# Accelerating BLE Neighbor Discovery via Wi-Fi Fingerprints

Tong Li[†‡], Bowen Hu[†], Guanjie Tu[†], Jinwen Shuai[†], Jiaxin Liang[‡], Yukuan Ding[§], Ziwei Li[¶], and Ke Xu[¶]
Renmin University of China[†], Huawei[‡], HKUST[§], Tsinghua University[¶]

***Abstract*—In this paper, we demonstrate the design of FiND, a novel neighbor discovery protocol that accelerates BLE neighbor discovery via Wi-Fi fingerprints without any hardware modifications. The design rationale of FiND is that the two modes of Wi-Fi and BLE show complementarity in both wireless interference and discovery pattern. When abstracting the neighbor discovery problem, this demonstration provides validation for the approach of reasoning-based presence detection in the real world.**

## I. Introduction

Bluetooth Low-Energy (BLE) neighbor discovery [1] acts as the prerequisite stage of the widespread Proximity Beacon scenarios such as marketing activities (e.g., advertising, promotions, and scheduling) and interactive applications (e.g., seamless access systems, robot navigation) [2]. BLE neighbor discovery suffers from the trade-off between latency and power consumption. As illustrated in Figure 1, the neighbor discovery latency is mainly bounded by the broadcaster's broadcast interval ($A$), the scanner's scan window ($W$), and scan interval ($T$), where the scan duty cycle is computed by $D = \frac{W}{T}$. In general, power consumption is proportional to $D$ and is inversely proportional to $A$. Since most devices are power-sensitive, a large $A$ (e.g., $>$ 1000 ms) and a low $D$ (e.g., $< 10\%$) are usually applied in modern neighbor discovery applications [3]. However, a larger $A$ or a lower $D$ results in an interleaved activity pattern between the broadcaster and the scanner, which may induce unacceptably large discovery latency (e.g., $> 5$ seconds) [1].

BLE neighbor discovery further suffers from wireless interference. Today, BLE Beacons (iBeacon, Eddystone, AltBeacon, aBeacon, HiBeacon, etc.) employing the 2.4 GHz ISM frequency band are widely deployed in public places. Moreover, with the advent of the Internet of Things, there is a sharp increase in Bluetooth-equipped devices, especially in wearable devices. It is anticipated that the wireless interference will result from all these devices operating in the same environment, this mutual interference leads to performance degradation of BLE neighbor discovery. For example, more than 50 Bluetooth signals may exist simultaneously in a medium-sized mall. In this case, even with a small $A$ and a large $D$, the latency of neighbor discovery may be far from satisfactory due to signal collisions (see §III).

This work is supported by the NSFC Projects (No. 62202473 and No. 61932016), the China National Funds for Distinguished Young Scientists (No. 61825204), and the Beijing Outstanding Young Scientist Program (No. BJJWZYJH01201910003011). Tong Li's work is partially done at Huawei.

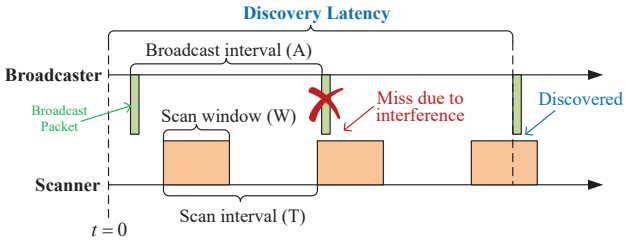
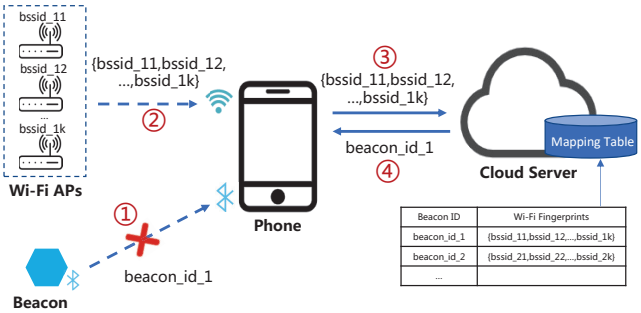



**Fig. 1: The duty-cycling of BLE neighbor discovery.**

Kindt et al. [1] have proposed the tight bounds on latency that no prior neighbor discovery parameter setting approaches can beat, and concluded there is no further potential to improve the relationship between latency and duty cycle. However, we argue that a huge room still exists for improvement with the auxiliary of ubiquitous Wi-Fi APs in practical scenarios. **First**, Wi-Fi and BLE show complementarity in wireless interference. Although both BLE and Wi-Fi operate in the 2.4GHz ISM band, the 3 channels (i.e., channels 37 (2402MHz), 38 (2426MHz), and 39 (2480MHz)) used by BLE neighbor discovery are almost unaffected by Wi-Fi interference (e.g., channels 1-11 (2412-2472MHz)). **Second**, Wi-Fi and BLE show complementarity in discovery patterns in two aspects: (a) Wi-Fi not only supports BLE-like passive scanning (see Figure 1) but also supports active scanning during which the client radio transmits a probe request and listens for a probe response from an AP. Generally, a passive scan takes more time in neighbor discovery, since the client must listen and wait for a beacon versus actively probing to find an AP. (b) Wi-Fi always returns discovery results (although might be from a previous scan if the current scan has not been completed or succeeded) [4], while BLE may return nothing.

When abstracting the neighbor discovery problem as the presence detection of the BLE Beacon in a certain space, considering the ubiquity of Wi-Fi APs, we have seen the possibilities of reasoning about the presence of BLE Beacon from the presence of Wi-Fi fingerprints, which is denoted by a list of Wi-Fi APs nearby the BLE Beacon. Thus we design FiND, a Fingerprint-based Neighbor Discovery protocol that makes full use of the complementarity between Wi-Fi and BLE. In the case of a long BLE neighbor discovery latency, FiND accelerates the discovery by deducing the presence of the BLE Beacons according to the presence of Wi-Fi fingerprints through the historical correlation between them.

We implement the proposed protocol FiND into the Android platform with the assistance of a cloud server (see




① Step 1: Phone fails to find the Beacon via BLE-based neighbor discovery
② Step 2: Phone collects the Wi-Fi fingerprint in its own surroundings
③ Step 3: Phone sends a request with the Wi-Fi fingerprint to Cloud Server
④ Step 4: Cloud Server sends a response with the corresponding Beacon ID

**Fig. 2: The basic workflow of FiND.**

https://github.com/litonglab/find). We further showcase two representative scenarios of different duty cycles and wireless interference: (1) Employing FiND to accelerate BLE neighbor discovery with a limited power budget (i.e., a low-duty cycle), and (2) employing FiND to accelerate BLE neighbor discovery with fierce interference. For both scenarios, FiND significantly reduces the latency of neighbor discovery.

## II. Design

### A. The Basic Workflow of FiND

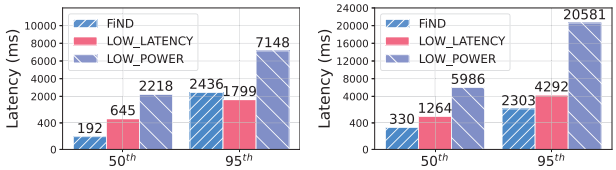
Figure 2 illustrates an example of the basic workflow of FiND where a Phone uses a Wi-Fi fingerprint to deduce the Beacon ID with the help of a remote Cloud Server. **Step 1**: The Phone starts BLE scanning for a certain period of time, if it fails to find the Beacon, it goes to Step 2. **Step 2**: The Phone collects the Wi-Fi fingerprint via Wi-Fi scanning or directly reads the historical records readily available in the cache. **Step 3**: The Phone sends a request with the Wi-Fi fingerprint to the Cloud Server. **Step 4**: Cloud Server then queries the Beacon ID according to the Wi-Fi fingerprint in the Mapping Table, and sends a response to the phone. The Phone then *indirectly* discovers the BLE Beacon by applying the FiND system.

### B. How to Build the Mapping Table?

As shown in Figure 2, a Mapping Table contains the mapping between the Beacon IDs (e.g., beacon_id_1, beacon_id_2) and Wi-Fi Fingerprints (e.g., {bssid_11, bssid_12, ..., bssid_1k}, {bssid_21, bssid_22, ..., bssid_2k}). In the case that the Mapping Table does not contain the historical mapping between a specific Wi-Fi Fingerprint and a Beacon ID, a.k.a, cold start, FiND does not provide acceleration but just initializes a record into the Mapping Table for future use. The Mapping Table can also be updated by other users (i.e., other Phones) in the case when Step 1 succeeds (i.e., the Phone gets the Beacon ID directly via BLE scanning).

### C. How to Match the Fingerprints

The instability of wireless signals leads to the dynamic nature of Wi-Fi Fingerprints. Instead of the exact matching of a list of Wi-Fi APs, we conduct fuzzy matching to improve the robustness of FiND. Our demonstration simply adopts the Jaccard index [5] to calculate the similarity coefficient between fingerprints: $J(\mathbf{A},\mathbf{B}) = \frac{|\mathbf{A}\cap\mathbf{B}|}{|\mathbf{A}\cup\mathbf{B}|}$. $\mathbf{A}$ and $\mathbf{B}$ are the sets of current and historical fingerprints, respectively. $|\cdot|$ denotes the size of a set. In this paper, we define the fingerprints are matched if $J(\mathbf{A},\mathbf{B}) \geq \alpha$ (e.g., $\alpha = 0.5$).




(a) Little interference (b) Fierce interference

**Fig. 3: Testbed results.**

## III. Demonstration and Evaluation

To verify the feasibility of FiND, we set up a testbed by running FiND on an Android phone (e.g., Huawei P40) that connects to a remote cloud server (e.g., Amazon EC2) with an average round-trip time (RTT) of 100 ms. The Beacon's broadcast interval is 1000 ms, and the phone considers two BLE scan modes labeled as LOW_POWER ($D = 10\%$) and LOW_LATENCY ($D = 100\%$) in Android system [6]. When running FiND, the phone works in the LOW_POWER scan mode. Two representative scenarios are demonstrated.

**Scenario 1: Little interference.** We first deploy the BLE Beacons and a phone in an open spot with little (no) Bluetooth interference, where over 10 Wi-Fi APs can be discovered. Figure 3(a) demonstrates that employing FiND significantly accelerates BLE neighbor discovery with a low-duty cycle. For example, when the phone runs in LOW_POWER mode, FiND reduces 91.3% and 65.9% of the $50^{th}$ and $95^{th}$ percentile latency, respectively.

**Scenario 2: Fierce interference.** We then deploy the BLE Beacons and the phone in an office with fierce interference (20+ alive BLE signals are randomly deployed nearby), where over 10 Wi-Fi APs can be discovered. Figures 3(a) and 3(b) demonstrate that FiND achieves stable and low-latency neighbor discovery while the legacy ways inevitably get worse in the case of fierce interference. For example, even compared with the way of using LOW_LATENCY mode, FiND still reduces 73.9% and 46.3% of the $50^{th}$ and $95^{th}$ percentile latency, respectively.

## IV. Conclusion

Using the cloud-assisted deployment with the complementarity between BLE and Wi-Fi signals, testbed results verify that the proposed FiND can achieve stable and low discovery latency regardless of the power budget or wireless interference.

## References

[1] P. H. Kindt and S. Chakraborty, "On optimal neighbor discovery," in *ACM SIGCOMM*, 2019, pp. 441–457.
[2] "Proximity beacon," https://altbeacon.org/, 2022.
[3] T. Li, J. Liang, D. Wang, Y. Ding, K. Zheng, X. Zhang, and K. Xu, "On design and performance of offline finding network," in *IEEE INFOCOM*, 2023, pp. 1–10.
[4] "Android wi-fi scan," https://developer.android.com/guide/topics/connectivity/wifi-scan, 2020.
[5] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912.
[6] "Android ble scan settings apis," https://developer.android.com/reference/android/bluetooth/le/ScanSettings.