

Theoretical and Experimental Analysis of WiFi Location Fingerprint Sampling Period

Qin Wu, Hao Lin, and Jiuzhen Liang^(✉)

Department of Computer Science, Jiangnan University,
Jiangsu, Wuxi 214122, China
jzliang@jiangnan.edu.cn

Abstract. Indoor positioning with smartphones is of great importance for a lot of applications and has attracted many researchers' interests these years. Received Signal Strength (RSS) fingerprinting has been considered as an efficient method for indoor positioning. Numerous systems have been developed based on it. Location fingerprint sampling is the first step of the RSS fingerprinting method. Slow sampling speed will delay the positioning speed and will reduce the accuracy if the tracking object is moving. Theoretically, the sampling period is about one fingerprint per second. However, our experiments on some Android phones/pads show that it may even take more than 10 s to sample a fingerprint occasionally. By analyzing the Android WiFi scanning framework, it is easy to find which part of the fingerprint sampling process costs more time. After theoretically analysis and experimental measurement, we provide some suggestions on how to improve sampling speed on some practical WiFi positioning system architectures. To contribute to the research community of WiFi positioning, we make all our measurement codes and our data sets available as open source.

Keywords: Indoor positioning · WiFi · Location fingerprint · Sampling period · Android

1 Introduction

Indoor positioning with smartphones is critical for many applications. In many environment (e.g. airport terminals, conferences, shopping malls), indoor positioning of mobile phones helps users to realize indoor navigation or find friends. Hospitals can use it to take care of patients. Businesses need it to analyze customers' group behaviour [1].

There are many methods for indoor positioning. And these methods can be roughly classified into three categories: ranging based TOA/TDOA (Time of Arrival/Time Differential of Arrival) [2], Radio Frequency (RF) Fingerprinting [3,4], and Inertial Sensor based dead-reckoning [5]. WiFi Received Signal Strength (RSS) Fingerprinting belongs to RF Fingerprinting. It is one of the most outstanding indoor positioning methods. It depends on the WiFi RSS of smartphones. Current work on WiFi RSS Fingerprinting mainly focuses on design

and implementation of WiFi based positioning systems [4,6], cutting down the workload of WiFi site surveys [7], and improving the robustness of indoor positioning [8,9].

The procedure of RSS Fingerprinting can be divided into two stages: offline stage and online stage. During the *offline* stage, a site survey is performed in the location area. The collected location coordinates/labels and their correlation RSS from nearby Access Points (APs) are saved into a database (also called Radiomap). During the *online* stage, a location positioning technique uses the currently observed RSS and previously collected information to calculate the estimated location.

The **fingerprint sampling** is the first step for both offline stage and online stage. It is often overlooked by many researchers. In this paper, the WiFi RSS Fingerprint sampling speed procedure is analyzed theoretically based on IEEE 802.11 scanning scheme. And we did many experiments to measure the sampling speed with different android phones/pads. Based on the theoretically analysis and experimental measurements, fingerprint sampling selection strategies are proposed for different indoor positioning system architectures. In order to contribute to the WiFi positioning research community, we make all our data and codes available as open source.

The rest of the paper is organized as follows. Section 2 introduces the IEEE 802.11 channel scanning mechanism, which is the theoretical foundation of WiFi fingerprint sampling. In Sect. 3, Android WiFi scanning framework is introduced and some experimental observations are presented. Some WiFi fingerprint sampling strategies on different WiFi positioning system architectures are proposed in Sect. 4. And conclusions are made in Sect. 5.

2 IEEE 802.11 Channel Scanning

Two scanning methods are defined in the IEEE 802.11 [10]: *passive scanning* and *active scanning*, which are depend on the *ScanType* parameter of the MLME-SCAN.request primitive. In the following, we discuss these two methods in details and calculate the sampling period via theoretical analysis.

2.1 Passive Scanning

In the passive scanning (as shown in Fig. 1(a)), the Station(STA) switches to a candidate channel and waits for a periodical Beacon from any AP on that channel to announce its presence. Since the time span of beacons generated by different AP is independent, the STA must therefore wait for a full period on each channel. Thus,

$$t_{passive} = N_{channel} \cdot T_{beacon} \quad (1)$$

where $t_{passive}$ is the total time for a round of passive scanning, $N_{channel}$ is the number of channels used by the passive scanning, $N_{channel}$ is less than or equal to the total number of channels used in a country. For example, there are 13 channels available for usage in the 802.11 2.4 GHz WiFi frequency range in China.

With all the 13 channels to be scanned and a beacon period of 102.4 ms, it will take roughly 1.33 s (we do not consider the channel switch time at this moment) for a sampling period. The passive scanning scheme assumes that the target is at a fixed location during the sampling period. However, in real application, the target is usually moving. Suppose a person walks at a speed 2 m/s and starts from the origin, then the distance that person walked is $2\text{ m/s} \times 1.33\text{ s} = 2.66\text{ m}$ after one sampling period. One may find that the sampling result is actually a blur line in the interval $[0\text{ m}, 2.66\text{ m}]$ of the walking distance. We call it motion blur. On the other hand, if fingerprints from several sampling periods are collected and their average is used to determine the location of an AP, then slow fingerprint sampling speed will influence the result badly.

2.2 Active Scanning

In the active scanning (Fig. 1(b)), the STA will do the following procedure, when it receives a MLME-SCAN.request primitive whose *ScanType* parameter is set as ACTIVE.

1. Wait until the *ProbeDelay* time has expired or a PHYRxStart.indication primitive has been received.
2. Contenting the media to send out the *ProbeRequest* frame.
3. Starting a *ProbeTimer*.
4. If the STA didn't receive any frame before the *ProbeTimer* reaches *MinChannelTime*, then the STA scans the next channel. Otherwise, the STA has to wait until the *ProbeTimer* reaches *MaxChannelTime*, then scans the next channel.

To determine the active scanning period, there are five parameters: *ProbeDelay*, *media contention time*, *MinChannelTime*, *MaxChannelTime*, and *channel switching time*. The default value of *ProbeDelay* is negligible ($<1\text{ ms}$) [11, 12]. And on lightly loaded networks, the value of *media contention time* is also negligibly small. The settings of *MinChannelTime* and *MaxChannelTime* have attracted many research's attentions [13, 14]. The *MinChannelTime* is recommended to be set as 6–7 ms, and *MaxChannelTime* is recommended to be set as 10–15 ms by [13]. The last parameter, *channel switching time*, is dependent of 802.11 cards.

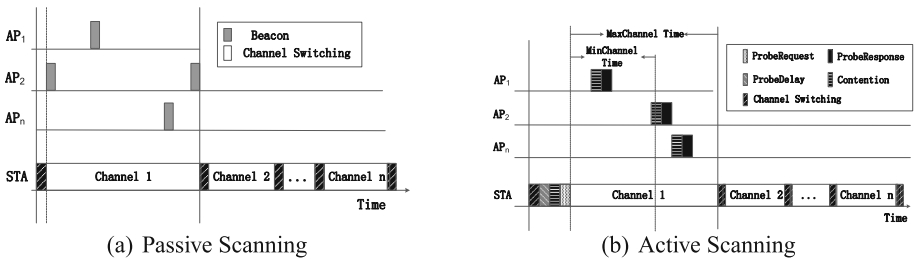


Fig. 1. Two types of scanning

For example, it is around 2.9 ms for cards with Intel chipsets, 4.8 ms for cards with Atheros chipsets [15] and 19 ms for cards with Intersil Prism2 chipsets [11]. Thus, the idealized bound of active scanning is calculated as

$$N_{channel} \cdot T_{min} \leq t_{active} \leq N_{channel} \cdot T_{max} \quad (2)$$

where $N_{channel}$ is the number of channels used to do active scanning, T_{min} is MinChannelTime, T_{max} is MaxChannelTime, t_{active} is the sampling period for active scanning. If taking the hardware dependent channel switching time into account, we should plus $(N_{channel} - 1) \cdot T_{switch}$, where T_{switch} is the channel switching time. Since the smallest channel switching time of the above three chipsets is 2.9 ms, MinChannelTime is 6 ms and there are 13 channels, then the total time of active scanning for each round is $13 \times 6 + 12 \times 2.9 = 112.8$ ms. If the MaxChannelTime is set to be 15 ms and the Intersil Prisms chipsets (whose channel switching time is 19 ms) is used, then the upper bound of the active scanning is $13 \times 15 + 12 \times 19 = 423$ ms.

A measurement conducted by [11] shows that, active scanning fired by Intersil Prism2-based 802.11b NICs (Network Interface Cards) takes around 350 to 400 ms (based on a firmware initiated scan), and Atheros 5212-based NICs takes roughly 500 ms (via a Windows XP driver-controlled scan).

3 WiFi Scanning on Android

3.1 Scanning Speed Measurements on Android

Last section shows the theoretically analysis of WiFi scanning and some measurement results on computers. But there is little work discussing the experimental measurement results based on popular Android phones/pads. So we conducted WiFi scanning experiments on 8 different android phones/pads. Figure 2 shows the phones/pads used in the experiments. The models and corresponding android OS (Operating System) versions of the phones/pads (marked as ‘A’ - ‘H’ in Fig. 2) are listed in Table 1.



Fig. 2. Android WiFi scanning experiment environment

Table 1. Phone/Pads’ models and OS versions

Phone	Model	Android version
A	SAMSUNG GT-P6800	4.0.4
B	HUAWEI C8650+	2.3.6
C	HUAWEI C8812	4.0.3
D	HTC ONE X	4.0.4
E	SAMSUNG GT-N7108	4.1.1
F	COOLPAD 5860	2.3.5
G	MEIZU M9	2.3.5
H	GOOGLE NEXUS 7	4.2.1

The easiest and most stable way to let the Android phones/pads to do scanning is using Android SDK API. Firstly, the `WifiManager.startScan()` is called to fire a `SCAN` event, and the `BroadcastReceiver.onReceive()` is overridden to receive the `SCAN_RESULTS_AVAILABLE_EVENT`. When the `SCAN_RESULTS_AVAILABLE_EVENT` is received, it means a round of scanning is done (this also means that we have sampled one fingerprint). Then the scan result is saved in a file. Right after it, another `WifiManager.startScan()` is fired and the procedure is repeated. The procedure is shown in Fig. 3.

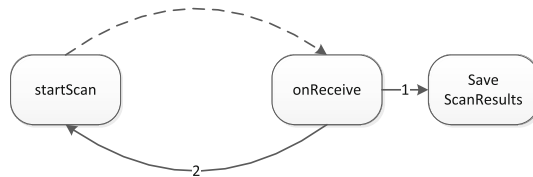


Fig. 3. SDK API scan procedure

We set AP generating beacons at channel 6, and used the 8 phones/pads to run the scan procedure about five minutes. Then we calculated the fingerprint sampling period by the difference the starting time of two successive `WifiManager.startScan()`. The results, which are shown in Fig. 4, are amazing. From Fig. 4, one may find that Phone A, C, F, G took over 10 s to sample a fingerprint occasionally. For Phone F, we sampled 103 times, and there were 12 times that the sampling period of Phone F was greater than 10 s, so the percentage of sampling period over 10 s is $12/103 \times 100\% = 11.7\%$. We also find that phone G is interesting, its fingerprint sampling time alternates between 0 s and 0.6 s. We guess it is because that, when the phone calls `WifiManager.startScan()`, the under layer started scan and immediately returned a `SCAN_RESULTS_AVAILABLE_EVENT`, which should not occur logically. And after hardware finished the scan, it reported another

SCAN_RESULTS_AVAILABLE_EVENT, which eventually caused the SCAN_RESULTS_AVAILABLE_EVENT to be reported twice. Phone B also comes to our attention as it used less than 0s to sampling a fingerprint, we guess it was caused by the background scanning and message queue mechanism. The scanning time on Phone E and Phone H is stable, but 1.59 s and 0.87 s is a little larger than the sampling period by theoretical analysis, which is around 400 ms to 500 ms (we also used a PC to monitor the phones/pads' Probe Request frames to confirm their active scanning periods in another separate experiment).

Is the difference caused by the Android OS? Can we go over the WiFi scanning framework on the android platform to improve the scanning speed? Is the program still stable in that case? In the next subsection, we dig into details of the Android based WiFi scanning framework, and find the keys to these questions.

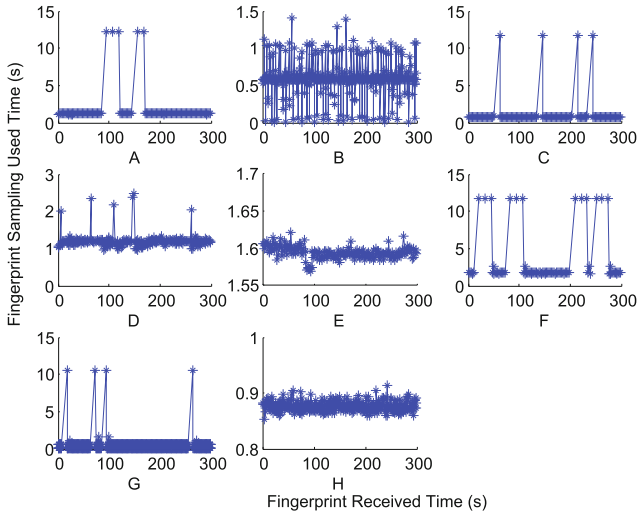


Fig. 4. Scan time by using SDK API

3.2 WiFi Scanning Workflow on Android Platform

The Android WiFi Scanning Framework can be divided into the following parts: SDK API, Android Framework, JNI Layer, HAL Layer, wpa_supplicant and network device driver. Figure 5 shows the more detailed partitions graphically. WifiMonitor.MonitorThread.run, eloop_run and inter layer socket communication represent time delays. In order to calculate these time delays, the most efficient way is to record the time right before and right after them. As presented in previous section, we already recorded SDK API time, here we also recorded the scan starting time at *wpa_supplicant_ctrl_iface_process()*, where a branch matches the “SCAN” event, we recorded the time in *if (os_strerror(buf, “SCAN”) == 0){...}*. And we also record the receive time below the *case NL80211_CMD_NEW_SCAN_RESULTS: ...* in *do_process_drv_event()* method.

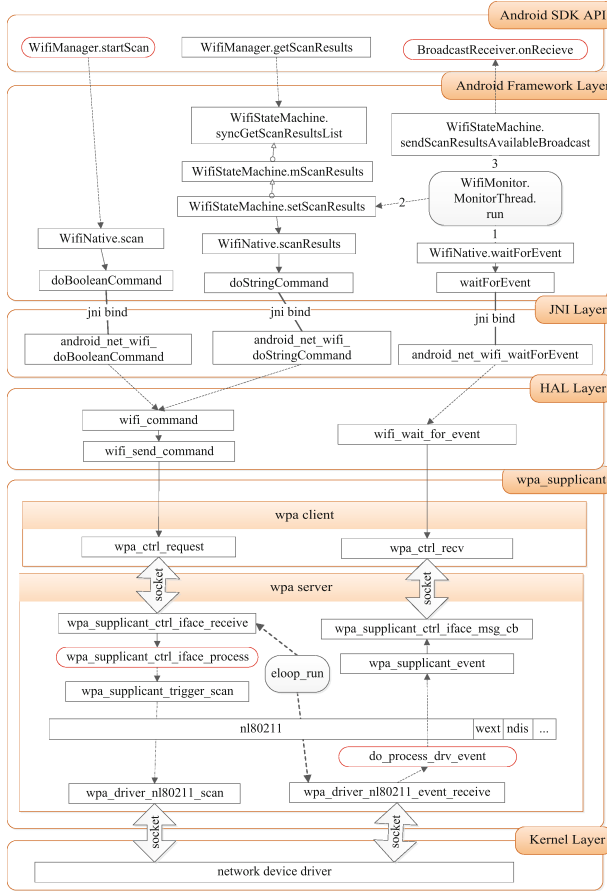


Fig. 5. Android WiFi scanning architecture

We conducted this experiment on Phone H, and the results are shown in Fig. 6. In the upper right corner of Fig. 6, one may find that the time used by all layers above the wpa_supplicant server is about 25 ms. We also calculated the average fingerprint sampling time, which is 841.7 ms.

Our experimental results show that it is hard to make the android OS scan faster even use the bottom layer. All the previous discussion is based on the general application scenarios, that is, STA samples the fingerprint and all channels have to be scanned. However, in some special scenarios, we can change WiFi location system frameworks and fingerprint sampling strategies to improve the sampling speed. We discuss fingerprint sampling strategies at different system frameworks in next section.

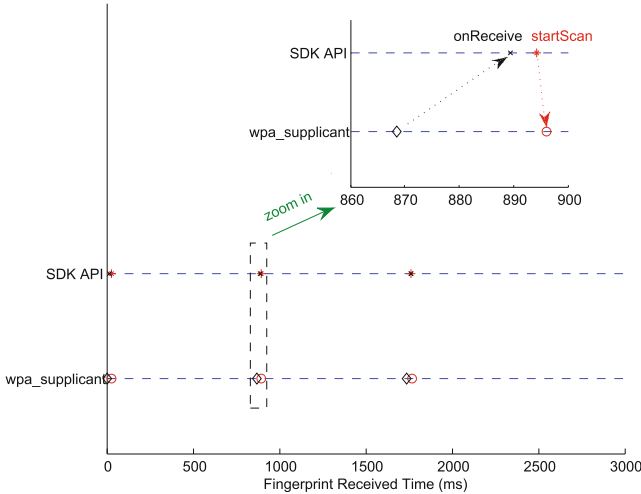


Fig. 6. Wpa_supplicant sampling speed

4 Fingerprint Sampling Strategies on Different System Architecture

Based on where the fingerprint location is calculated, positioning systems can be divided into four different system topologies according to [16] and [17]. The first one is the *remote positioning* system. Its signal transmitter is mobile (in our case, it can be looked as STA) and several fixed measuring units (they can be looked as APs in our case) receive the signal emitted by the transmitter. All the measurements are collected and used to compute the location of the transmitter at a central station. The second one is *self-positioning*. The measuring unit receives signals from several geographically distributed transmitters, and determine its position based on these measurements. The third one is *indirect remote positioning*, where the self-positioning measuring unit sends the position measurement to a remote side via a data link. Conversely, if the position measurement is transmitted from a remote positioning system to a mobile unit using a data link, it is called *indirect self-positioning*, which is the fourth system topology. All these four topologies are shown in Fig. 7.

Based on whether we have full control on APs, we add two classification criteria: AP channel setting permission and AP fingerprint sampling permission. AP channel setting permission is that we have the right to set the APs in the location service area work at specific channels. If APs can report the RSS sampling from STA (AP can measure the RSS of STA's ProbeRequest frame) and we have the right to collect these information to the Master Station, then we say that we have the AP fingerprint sampling permission. Actually, the remote positioning system and indirect self-positioning system implicitly need the AP fingerprint sampling permission. Generally, We should not assume that we have

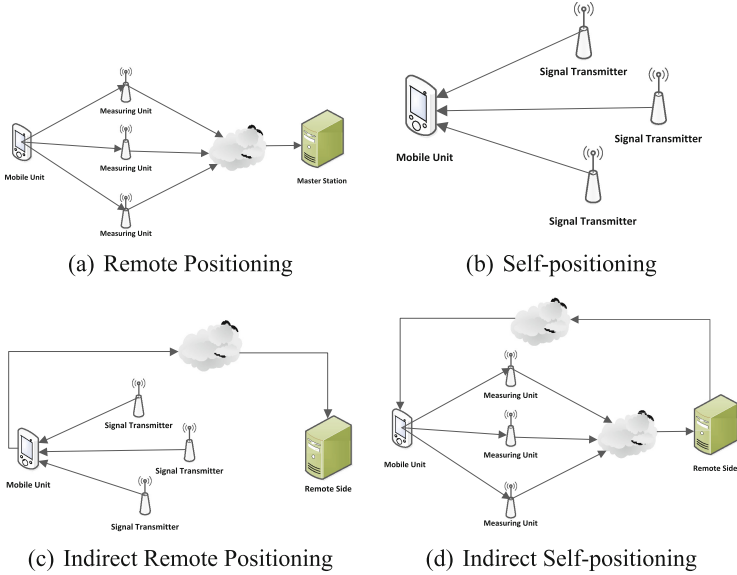


Fig. 7. Four different location system topologies

any permission to APs and we have to conduct the scanning procedure on every channel one after another.

If we have the permission to set the AP working channel, then we can set all APs in less than the max number of available channels defined by each country. For example, we can just use 3 non-overlap (actually they may still be a little overlap) channels: 1, 6, 11, to cover the location service area. In that circumstance, We can just scan 3 channels, and the scanning time will be less than $3 \cdot T_{max} + 2 \cdot T_{switch}$ in active scanning mode, and $3 \cdot T_{beacon} + 2 \cdot T_{switch}$ in passive scanning.

If we have the AP sampling permission, what will happen? The STA can dwell on each channel 0 second. Eventually, we need only $(N_{channel} - 1) \cdot T_{switch}$ seconds to scan all channels. If we also have the AP working channel set permission at the same time, we need only $2 \cdot T_{switch}$ seconds to scan 3 channels.

5 Conclusion

In this paper, we studied the WiFi fingerprint sampling problem. We theoretically analyzed the fingerprint sampling procedure, and calculated the fingerprint sampling speed. The Fingerprint sampling experiments were done on 8 phones/pads with Android SDK API. All our source codes and data are available as open source. We believe more measurements should be conducted on different phones/pads. In order to explore whether WiFi location fingerprint can be sampled more quickly, we analyzed the Android WiFi sampling framework and measured the fingerprint sampling speed at different layers. If we have

full control on APs, we showed that it is possible to improve the fingerprint sampling speed dramatically comparing to the general positioning systems. We believe that our systematical analysis and practical measurement results will contribute to the WiFi positioning research community and can provide some reference to practical WiFi positioning system design.

Acknowledgement. This work is partially supported by Blue Project of Universities in Jiangsu Province Training Young Academic Leaders Object, the six talent peaks project of Jiangsu Province (No. DZXX-028) and National Natural Science Foundation of China (No.61170121, 61202312).

References

1. Lam, K.Y., Ng, J.K., Wang, J.T.: A business model for personalized promotion systems on using WLAN localization and NFC techniques. In: 27th Advanced International Conference on Information Networking and Applications Workshops (WAINA), pp. 1129–1134 (2013)
2. Qi, Y., Soh, C.B., Gunawan, E., et al.: An accurate 3D UWB hyperbolic localization in indoor multipath environment using iterative taylor-series estimation. In: IEEE 77th Vehicular Technology Conference (VTC Spring), pp. 1–5 (2013)
3. Bahl, P., Padmanabhan, V.N.: RADAR: an in-building rf-based user location and tracking system. In: Proceedings of IEEE INFOCOM, pp. 775–784 (2000)
4. Youssef, M., Agrawala, A.K.: The horus wlan location determination system. In: Proceedings of ACM MobiSys, pp. 205–218 (2005)
5. Wang, H., Sen, S., Elgohary, A., et al.: No need to war-drive: unsupervised indoor localization. In: Proceedings of ACM MobiSys, pp. 197–210 (2012)
6. Laoudias, C., Constantinou, G., Constantinides, M., et al.: The airplace indoor positioning platform for android smartphones. In: IEEE 13th International Conference on Mobile Data Management (MDM), pp. 312–315 (2012)
7. Yang, Z., Wu, C., Liu, Y.: Locating in fingerprint space: wireless indoor localization with little human intervention. In: Proceedings of ACM MOBICOM, pp. 269–280 (2012)
8. Fang, S.-H., Lin, T.-N.: A dynamic system approach for radio location fingerprinting in wireless local area networks. *IEEE Trans. Commun.* **58**(4), 1020–1025 (2010)
9. Laoudias, C., Constantinou, G., Constantinides, M., Nicolaou, S., Zeinalipour-Yazti, D., Panayiotou, C.G.: An online sequential extreme learning machine approach to wifi based indoor positioning. In: IEEE World Forum on Internet of Things (2014)
10. IEEE Computer Society LAN/MAN Standards Committee. Ieee standard for information technology: Part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications (2012)
11. Ramani, I., Savage, S.: Syncscan: practical fast handoff for 802.11 infrastructure networks. In: Proceedings of IEEE INFOCOM, pp. 675–684 (2005)
12. Almulla, M., Wang, Y., Boukerche, A., et al.: A fast location-based handoff scheme for vehicular networks. In: IEEE International Conference on Communications (ICC), pp. 1464–1468 (2013)
13. Mishra, A., Shin, M., Arbaugh, W.A.: An empirical analysis of the ieee 802.11 MAC layer handoff process. *ACM SIGCOMM Comput. Commun. Rev.* **33**(2), 93–102 (2003)

14. Montavont, N., Arcia-Moret, A., Castignani, G.: On the selection of scanning parameters in IEEE 802.11 networks. In: IEEE 24th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), pp. 2137–2141 (2013)
15. Chen, X., Qiao, D.: Hand: Fast handoff with null dwell time for ieeeee 802.11 networks. In: Proceedings of IEEE INFOCOM, pp. 1–9 (2010)
16. Liu, H., Darabi, H., Banerjee, P.P., et al.: Survey of wireless indoor positioning techniques and systems. *IEEE Trans. Syst. Man Cybern.* **37**(6), 1067–1080 (2007)
17. Drane, C., Macnaughtan, M., Scott, C.: Positioning GSM telephones. *IEEE Commun. Mag.* **36**(4), 46–54 (1998)