wi-Fi channel i with center frequency f_{w_i} is passed down, a tuple of scanning points (f_1, f_2) is mapped onto $(f_{w_i} - 5, f_{w_i} + 5)$. Since the Wi-Fi channels are spaced 5 MHz apart from adjacent channels, the tuple can be represented as ($f_{w_{i-1}}$, $f_{w_{i+1}}$). The reason is to consider the effective bandwidth of the Wi-Fi's signal and make every scanning points independent of neighboring Wi-Fi channels' signals.

C. Binary Converter

Binary converter takes series of raw RSSI sample sequences and converts each value into a binary digit, 0 or 1, forming a binary string of length n: $(B_{f_1}, B_{f_2}, \ldots, B_{f_n})$ where B_{f_i} denotes a binary bit that represents the state of frequency f_i. If an RSSI value collected from a scanning point shows no traces of signals, the corresponding bit is set to 0 to indicate that the frequency is idle. Otherwise, if the sample exhibits any signal presence, the bit is set to 1 to indicate the busy state of the channel.

Although there are several ways to decide whether a frequency is idle or busy, the baseline algorithm chooses the most intuitive criteria: the CS threshold ($\mathrm{Th_{cs}}$) of -80 dBm. Specifically, a bit is set to 0 if the RSSI value is less than the CS threshold and 1 otherwise. The converted binary strings are passed to *channel inspector*.

Binary strings of length n have 2^n different combinations, but only small portions are meaningful. A converted binary string may indicate a sign of Wi-Fi signal when it contains



Slots	CDF
1	0.004199
2	0.013296
3	0.046886
4	0.053184
· · ·	
17	0.963611
18	0.983905
19	0.993002
20	1

Fig. 8. CDF of beacon lengths collected at various locations (e.g., subway, cafe, and office environments).

consecutive 1's. Since signals from a Wi-Fi channel range over multiple adjacent scanning points, strings with a single 1 or no 1's, and strings with no consecutive 1's are safely ignored during further AP detection process.

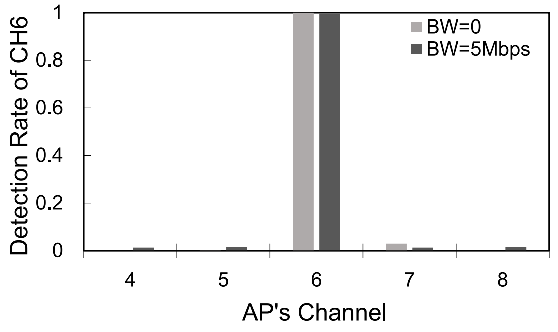
D. Channel Inspector

Channel inspector receives a series of length n binary strings to make the final decision on presence or absence of Wi-Fi APs in a channel. As mentioned, a binary string may indicate a sign of Wi-Fi signal only when it contains consecutive 1's. The baseline algorithm assumes that there is a Wi-Fi signal when both scanning points are considered busy (i.e., a binary string of "11"). We then can infer the length of a signal by looking at the number of successive signs. For an example, if a binary string 11 is observed three times in a row, the length of signal should be around $3 \cdot \tau$. *channel inspector* keeps track of the length of valid signals and counts occurrences of signals based on length, during T.

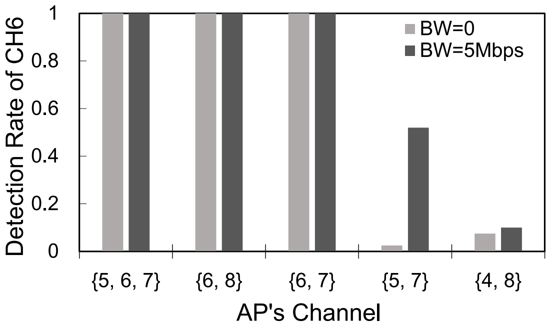
Short signals are likely to appear as noise in dynamic environments and, conversely, signals that last relatively longer imply that they are reliable signs of Wi-Fi APs. Therefore, we score the likelihood of AP presence in a channel by reflecting the weight in accordance with the length of the detected signals. The equation that scores the likelihood of AP presence on channel w_i is as follows:

$$\mathrm{Score}_{w_i} = \sum_{l=1}^{L} l \cdot N_{w_i}^{l}$$

where l denotes the length of a signal observed during T, and $N_{w_i}^{l}$ is the number of times w_i's signal with length l has been detected during the scanning period T. The larger the l, the more the count contributes to the final score. We obtain the threshold(θ) of the score to judge the AP presence from heuristic data we obtained from beacon length experiments, as shown in Fig. 8. Therefore, false negative (FN) rates are kept under 5 %. In the experiments, a number of beacons were collected from various places like subway stations, cafes, and offices to reflect real-world environments. The beacon length distribution is represented in form of a CDF. Since the majority of beacons prolonged on air fairly long, we safely assumed that the signals of length equal to or less than 4 slots are noises with high probability. A beacon is transmitted (T/beacon interval) times during the scan and hence, θ is set to $4 \cdot (T/\text{beacon interval})$. If the target Wi-Fi channel scores higher than θ, it is considered to contain at least one AP. *Channel inspector* passes the result



(a)



(b)

Fig. 9. Detection rate for channel 6 using the baseline algorithm in controlled environments (a) with a single AP located nearby, and (b) with multiple APs present on different channel combinations.

to *scanning scheduler* so that the target channel is included in the available channel set ($W_{\text{available}}$) sorted by its score.

E. Experiments

We first implemented the baseline algorithm on the application layer to validate the method for a massive RSSI dataset we collected from a set of controlled environments. To closely observe the behavior of the algorithm, we ran *C-SCAN*'s *RSSI sampler* using an Ubertooth transceiver in an anechoic chamber where wireless signals from outside are completely blocked. To cover different channel densities, traffic densities, and network topologies, we varied number of APs and mobile clients, traffic loads, and their placements in terms of distance. In a single-AP scenario, one AP was used to test the baseline algorithm varying few external environmental factors. The AP was connected with a laptop by wire to generate 0 or 5 Mb/s UDP traffic. Additionally, it was paired with a mobile client and their distance from the Ubertooth transceiver was varied {0, 5, and 15 m}. During the whole experiment, two scanning points of the *RSSI sampler* were fixed to detect the target Wi-Fi channel 6 while the AP was located on channel 5–7 in each single-AP scenario, respectively. In multiple-AP scenarios, two APs were located on channels {4, 8}, {5, 7}, {6, 7}, and {6, 8}, and three APs were deployed on channels {5, 6, 7}, again varying the UDP loads and distances. For each scenario, we ran the algorithm over the dataset 100 times and measured the average score and detection ratio.

Our experimental results (Fig. 9) show that the baseline *C-SCAN* can reliably identify the presence of an AP on a target channel and reject channels that involve no Wi-Fi activities under both light and heavy traffics. In a multiple-AP scenario, the baseline *C-SCAN* can always detect the AP on the target channel, as depicted in Fig. 9(b). The false positive detection rate (FPR) is kept relatively low, but it grows rapidly under environments where APs are closely located and traffics are heavily loaded.

F. Discussion

Although the experimental results seem promising in controlled environments, there are still challenges in exploiting the baseline algorithm in real-world environments.

1) O(n) Time Complexity: For a given target channel set W_C, the latency of the baseline *C-SCAN* algorithm, which works in the middle of existing Wi-Fi scanning procedure and stays on each channel $w \in W_C$ for T to check any AP presence, is not ideal for a scanning service that requires a certain degree of agility, because the scanning delay increases in proportion to the number of target channels $|W_C|$. If it incurs too much time overhead, incorporating *C-SCAN* has no merits.

2) Impact of Neighboring Interference: Signals from neighboring channels often affect scanning points located outside their effective bandwidth. Due to imperfections of hardware and different techniques of manufacturers, the Wi-Fi's signal tends to overflow out of its 20 MHz bandwidth, especially at close distances. If APs on adjacent channels release a signal that confuses both scanning points of the baseline algorithm, it will ring a false positive (FP) alarm in a situation in which no AP exists on the target channel. Even in the ideal situation where neighboring APs do not affect both scanning points, problems can still occur. In a lightweight traffic scenario, signals from different APs are unlikely to collide, potentially aided by CSMA/CA. However, the effect of CSMA/CA is limited (e.g., due to hidden terminal problem) and the problem gets worse in a heavy traffic scenario, inducing collisions between signals. When signals from neighboring APs located on the channel at both ends (i.e., w_{i-1} and w_{i+1}) of the target channel (w_i) coincide, both scanning points are affected and generate an FP detection of channel w_i.

V. Enhanced C-SCAN

In this section, we propose an enhanced version of the *C-SCAN* algorithm that performs low-latency operations and achieves excellent levels of stability in heavy-traffic scenarios with multiple APs.

A. Minimization of Scanning Points

In the baseline algorithm of *C-SCAN*, we use the number of scanning points $n = 2$ for RSSI sampling on each Wi-Fi channel, which is the minimum number required to verify the presence of an AP on a *single* Wi-Fi channel. However, as described above, this approach may suffer from the increased scanning delay when the number of target channels is large.

Our key observation to minimize the scanning latency is that, with a certain degree of scanning points, e.g., $n = 3$,

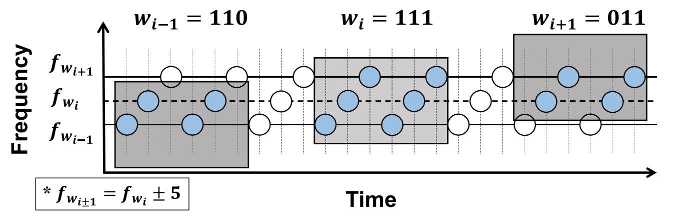


Fig. 10. Multiple channel scan with binary string classification in the enhanced *C-SCAN* algorithm.

we can inspect more than one channel at a time. For example, consider a situation where *C-SCAN* scans and inspects a Wi-Fi channel w_i with three scanning points ($f_{w_{i-1}}, f_{w_i}, f_{w_{i+1}}$) for a given target channel w_i as shown in Fig. 10. *RSSI sampler* will generate a sequence of RSSI samples ($S_{f_{w_{i-1}}}, S_{f_{w_i}}, S_{f_{w_{i+1}}}$). In an ideal case, a sequence of consecutive binary stings of "110," "111," and "011" imply the presence of APs on channels w_{i-1}, w_i, and w_{i+1}, respectively. This implies that we can monitor three consecutive Wi-Fi channels, i.e., w_{i-1}, w_i, and w_{i+1}, with a single execution of *C-SCAN*.

It is possible to monitor more channels at once by increasing the number of scanning points. However, it is not cost-free and there is a tradeoff between monitoring granularity and monitoring scope. Since we are trying to collect samples over multiple frequencies using a single radio, if the number of frequencies to be hopped increases, the sampling period regarding each scanning point also increases naturally. This in turn reduces the monitoring granularity. Specifically, given a sampling period of a WPAN radio, the slot time τ, it takes $n \cdot \tau$ to collect two consecutive RSSI samples from a single target frequency. If signals of lengths shorter than $n \cdot \tau$ move in the air, they are unlikely to be detected. Thus, adopting large n may cause too many signals uncatchable, raising FN rates.

Base on this reason, we choose $n = 3$ for the enhanced algorithm since it holds a good balance between monitoring granularity and monitoring scope. To inspect all candidate Wi-Fi channels $\forall w \in W_C$, *scanning scheduler* bundles as many groups of three consecutive Wi-Fi channels as possible, and schedules *C-SCAN* operations at the center frequencies of the center channels belonging to three Wi-Fi channels. For each iteration, *RSSI sampler* maps three scanning points (f_1, f_2, f_3) onto ($f_{w_{i-1}}, f_{w_i}, f_{w_{i+1}}$) for given target channel w_i, and generates a sequence of RSSI samples ($S_{f_{w_{i-1}}}, S_{f_{w_i}}, S_{f_{w_{i+1}}}$) by alternating between three sampling points in the same circular manner as depicted in Fig. 10. Note that the scanning points are now spaced 5 MHz apart from their neighbors (the 10 MHz space was used in the baseline algorithm).

B. Sample Normalization and Similarity Analysis

One challenging issue to realize this approach is that APs on neighbor channels can cause FP alarms. For example, although Wi-Fi signals are on channel w_{i-1} (e.g., the first gray box in Fig. 12), all RSSI values measured on three sampling points ($f_{w_{i-1}}, f_{w_i}, f_{w_{i+1}}$) could be above a predefined threshold, e.g., -80 dBm due to the time-varying nature of wireless media or co-channel interference. Then, its binary string will be represented as 111 instead of 110, which is falsely recognized as w_i.

To address this issue, we present a min–max-based sample normalization and similarity analysis method. As observed from our pilot experiments in Section III-C, the coherence bandwidth of Wi-Fi signals is larger than 10 MHz but smaller than 20 MHz. To exploit this, we analyze the similarity between RSSI values in a sample ($S_{f_{w_{i-1}}}, S_{f_{w_i}}, S_{f_{w_{i+1}}}$) to generate a binary string from the measured RSSI values, instead of simply comparing their values to a predetermined threshold. Prior to similarity check, we normalize them in order to alleviate the effect of RSSI variation caused by differing distances to the APs. Among the various normalization techniques in data mining [55], *min–max*-based normalization better conforms with the requirements of our implementation for the following reasons.

1) *Min–max* algorithm, also known as *feature scaling*, is suitable for standardizing features of data whose range varies widely. It correctly identifies the frequency-domain similarity between n scanning points, where overlapping wideband Wi-Fi channels with different distance provoke different ranges of RSSI data.
2) It provides computational simplicity since the sample size is always fixed at n. Normalizing RSSI values in each sample is done extremely fast and efficiently, which is a crucial property in detecting Wi-Fi signals in real-time.
3) Noisy RSSI signals are effectively ignored and their influence is minimized to ensure stability, since normalization and similarity comparison only focus on good signals, i.e., signals with RSSI values above the CS threshold (−80 dBm).

The threshold value was carefully chosen from our pilot experiment Section III-C. In evaluation part, we verify that the threshold of −80 dBm can effectively find nearly all available APs in vincinity. Few APs, which *C-SCAN* could not detect, were neither found by original Wi-Fi scan nor usable due to bad link quality. The min–max-based normalization can be written as

$$v' = (v - \text{Min})/(\text{Max} - \text{Min})$$

where v is a raw RSSI value in a sample, and v' is normalized value of range [0, 1]. RSSI min and max values are determined as follows:

$$\text{Min} = \begin{cases} \text{-80 dBm}, & S_i > -80; \quad \forall 1 \le i \le n \\ \min(S_1, S_2, \dots, S_n), & \text{otherwise} \end{cases}$$
$$\text{Max} = \max(S_1, S_2, \dots, S_n).$$

Binary converter confirms with a similarity check if active signals are actually from Wi-Fi channels or if they are simply composite noises. Similarity between two RSSI values within a sample is simply computed by the following equation:

$$\text{Sim}(S_i, S_j) = 1 - |S_i - S_j|.$$

Note that the process of similarity comparison may vary depending on the number of scanning points, n. Here, we focus

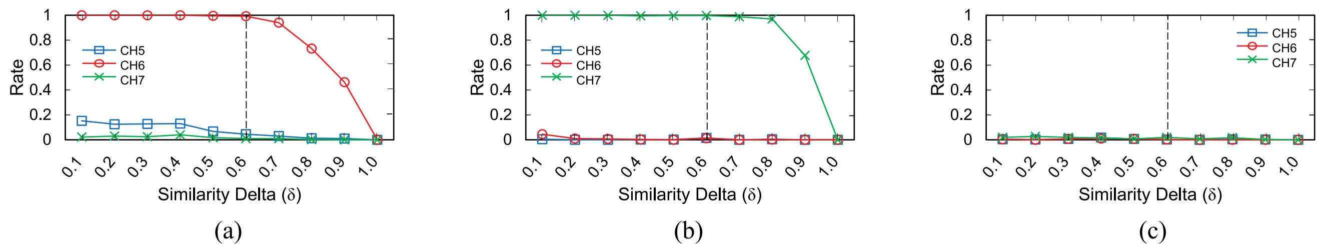


Fig. 11. Detection performance of enhanced *C-SCAN* in TPR and FPR with varying similarity δ. (a) TPR of CH6 and FPR of CH5 and CH7 when AP is on channel 6. (b) TPR of CH7 and FPR of CH5 and CH6 when AP is on channel 7. (c) FPR of CH5, CH6, and CH7 when AP is on channel 8.

on the case $n = 3$ for the enhanced algorithm. Signals with mask 110, 111, and 011 imply the presence of APs on channels w_{i-1}, w_i, and w_{i+1}, respectively, for a given target channel w_i. When all RSSI values are above -80 dBm, the similarity check is done for all three possible pairs: if $\text{Sim}(S_1, S_2)$, $\text{Sim}(S_2, S_3)$, and $\text{Sim}(S_1, S_3) \geq \delta$, with δ as a predefined threshold, it is confirmed to be a binary string of 111. if the first or last two RSSI values are above -80 dBm, the similarity check $\text{Sim}(S_1, S_2) \geq \delta$ is done for 110 and $\text{Sim}(S_2, S_3) \geq \delta$ is done for 011. This step alleviates FP alarms caused by neighboring channel dependency. Among the available signal masks 110, 111, and 011 in the *binary converter*, only the ones confirmed to be similar are counted for scoring in the next step.

To search for the optimal value of the threshold δ, we measured the detection performance of enhanced *C-SCAN* in true positive rate (TPR) and FP rate (FPR), as shown in Fig. 11. Under controlled environments, where an AP was located at the target channel, a channel away from the target, and two channels away from the target, the similarity δ of value 0.6 showed the highest TPR as well as the lowest FPR. Hence, in subsequent performance evaluations of enhanced *C-SCAN*, the threshold value δ is set to 0.6.

C. Results

Under the same controlled environments as in the baseline *C-SCAN* algorithm, including both single- and multiple-AP scenarios with varying bandwidth and distance, we again ran *RSSI sampler* using an Ubertooth transceiver on the target Wi-Fi channel 6. Then, enhanced *C-SCAN* is shown that the neighboring channels, 5 and 7, can also be monitored. In Fig. 12(a), the results show that the enhanced *C-SCAN* can cover three channels at a time, reliably identifying the presence of an AP on a target channel and rejecting channels that involve no Wi-Fi activities, under both light and heavy traffic. The enhanced *C-SCAN* achieves high detection performance even in the multiple-AP scenario. The FPR is kept low in most cases, but it increases lightly under the environments in which APs are closely located and heavy traffic appears. Note that unlike the real-world environment where APs are randomly distributed, the distances between APs and Ubertooth transceiver are kept approximately constant in multiple-AP controlled environments, which makes it difficult to perform accurate similarity analysis. In Section VII, we evaluate the performance of enhanced *C-SCAN* in real-world environments and show that FPR is effectively mitigated by a similarity analysis.

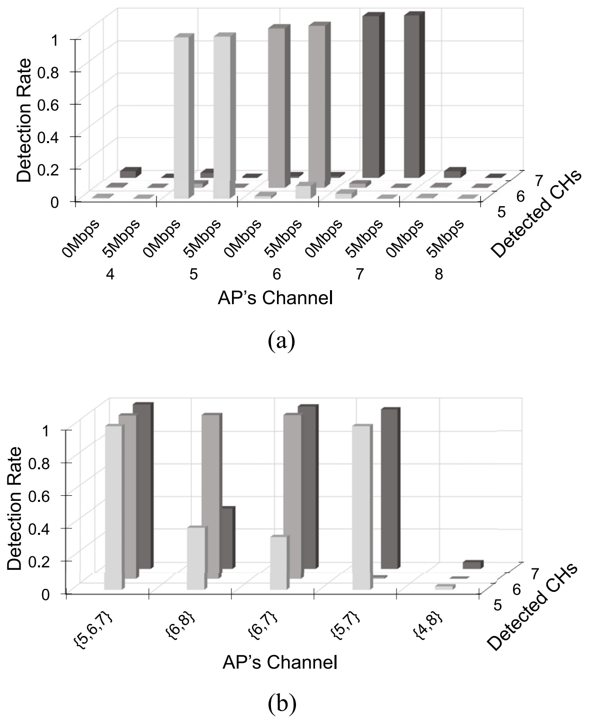


Fig. 12. Detection rate of channels 5–7 using the enhanced algorithm in controlled environments (a) with a single-AP located nearby, and (b) with multiple-APs present on different channel combinations.

VI. Implementation of C-SCAN in Android

We implemented a prototype of *C-SCAN* by using Ubertooth One [11], an open-source 2.4 GHz wireless development board, in Android devices. We developed all components described in Sections IV and V as Android application and/or Ubertooth firmware. For the target mobile device, we used an Android tablet (Google Nexus 7 - 2nd version) equipped with a Qualcomm Atheros WCN3660 Wi-Fi chipset. Then, Ubertooth was attached to the tablet via a Micro USB Gender, as shown in Fig. 13.

Fig. 14 depicts the implementation architecture of *C-SCAN* on Android platform. Two core components of it, *scanning scheduler* and *channel inspector*, are implemented in Java and run in a standalone Android application. By implementing an additional integrated controller on the application layer, we enable *scanning scheduler* to interact with Ubertooth and the Wi-Fi chipset through the USB interface and Android's functional API, respectively. *scanning scheduler* obtains a list of

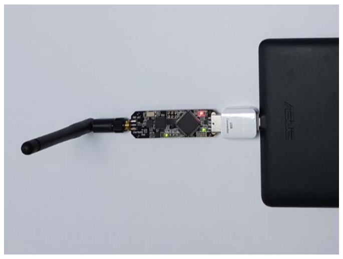


Fig. 13. Prototype of *C-SCAN* that consists of a tablet and Ubertooth connected through a micro OTG gender.

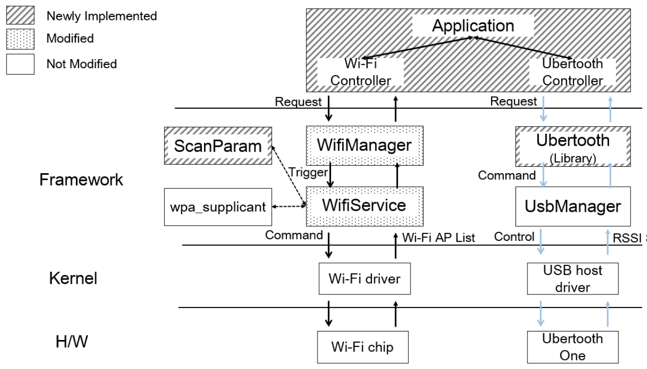


Fig. 14. Implementation architecture of *C-SCAN* on Android. It periodically monitors nearby Wi-Fi APs via WPAN radio and performs energy-efficient Wi-Fi scanning.

target scanning channels from the system configuration file or user specification. It invokes *RSSI sampler* in the firmware of Ubertooth via the USB interface with a set of target channels and the sampling period as parameters.

In addition, we modified Android's scanning module at the framework layer to implement *selective scanning*. By default, Android is designed to scan the full set of Wi-Fi channels, and the related scan parameters defined in `wpa_supplicant`, such as interval and timeout values, are fixed and not configurable at run-time. We added new features to the framework layer to scan only the specified set of Wi-Fi channels using user-configurable scan parameters, instead of the entire channels. *RSSI sampler* is implemented in the firmware of Ubertooth. We extend the original firmware to accomplish circular channel hopping and RSSI sampling. In particular, the sampler reads an RSSI value from the CC2400 transceiver every $\tau = 160$ us and hops to the next channel. The sampler stores the RSSI samples into a buffer and transfers them to *channel inspector* via the USB interface.

VII. Performance Evaluation

In this section, we present the performance evaluation results of two *C-SCAN* algorithms: 1) *baseline* and 2) *enhanced*. We compare the performance of *C-SCAN* with two Wi-Fi standard scanning methods: 1) active and 2) passive. *Legacy* denotes the original Wi-Fi scan method of an Android device. We first describe the experimental setups and parameters, and then discuss the results in terms of various performance metrics: *detection accuracy*, *detection latency*, *energy consumption*, and *throughput*.

TABLE I
Parameters Settings for *C-SCAN*

Parameter	Value
Carrier Sensing (CS) Threshold (Th_{cs})	-80 dBm
Sampling Timer (T)	102.4 ms
Maximum Round (α)	3
Similarity Delta (δ)	0.6
Theta (θ)	10
Scan Interval (I)	10 s
MinChTime ($MinChTime$)	10 ms
MaxChTime ($MaxChTime$)	102.4 ms

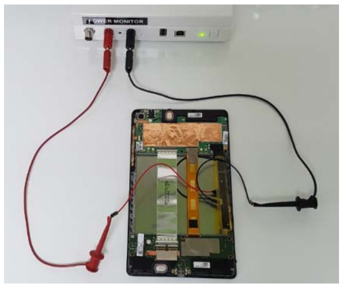


Fig. 15. Setup of energy measurements for Wi-Fi scanning using the Nexus 7. We detached the control chipset from the battery and connected a Monsoon power monitor.

A. Experimental Setups and Parameters

In order to demonstrate the feasibility of our solution, we conduct experiments in real-world environments with various channels conditions and network topologies. We carried out the experiments in lots of places, including subway platforms, cafes, and office buildings in Seoul, South Korea, where the channel distributions of deployed APs are very diverse. Depending on the degree of channel occupancy, the experiment results were classified into three states: 1) *sparse*; 2) *office*; and 3) *dense*. *Sparse* represents an environment where 10% of all Wi-Fi channels are occupied by nearby APs. The percentage of occupied channels in *dense* environments are 50% or higher, and *office* environments have a percentage between 30% and 50%.

We configure system parameters of *C-SCAN* with reference to the Android devices' default settings. The details of parameters for Ubertooth and Wi-Fi are given in Table I.

B. Detection Accuracy

First, we evaluate the detection accuracy of the two algorithms (*baseline* and *enhanced*). We conducted experiments in multiple places and repeated them 100 times at each place. For the evaluation, we use two standard accuracy metrics: 1) FP and 2) FN. We define an FP as the result when *C-SCAN* detects an active channel when there is no AP on the target channels. An FN is defined as the result when *C-SCAN* incorrectly indicates that a channel is inactive (i.e., no APs detected) but at least one AP exists on the channel.

Fig. 16 illustrates the detection accuracy of the *baseline* and *enhanced* algorithms based on FP and FN rates for different channel conditions depending on the occupancy of nearby Wi-Fi APs: *sparse*, *office*, and *dense*. The result shows that

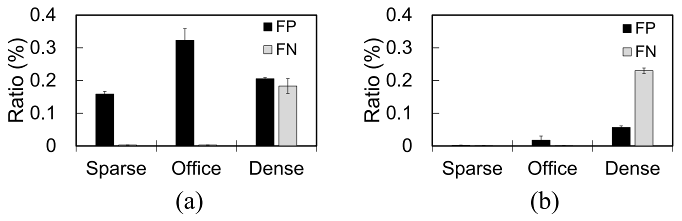


Fig. 16. Detection accuracy of *C-SCAN* algorithms. (a) Baseline. (b) Enhanced.

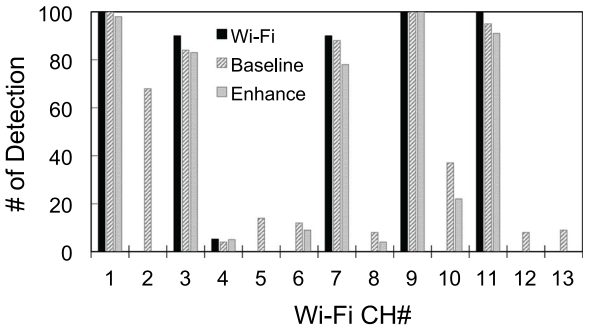


Fig. 17. Comparison of AP detection counts between Wi-Fi interface and C-SCAN (basedline and enhanced) in dense environment. We conducted the same experiment for inspecting all Wi-Fi channels 100 times.

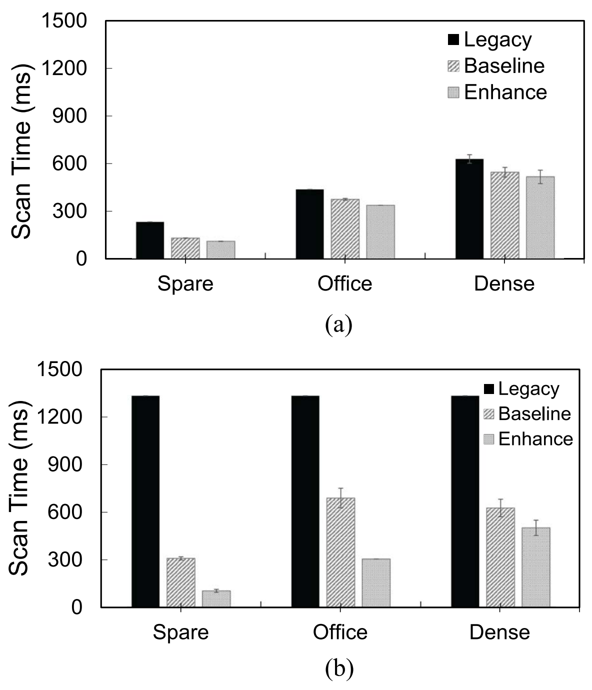


Fig. 18. Detection latency for Wi-Fi scanning: *C-SCAN* versus legacy. (a) Active. (b) Passive.

both algorithms achieve very low FN rates (approximately zero) in sparse and office environments. Further, the results also show that the overall error rate of the baseline algorithm, as expected, is greater than that of the enhanced algorithm because of its simple binary converter which causes FPs on adjacent channels. The FP rate of the enhanced algorithm is significantly decreased compared with that of the baseline algorithm: 16% (sparse), 30% (office), and 15% (dense) on average.

In dense environments, however, both fail to maintain accuracy. This is mainly due to the occurrence of FNs on some active channels. On these channels, there were very few frames whose signal strengths are stronger than the CS threshold.

Fig. 17 shows the cause of poor performance in dense environments. We gathered AP scan results using the Wi-Fi interface at the same places and compared the number of detected Wi-Fi APs on each channel: 1) Wi-Fi interface; 2) baseline algorithm; and 3) enhanced algorithm via the Bluetooth-compliant transceiver. During experiments, the number of identified APs varied according to the received signal strength of frames from APs on each channel. In this environment, six channels are occupied by nearby APs : 1, 3, 4, 7, 9, and 11. In particular, on channel 4, the average number of APs being identified via the Wi-Fi interface is only 5.25. Compared with the scan results of the Wi-Fi interface, the average FN rates of baseline and enhanced algorithms decrease from 18.3% to 10.4% and from 23% to 15.3%, respectively.

C. Detection Latency

Detection latency is one of the most important performance metrics in Wi-Fi scanning. We measured the total time taken to obtain a scanning result of *C-SCAN*, i.e., the Wi-Fi channel information of vicinity. In particular, latency is defined as the time difference between when *C-SCAN* starts channel scanning for all Wi-Fi channels and the end. Since *C-SCAN* operates the channel discovery until the sampling timer expires and gives several tries to identify the absence of AP(s) if no Wi-Fi signal is detected on the channel, we compared the difference between the latency of baseline and enhanced algorithm.

We first compared the delay performance of *C-SCAN* with *legacy*. The results are shown in Fig. 18. For active Wi-Fi scanning [Fig. 18(a)], the scan times of both *C-SCAN* algorithms are shorter than that of *legacy* because the first do not send unnecessary probe requests on channels where no AP exists. We observed that the latency of the enhanced algorithm is shortest due to low FPs that occur on adjacent channels. Similarly, when the Wi-Fi interface does AP scanning passively [Fig. 18(b)], the latency of *C-SCAN* is significantly reduced in all experimental cases. Note that the performance gain on latency is noticeable in passive scanning, which needs a much longer waiting time than active scanning to listen to periodic beacon frames.

We investigated how the enhanced algorithm improves performance in terms of latency compared to the baseline algorithm. Fig. 19 depicts the cumulative distribution of the detection latency difference of the baseline and enhanced algorithm for different levels of channel occupancy. The latency gap implies the reduced detection time by adopting the enhanced algorithm instead of the baseline algorithm. The negative gap suggests that the detection time of the enhanced algorithm is faster than that of baseline algorithm. Our results indicate that the Wi-Fi discovery of the enhanced algorithm finishes more quickly than that of baseline algorithm in all environments under various channel conditions. This

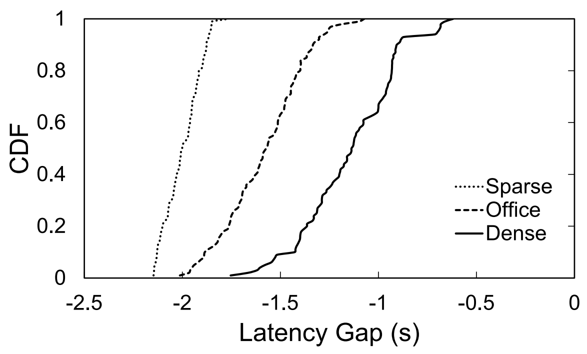


Fig. 19. Comparison of detection latency at different levels of channel occupancy in real environments. The latency gap is the difference in detection times between baseline and enhanced algorithms (gap = time(enhanced) - time(baseline)).

is because the enhanced algorithm operates RSSI sampling on multiple WPAN channels and determines the presence of Wi-Fi signal for multiple Wi-Fi channels. Specially, in sparse environment, the enhanced algorithm is 2.3 times faster than the baseline algorithm (baseline: 3.53 s and enhanced: 1.54 s).

D. Energy Consumption

To evaluate the energy efficiency of *C-SCAN* compared with *legacy*, we performed experiments in multiple places, including sparse, office, and dense environments. We used a Monsoon power monitor and its PowerTool software [56] to measure the energy consumption during AP scanning of the Wi-Fi interface. It records the energy consumption every 200 ms. Fig. 15 shows the setup of our power measurement experiments. The battery of our test device is nonremovable and is integrated with a battery control chipset. To supply power with the power monitor, it needs a physical modification (detaching a battery control chipset from battery). After modification, the test device is connected to the power monitor through the chipset.

In the experiments, we restricted our device with the following settings to maintain reasonable experimental results. We set the backlight level of the phone screen to its lowest possible value and closed all unnecessary running apps and services. In addition, we also disabled all radio and other interfaces (e.g., NFC, GPS, and Sync) except for Wi-Fi.

We first measured the base energy consumption for 300 s, when the device contains its newly implemented application with the aforementioned restrictions while the Wi-Fi interface is turned off. The average and standard deviation of power consumption are 8155.57 and 0.21 uAh, respectively. We conducted the measurement for each scan method (*C-SCAN* and *legacy*) in various settings for equal duration. The average energy consumption of each scan method (active Wi-Fi scanning) can be approximated by subtracting the average base consumption from the energy consumption.

Fig. 20 shows the average energy consumption for the Wi-Fi interface. We focused on the active Wi-Fi scanning. Legacy consumes the highest energy in all settings since it scans full channel sets regardless of the channel conditions. In comparison, the *C-SCAN* uses lower energy by dispensing with

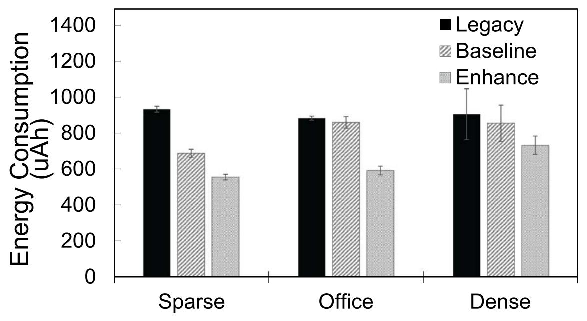


Fig. 20. Energy consumption.

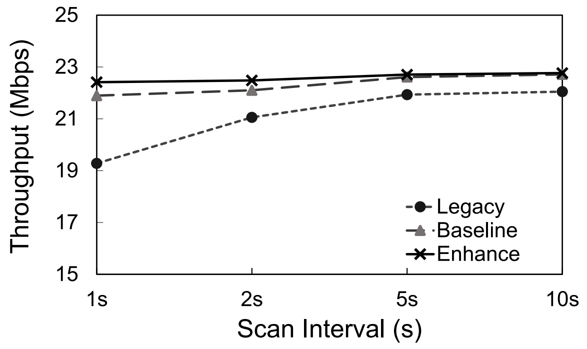


Fig. 21. Throughput comparison: *C-SCAN* versus legacy.

unnecessary scanning of AP-free Wi-Fi channels. Notably, the sparse environment presents a 26% (baseline) and 40% (enhanced) decrease in energy consumption for legacy despite of the FP cases. We further observed for the mentioned settings that the average consumption of the enhanced algorithm is more reduced than that of baseline due to low FPs.

E. Throughput

To identify any throughput improvement of *C-SCAN* in comparison with *legacy*, we conducted experiments using *iPerf3*, configured for TCP, in a downlink scenario. We used a TP-Link AP, which supports 802.11 b/g/n in 2.4 GHz, as a WLAN network. The testing device was the only node associated with the AP and its distance from the AP was roughly 1 m. A Samsung laptop, which generates downlink traffic for TCP, was directly connected to the AP through a Gigabit Ethernet. Since the throughput can be affected by signal interference, we designed a controlled environment where no signal interference exists (i.e., anechoic chamber). Each experiment was repeated 100 times with a measurement duration of 100 s.

Fig. 21 presents the throughput measurements of *C-SCAN* and legacy for different scan intervals. The latter is gradually increased from 1 to 10 s. As shown in the figure, both *C-SCAN* algorithms maintain high throughput even with aggressive scan interval. This indicates that *C-SCAN* has a minimal impact on the throughput since it prevents unnecessary probings. Notably, in the shortest scan interval, the performance improvement is the largest among the various scan intervals: 19.3 Mb/s for legacy, 21.9 Mb/s for baseline, and 22.4 Mb/s for the enhanced algorithm on average. However, the improvement decreases as the scan interval increases. Even though the improvement for longer intervals is not higher than that for the shortest interval, *C-SCAN* still achieves a performance gain in the longer scan intervals. When the scan interval is set to 10 s, the throughput improvements for legacy are about 2.9% (0.6 Mb/s) and 3.2% (0.7 Mb/s) for the baseline and enhanced algorithm, respectively.

VIII. Conclusion

In this paper, we presented *C-SCAN*, a novel and lightweight scheme for mobile devices, that utilizes coexisting low-power WPAN radios to identify available Wi-Fi channels. We implemented a prototype of *C-SCAN* using a Bluetooth-compliant wireless transceiver and demonstrated that *C-SCAN* achieves a high detection accuracy with only a short delay.

In the future, we would like to extend this paper to pinpoint the channel numbers of Wi-Fi signals and to estimate the link quality for a given target AP using WPAN radios. This can be exploited for the energy-efficient AP selection problem. We will investigate methods to process highly uncertain RSSI samples to operate even in high mobility environments.

References

[1] "Cisco visual networking index: Forecast and methodology, 2016–2021," San Jose, CA, USA, Cisco, White Paper, Jun. 2017.

[2] K.-H. Kim, A. W. Min, D. Gupta, P. Mohapatra, and J. P. Singh, "Improving energy efficiency of Wi-Fi sensing on smartphones," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2011, pp. 2930–2938.

[3] N. Montavont, A. Arcia-Moret, and G. Castignani, "On the selection of scanning parameters in IEEE 802.11 networks," in *Proc. IEEE Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, London, U.K., 2013, pp. 2137–2141.

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

[5] C.-F. Chiasserini and R. R. Rao, "Improving battery performance by using traffic shaping techniques," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 7, pp. 1385–1394, Jul. 2001.

[6] G. Ananthanarayanan and I. Stoica, "Blue-Fi: Enhancing Wi-Fi performance using Bluetooth signals," in *Proc. ACM Int. Conf. Mobile Syst. Appl. Services (MobiSys)*, 2009, pp. 249–262.

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

[8] K. Chebrolu and A. Dhekne, "Esense: Energy sensing-based cross-technology communication," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2303–2316, Nov. 2013.

[9] J. Choi, "WidthSense: Wi-Fi discovery via distance-based correlation analysis," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 422–425, Feb. 2017.

[10] K. Wu *et al.*, "CSI-based indoor localization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1300–1309, Jul. 2013.

[11] *Ubertooth-One*. Accessed: Jan. 10, 2017. [Online]. Available: http://ubertooth.sourceforge.net/

[12] *How Wi-Fi Drains Your Cell Phone*. Accessed: Jun. 24, 2010. [Online]. Available: http://www.technologyreview.com/

[13] G. Castignani, A. Arcia, and N. Montavont, "A study of the discovery process in 802.11 networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 15, no. 1, pp. 25–36, 2011.

[14] A. Gupta and P. Mohapatra, "Energy consumption and conservation in WiFi based phones: A measurement-based study," in *Proc. IEEE Int. Conf. Sens. Commun. Netw. (SECON)*, San Diego, CA, USA, 2007, pp. 122–131.

[15] R. Raghavendra, E. M. Belding, K. Papagiannaki, and K. C. Almeroth, "Unwanted link layer traffic in large IEEE 802.11 wireless networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 9, pp. 1212–1225, Sep. 2010.

[16] A. Gupta, J. Min, and I. Rhee, "WiFox: Scaling WiFi performance for large audience environments," in *Proc. ACM Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, Nice, France, 2012, pp. 217–228.

[17] J. Yeo, M. Youssef, and A. Agrawala, "A framework for wireless LAN monitoring and its applications," in *Proc. ACM Workshop Wireless Security (WiSE)*, Philadelphia, PA, USA, 2004, pp. 70–79.

[18] S. Rayanchu, A. Patro, and S. Banerjee, "Catching whales and minnows using WiFiNet: Deconstructing non-WiFi interference using WiFi hardware," in *Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2012, p. 5.

[19] K. Doppler, C. B. Ribeiro, and J. Kneckt, "On efficient discovery of next generation local area networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2011, pp. 269–274.

[20] X. Hu, L. Song, D. Van Bruggen, and A. Striegel, "Is there WiFi yet?: How aggressive probe requests deteriorate energy and throughput," in *Proc. ACM Internet Meas. Conf. (IMC)*, Tokyo, Japan, 2015, pp. 317–323.

[21] W. Wang, M. Motani, and V. Srinivasan, "Opportunistic energy-efficient contact probing in delay-tolerant applications," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1592–1605, Oct. 2009.

[22] H. Velayos and G. Karlsson, "Techniques to reduce the IEEE 802.11b handoff time," in *Proc. IEEE Commun.*, Paris, France, 2004, pp. 3844–3848.

[23] H. Wu, K. Tan, Y. Zhang, and Q. Zhang, "Proactive scan: Fast handoff with smart triggers for 802.11 wireless LAN," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2007, pp. 749–757.

[24] C. Xu, J. Teng, and W. Jia, "Enabling faster and smoother handoffs in AP-dense 802.11 wireless networks," *Comput. Commun.*, vol. 33, no. 15, pp. 1795–1803, 2010.

[25] S. Shin, A. G. Forte, A. S. Rawat, and H. Schulzrinne, "Reducing MAC layer handoff latency in IEEE 802.11 wireless LANs," in *Proc. ACM Int. Symp. Mobility Manag. Wireless Access (MobiWac)*, Philadelphia, PA, USA, 2004, pp. 19–26.

[26] M. F. Tuysuz and H. A. Mantar, "Smart channel scanning with minimized communication interruptions over IEEE 802.11 WLANs," in *Proc. IEEE Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, London, U.K., 2013, pp. 2218–2222.

[27] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: Vehicular content delivery using WiFi," in *Proc. ACM Int. Conf. Mobile Comput. Netw. (MobiCom)*, San Francisco, CA, USA, 2008, pp. 199–210.

[28] S. Chakraborty, Y. Dong, D. K. Y. Yau, and J. C. S. Lui, "On the effectiveness of movement prediction to reduce energy consumption in wireless communication," *IEEE Trans. Mobile Comput.*, vol. 5, no. 2, pp. 157–169, Feb. 2006.

[29] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das, "Predictive methods for improved vehicular WiFi access," in *Proc. ACM Int. Conf. Mobile Syst. Appl. Services (MobiSys)*, Kraków, Poland, 2009, pp. 263–276.

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

[31] N. Poosamani and I. Rhee, "Wi-Fi hotspot auto-discovery: A practical & energy-aware system for smart objects using cellular signals," in *Proc. 12th EAI Int. Conf. Mobile Ubiquitous Syst. (MobiQuitous)*, Coimbra, Portugal, 2015, pp. 90–99.

[32] Y. Agarwal *et al.*, "Somniloquy: Augmenting network interfaces to reduce PC energy usage," in *Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Boston, MA, USA, 2009, pp. 365–380.

[33] A. Currid, "TCP offload to the rescue," *ACM Queue*, vol. 2, no. 3, pp. 58–65, 2004.

[34] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha, "FM-based indoor localization," in *Proc. ACM Int. Conf. Mobile Syst. Appl. Services (MobiSys)*, 2012, pp. 169–182.

[35] T. Li, C. An, R. Chandra, A. T. Campbell, and X. Zhou, "Low-power pervasive Wi-Fi connectivity using WiScan," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. (UbiComp)*, Osaka, Japan, 2015, pp. 409–420.

[36] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-efficiency of MIMO and cooperative MIMO techniques in sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 6, pp. 1089–1098, Aug. 2004.

[37] P. Liu, Z. Tao, Z. Lin, E. Erkip, and S. Panwar, "Cooperative wireless communications: A cross-layer approach," *IEEE Wireless Commun.*, vol. 13, no. 4, pp. 84–92, Aug. 2006.

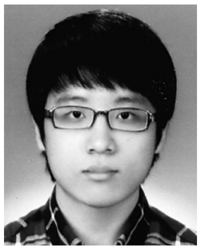
[38] J. Park, E. Song, and W. Sung, "Capacity analysis for distributed antenna systems using cooperative transmission schemes in fading channels," *IEEE Trans. Wireless Commun.*, vol. 8, no. 2, pp. 586–592, Feb. 2009.

[39] H. Wang, X.-G. Xia, and Q. Yin, "Distributed space-frequency codes for cooperative communication systems with multiple carrier frequency offsets," *IEEE Trans. Wireless Commun.*, vol. 8, no. 2, pp. 1045–1055, Feb. 2009.

[40] B. Sirkeci-Mergen and A. Scaglione, "Randomized space-time coding for distributed cooperative communication," *IEEE Trans. Signal Process.*, vol. 55, no. 10, pp. 5003–5017, Oct. 2007.
[41] X. Li, T. Jiang, S. Cui, J. An, and Q. Zhang, "Cooperative communications based on rateless network coding in distributed MIMO systems," *IEEE Wireless Commun.*, vol. 17, no. 3, pp. 60–67, Jun. 2010.
[42] Q. Zhang, J. Jia, and J. Zhang, "Cooperative relay to improve diversity in cognitive radio networks," *IEEE Commun. Mag.*, vol. 47, no. 2, pp. 111–117, Feb. 2009.
[43] J. Jia, J. Zhang, and Q. Zhang, "Cooperative relay for cognitive radio networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2009, pp. 2304–2312.
[44] L. Li *et al.*, "Simplified relay selection and power allocation in cooperative cognitive radio systems," *IEEE Trans. Wireless Commun.*, vol. 10, no. 1, pp. 33–36, Jan. 2011.
[45] Y. Zou, Y. Yao, and B. Zheng, "Cooperative relay techniques for cognitive radio systems: Spectrum sensing and secondary user transmissions," *IEEE Commun. Mag.*, vol. 50, no. 4, pp. 98–103, Apr. 2012.
[46] N. Mishra, K. Chebrolu, B. Raman, and A. Pathak, "Wake-on-WLAN," in *Proc. ACM Int. Conf. World Wide Web (WWW)*, Edinburgh, U.K., 2006, pp. 761–769.
[47] N. Mishra, D. Golcha, A. Bhadauria, B. Raman, and K. Chebrolu, "S-WOW: Signature based wake-on-WLAN," in *Proc. IEEE Int. Conf. Commun. Syst. Softw. Middleware (COMSWARE)*, Bengaluru, India, 2007, pp. 1–8.
[48] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins, "Turducken: Hierarchical power management for mobile devices," in *Proc. ACM Int. Conf. Mobile Syst. Appl. Services (MobiSys)*, Seattle, WA, USA, 2005, pp. 261–274.
[49] X. Zhang and K. G. Shin, "E-MiLi: Energy-minimizing idle listening in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 9, pp. 1441–1454, Sep. 2012.
[50] W. Wang, Y. Chen, L. Wang, and Q. Zhang, "Sampleless Wi-Fi: Bringing low power to Wi-Fi communications," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1663–1672, Jun. 2017.
[51] Texas Instruments. *CC2420 IEEE 802.15.4 Compliant and ZigBee Ready RF Transceiver*. Accessed: May 1, 2016. [Online]. Available: http://www.ti.com/product/CC2420
[52] *Android WiFiManager*. Accessed: May 1, 2016. [Online]. Available: https://developer.android.com/reference/android/net/wifi/WifiManager.html
[53] *List of 2.4 GHz Radio Use*. Accessed: May 1, 2016. [Online]. Available: https://en.wikipedia.org/wiki/List_of_2.4_GHz_radio_use
[54] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.
[55] L. Al Shalabi, Z. Shaaban, and B. Kasasbeh, "Data mining: A preprocessing engine," *J. Comput. Sci.*, vol. 2, no. 9, pp. 735–739, 2006.
[56] *Monsoon Power Monitor*. Accessed: Mar. 1, 2017. [Online]. Available: https://www.msoon.com/LabEquipment/PowerMonitor/

Jonghwan Chung received the B.S. degree in computer science from Soongsil University, Seoul, South Korea, in 2015, and the M.S. degree in computer science and engineering from Seoul National University, Seoul, in 2017.

He is currently a Researcher of system kernels with Tmax OS, Seongnam, South Korea. His current research interests include wireless networking, with emphasis on multipath transport protocol, Internet measurement, and Internet of Things.

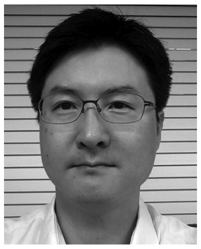


Junhyun Park (M'15) received the B.S. degree in computer science from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2013. He is currently pursuing the Ph.D. degree at the School of Computer Science and Engineering, Seoul National University, Seoul, South Korea.

His current research interests include wireless networks, specifically IEEE 802.11 WLAN and Internet of Things, and social networks security.

Chong-Kwon Kim received the B.S. degree in industrial engineering from Seoul National University, Seoul, South Korea, in 1981, the M.S. degree in operations research from the Georgia Institute of Technology, Atlanta, GA, USA, in 1982, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 1987.

In 1987, he joined Bellcore, as a Technical Staff Member, where he was involved with broadband ISDN and ATM. Since 1991, he has been with Seoul National University, as a Professor with the School of Computer Science and Engineering. His current research interests include wireless and mobile networking, social network analysis, SNS security, and performance evaluation.



Jaehyuk Choi (GS'07–M'11) received the Ph.D. degree in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2008.

He is currently an Associate Professor with the Department of Software, Gachon University, Seongnam, South Korea. He was with the Real-Time Computing Laboratory, University of Michigan, Ann Arbor, MI, USA, as a Post-Doctoral Researcher from 2008 to 2011. His current research interests include wireless/mobile systems, Internet of Things connectivity and sensing systems, with an emphasis on wireless LAN/PAN, edge computing, multipath TCP, network management, and ubiquitous passive sensing.