WiFi Location Fingerprint Sampling Speed Measurement and Analysis

Hao Lin*, Jiuzhen Liang*, Junfei Li*

*School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China

Indoor localization on smartphones is of great importance for a range of applications and attracting many research interests these years. Received Signal Strength (RSS) fingerprinting has be considered as an efficient method in this area and numerous systems [1], [2] have been developed based on it. Fingerprint sampling is the first step of RSS fingerprinting method, slow sampling speed will delay the localization speed and will reduce the localization accuracy if location target is moving. Our measurements on 8 android phone/pads show that some phone/pads will even use above 10 s to sampling a fingerprint occasionally. Then we analysis the fingerprint sampling process by exploring the open source android system. In order to get more data sets and to contribute to the community, we made all our measurement codes and our data sets open source. After analysis and mesurement we also make some suggestions to how to improve the sampling speed on some practical location system architectures.

Index Terms—**Indoor Localization; WiFi Fingerprint Sampling; WiFi Scan; Android**

I. INTRODUCTION

Indoor localization on smartphones is critical on many applications. In many environment (e.g. airport terminals, conferences, shopping malls), the location helps users to access navigation, friend finding, promotion information; hospitals can use it to take care of patients; businesses need it to understand customers group behavior.

There exist many methods to do indoor localization and can be roughly classified as three categories: ranging based TOA/TDOA (Time of Arrival / Time Differential of Arrival) [3], Radio Frequency (RF) fingerprinting [4], [5] and Inertial Sensor based dead-reckoning [6]. WiFi RSS fingerprinting belongs to RF fingerprinting is one of the most outstanding indoor localization methods depends on the pervasive WiFi RSS on the smartphones. Current works on the WiFi RSS fingerprinting is focused on relieving the work load of site survey [7], making the location algorithm more robust [8], and systematically designing and implementing the location system [5].

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 as Radiomap). During the online stage, a location positioning technique uses the currently observed RSS and previously collected information to figure out an estimated location.

Fingerprint sampling is the first step of RSS fingerprinting procedure both on online and offline stages. But there are little works on analysis of it. In this paper, we will use IEEE 802.11 scanning scheme to analysis the WiFi RSS fingerprint sampling procedure theoretically and conduct many experiments to measure the sampling speed on android phone/pads practically.

Manuscript received July 21, 2013; revised Mon Day, Year.

Communication author: Jiuzhen Liang, born in 1968, male, professor. School of Internet of Things Engineering, Jiangnan University, Wuxi, 214122, China. Email: jz.liang@jiangnan.edu.cn.

In order to gain more experiment data to do analysis and to contribute to the community, we made all our measurement codes and data sets open source[1]. Based on the analyses and measurements, we also introduce the selection of fingerprint sampling strategies of different location system architectures.

The rest of the paper is organized as follows. Section II introduces the IEEE 802.11 channel scanning mechanism, which is the theoretical basis of WiFi fingerprint sampling. We show android WiFi scan architecture in Section III, and some experimental observation is also presented in this section. With the knowledge of the WiFi fingerprint sampling in theoretical and experimental, we analyze the selection of WiFi fingerprint sampling strategies on different WiFi location system architectures.

II. IEEE 802.11 CHANNEL SCANNING

There exist two types of IEEE 802.11 [9] channel scanning scheme, *passive scanning* and *active scanning*, depending on the *ScanType* parameter of the MLME-SCAN.request primitive.

A. Passive Scanning

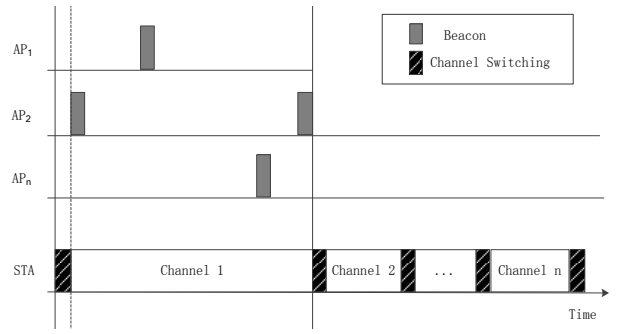
In the passive scanning (Figure 1(a)), the Station (STA) switches to a candidate channel and listens for periodic *Beacon* frame generated by APs to announce their presence (typically every 1 TU[2]). Since the time of beacons generated by different APs is independent, the STA must therefore wait the full interval on each channel. Thus:

$$t_{passive} = N_{channel} \cdot T_{beacon} \tag{1}$$

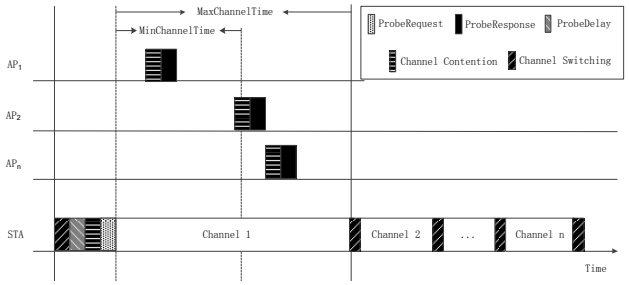
Where $t_{passive}$ is the total passive scanning time; $N_{channel}$ is the number of channels we used to do passive scanning, which is less than or equal to the total number of channels can be

[1]Our open source web site is https://github.com/imlinhao/wifiscan

[2]Time Unit (TU): A measurement of time equal to 1024 μs.



(a) Passive Scanning



(b) Active Scanning

Figure 1: Two Types of Scanning

used in a country, for example, in China there are 13 channels available for use in the 802.11 2.4GHz WiFi frequency range.

So with all the 13 channels to be scanned and a beacon interval of 102.4 ms, we will use roughly 1.33 s (we do not consider the channel switch time now). With a normal human walking speed 2 m/s, we will find that our sampling point is actually a blur line in 2.66 m area, which will reduce the localization accuracy. Some algorithms[8] using more fingerprints to smooth the RSS, will be more injured by this slow fingerprint sampling speed.

B. Active Scanning

In the active scanning (Figure 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.

There exist five parameters to determine the active scanning time. The default values of *ProbeDelay* is negligible (<1 ms) [10]. 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 [11], [12], [13], and the MinChannelTime is recommended to be set as 6-7 ms, and MaxChannelTime is recommended to be set as 10-15 ms by [11]. The last parameter *channel switching time* value is dependent of 802.11 card, for example it is around 2.9 ms for cards with Intel chipsets, 4.8 ms for cards with Atheros chipsets[14] and 19 ms for cards with Intersil Prism2 chipsets[10]. Thus the idealized bound of active scanning is calculated as

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

where $N_{channel}$ is the number of channels we used to do active scanning, T_{min} is MinChannelTime, T_{max} is MaxChannelTime, t_{active} is the time we used to do active scanning. If considering the hardware dependent channel switching time, we should plus $(N_{channel} - 1) \cdot T_{switch}$, where T_{switch} is the channel switching time. Using the smallest channel switching time of the above three chipsets, and the 6 ms MinChannelTime to active scanning the 13 channels, we can calculate the using time is $13 \times 6 + 12 \times 2.9 = 112.8ms$. We can also calculated out the up bound of the active scanning is $13 \times 15 + 12 \times 19 = 423ms$ by setting MaxChannelTime as 15 ms, and selecting the Intersil Prisms chipsets whose channel switch time is 19 ms. A measurement conducted by [10] shows that, active scanning fired by Intersil Prism2-based 802.11b NICs (Network Interface Cards) using between 350 and 400 ms (based on a firmware initiated scan) and Atheros 5212-based NICs using roughly 500 ms (via a Windows XP driver-controlled scan).

III. WiFi Scanning on Android

A. Measuring Scanning Speed on Android

Previous section shows the theoretically analysis of WiFi scanning and some measurement results on computer. But there are little works to show the similar results on popular used android phone/pads. So we conduct a WiFi scanning experiment on 8 android phone/pads (Figure 2 shows the experiment environment). The model and used android OS

Figure 2: Android WiFi Scanning Experiment Environment

(Operating System) version of the phone/pads marked as 'A' - 'H' is listed in Table 1. The easiest and most stable method to let the Android phone/pads to do scanning is using Android SDK API. We call the WifiManager.startScan() to fire a SCAN

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

Table 1: Phone/Pads' Model and OS Version

event, and override the BroadcastReceiver.onReceive() to receive the SCAN_RESULTS_AVAILABLE_EVENT. When SCAN_RESULTS_AVAILABLE_EVENT is received, it means a round of scanning is done (this also means we have sampled one fingerprint), we save the scan results in a file, and right after it we fire another WifiManager.startScan(). We present this procedure in Figure 3. We set AP generating beacons

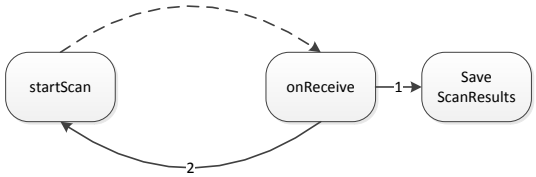


Figure 3: SDK API Scan Procedure

at channel 6, and use the 8 phones/pads to run the previous mentioned procedure little above five minutes. The results in Figure 4 are amazing (we calculate the fingerprint sampling used time by minus the time marked in the entry of onReceive and the time marked right before call place of startScan). Phone A, C, F, G needs 10 s to sample a fingerprint sometimes. We calculate the most severe case F, and find the percentage of 10 s sampling is $12/103 \times 100\% = 11.7\%$. We also find phone G is funny, it will use 0 s to sample a fingerprint alternate with a 0.6 s sampling.

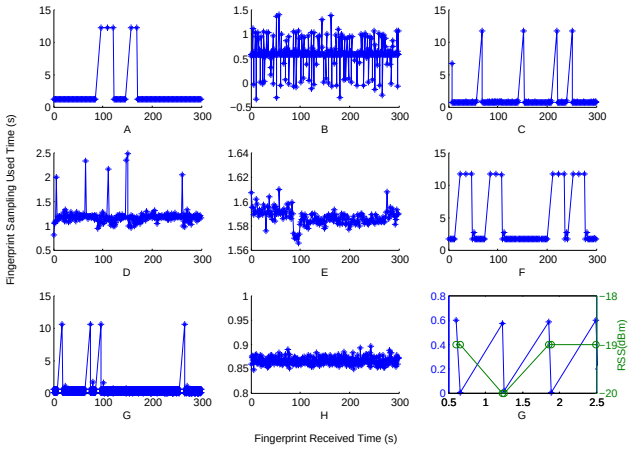


Figure 4: Scan Time by Using SDK API

We zoom in the G's plot, and append the plot of RSS it received from our pre-installed AP (also shown in Figure 2), we find that the 0 s sampling's RSS is same as the 0.6 s RSS. We think when the phone calls startScan the under layer will do scan and immediately return a SCAN_RESULTS_AVAILABLE_EVENT, which should not occur on logical, until the hardware finished the scan, it will report one EVENT, which will eventually cause the SCAN_RESULTS_AVAILABLE_EVENT. Phone B is also curious as it will uses < 0 s to sampling a fingerprint, we think it is caused by the background scanning and message queue mechanism, further explanation is left on our opensource project. 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 previous theoretical analysis which is 400-500 ms (we also use an third PC to monitor the phone/pads' Probe Request frame to confirm they are doing active scanning in another separated experiment).

So is the difference caused by the android OS ? And whether can we improve the scanning speed using some lower android api? In the next section we dig the android WiFi scanning architecture, and to find the key to these questions.

B. *Dig Android WiFi Scanning Architecture*

The Android WiFi Scanning Architecture can be divided into following parts: SDK API, Android Framework, JNI Layer, HAL Layer, wpa_supplicant and network device driver. Figure 5 shows the more detailed partition graphically.

Thread.run, eloop_run and inter layer socket communication mean time delay. So we may want to set some time anchors to these places. The most efficient way is setting some anchors on the most above part and most below part of the architecture first. As we already set the anchor on the most above SDK API previous, the next step is setting an start time anchor at the line *if (os_strcmp(buf, "SCAN") == 0){...}* in the function *wpa_supplicant_ctrl_iface_process*. And setting an receive time anchor below the *case NL80211_CMD_NEW_SCAN_RESULTS: ...* in the function *do_process_drv_event*.

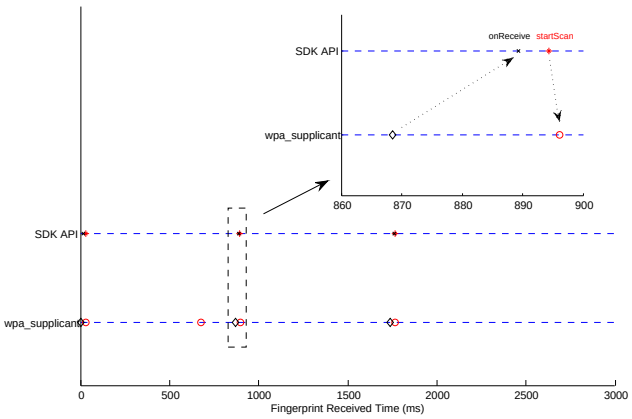


Figure 6: Wpa_supplicant Sampling Speed

We conduct this experiment in Phone H, and the results are shown in Figure 6, from the right above of the Figure 6, we find the time used by the android OS above the wpa_supplicant server is about 25 ms. By removing the start time on wpa_supplicant fired by the android background scan, like the second red circle on Figure 6's wpa_supplicant line, we get the calculated average fingerprint sampling time is

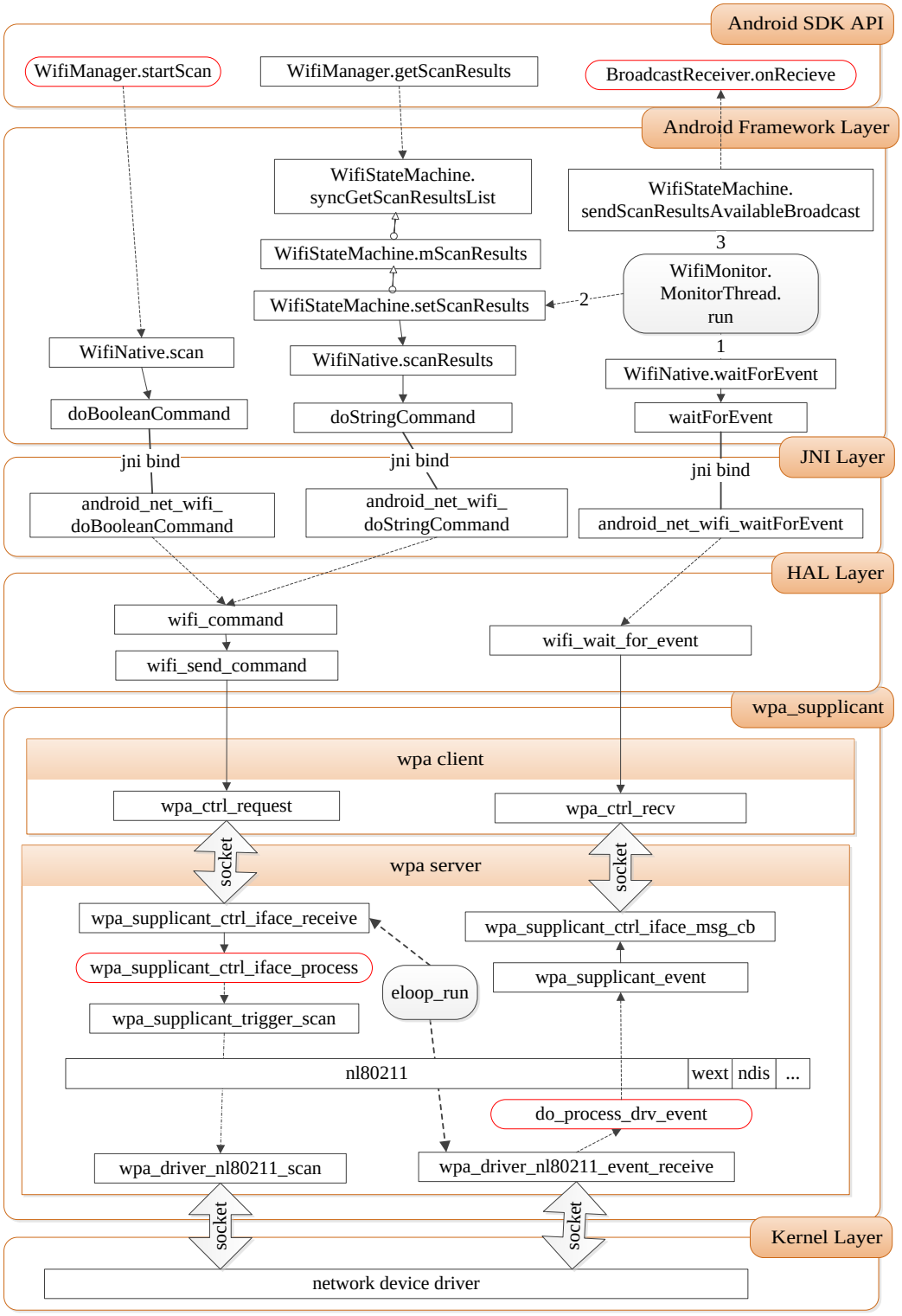


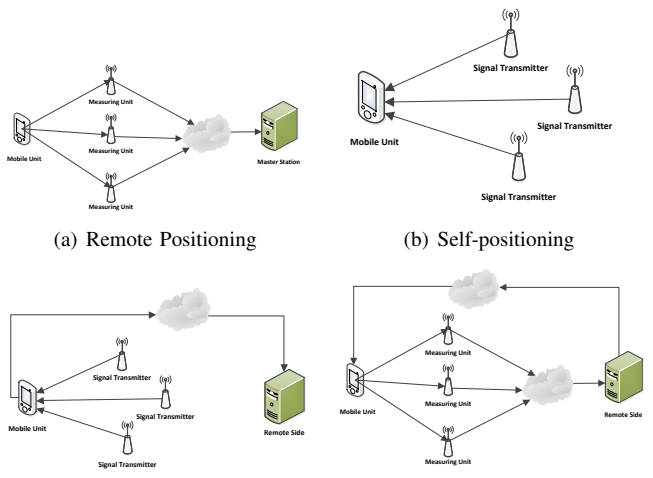
Figure 5: Android WiFi Scanning Architecture

841.7 ms. The experiments of the other phone/pads are delayed to our opensource project, and won't discussed at here now.

Previous experiment results show that we can hardly make the android to scan more faster even use the bottom layer of android. But all the previous discuss is based on the general application scenarios, STA samples the fingerprint and all channels have to be scanned. But in some special scenarios we can change the WiFi location system architectures and fingerprint sampling strategies to improve the sampling speed. We analysis the fingerprint sampling strategy at different system architecture in next section.

IV. Fingerprint Sampling Strategies on Different System Architecture

Considering where the location calculation will be conducted, location systems can be divided into four different system topologies according to [15] and [16]. The first one is the *remote positioning* system, whose signal transmitter is mobile (in our case, it can be thinking as STA) and several fixed measuring units (also it can be thinking as AP in our case) receive the signal emitted by transmitters. The results from all measuring units are collected, and the location of the transmitter is computed in a master station. The second is *self-positioning* in which the measuring unit is mobile. This unit receives the signals of several transmitters in known locations, and has the capability to compute its location based on the measured signals. If a wireless data link is provided in a positioning system, it is possible to send the measurement result from a self-positioning measuring unit to the remote side, and this is called *indirect remote positioning*, which is the third system topology. If the measurement result is sent from a remote positioning side to a mobile unit via a wireless data link, this case is named *indirect self-positioning*, which is the fourth system topology. We show all these four topologies in Figure 7.



(a) Remote Positioning (b) Self-positioning

(c) Indirect Remote Positioning (d) Indirect Self-positioning

Figure 7: Four Different Location System Topologies

Considering our control permissions to the APs, we add two division criteria, AP channel set permission and AP fingerprint sampling permission. AP channel set permission is that we can set the APs in the location service area working at specifical channels. And if APs can report the RSS sampling from STA (AP can measure the RSS of STA's ProbeRequst frame) and we can collect these information to the Master Station, then we say 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 we have any permissions to the APs. So we should conduct the scanning procedure on every channel one after another. The time we will use is analyzed in section II and measured in section III.

But if we have the permission to set the AP working channel, then we can set all the APs in less than the max number of available channels defined by each country. For example, we can just use the 3 no overlap (actually they are still a little overlap) channel 1, 6, 11 to cover the location service area. In these circumstance, I 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? Yes, the STA can dwell on each channel 0 time, and eventually we will need $(N_{channel} - 1) \cdot T_{switch}$ time to scan all channels, and we can just use $2 \cdot T_{switch}$ time to scan 3 channels, if we also have the AP working channel set permission.

V. Conclusion

In this paper, we study the WiFi fingerprint sampling problem deeply. We theoretically analysis the fingerprint sampling procedure, and calculate the fingerprint sampling speed. Fingerprint sampling speed experiment using Android SDK API programming is conducted on 8 phone/pads, with the sourcecodes to be opensource, we believe more measurements will be conducted on different phone/pads. In order to explore whether we can sampling WiFi fingerprint more quick, we analysis the android WiFi sampling architecture and measure the fingerprint sampling speed at different layers. With the problem whether we can sample WiFi fingerprint even more fast and considering multiple different location system topologies and our permission to the deployed location system, we shows in some systems we can even control the STA dwell time in 0 and improve the fingerprint sampling speed dramatically compared to the general location systems. We think our systematical analysis and practical measurement results will contribute to the community and can provide the reference to the practical WiFi location system designing and deploying.

References

[1] Ekahua. Ekahau Website. http://www.ekahau.com/. 2013.5.
[2] Cisco. Wi-fi location-based services 4.1 design guide. http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/lbswifig_external.pdf. 2013.5.
[3] Chunyi Peng, Guobin Shen, Yongguang Zhang, et al. Beepbeep: a high accuracy acoustic ranging system using cots mobile devices. SenSys, New York, USA, 2007, 1-14.
[4] Paramvir Bahl, Venkata N. Padmanabhan. RADAR: An in-building rf-based user location and tracking system. INFOCOM, Tel Aviv, Israel, 2000, 775-784.

[5] Moustafa Youssef, Ashok K. Agrawala. The horus wlan location determination system. MobiSys, Washington, USA, 2005, 205-218.
[6] He Wang, Souvik Sen, Ahmed Elgohary, et al. No need to war-drive: unsupervised indoor localization. MobiSys, Ambleside, Lake District, UK, 2012, 197-210.
[7] Zheng Yang, Chenshu Wu, Yunhao Liu. Locating in fingerprint space: wireless indoor localization with little human intervention. MOBICOM, Istanbul, Turkey, 2012, 269-280.
[8] Shih-Hau Fang, Tsung-Nan Lin. A dynamic system approach for radio location fingerprinting in wireless local area networks. *IEEE Transactions on Communications*, **58**(2010)4, 1020-1025.
[9] 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.
[10] Ishwar Ramani, Stefan Savage. Syncscan: practical fast handoff for 802.11 infrastructure networks. INFOCOM, Miami, FL, USA ,2005, 675-684.
[11] Arunesh Mishra, Minho Shin, William A. Arbaugh. An empirical analysis of the ieee 802.11 mac layer handoff process. ACM *SIGCOMM Computer Communication Review*, **33**(2003)2, 93-102.
[12] Haitao Wu, Kun Tan, Yongguang Zhang, et al. Proactive scan: Fast handoff with smart triggers for 802.11 wireless lan. INFOCOM, Anchorage , Alaska , USA, 2007, 749-757.
[13] Jin Teng, Changqing Xu, Weijia Jia, et al. D-scan: Enabling fast and smooth handoffs in ap-dense 802.11 wireless networks. INFOCOM, Rio de Janerio, Brazil, 2009, 2616-2620.
[14] Xi Chen, Daji Qiao. Hand: Fast handoff with null dwell time for ieee 802.11 networks. INFOCOM, San Diego, CA ,USA 2010, 1-9.
[15] H. Liu, Houshang Darabi, Pat P. Banerjee, et al. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics*, **37**(2007)6, 1067-1080.
[16] Christopher Drane, Malcolm Macnaughtan, Craig Scott. Positioning gsm telephones. *IEEE Communications Magazine*, **36**(1998)4, 46-54.