2012 International Conference on Indoor Positioning and Indoor Navigation, 13-15th November 2012

Time-of-flight positioning using the existing Wireless Local Area Network infrastructure

Ivan Casacuberta
Independent Researcher
Barcelona, Spain
ivan.casacuberta@gmail.com

Alejandro Ramirez
Corporate Technology
Siemens AG
Munich, Germany
alejandro.ramirez@siemens.com

***Abstract*—The ubiquity of IEEE 802.11 WLAN as well as the advancements in location based services and applications make locating a device more important than ever.**

While proprietary localization systems exist, extending an already deployed WLAN infrastructure with localization capabilities would lead to lower investment costs. For implementations based on commercial off the shelf (COTS) hardware, most research concentrates on using the received signal strength indicator (RSSI). The RSSI is inherently nondeterministic in dynamic environments, which limits its use in real world scenarios.

This paper presents three measurement methods to obtain the time-of-flight of a WLAN frame exchange; more specifically the round-trip-time (RTT). Once the RTT is known, simple algorithms such as multilateration can be used to determine the position of the user. All three methods have been implemented. The first method uses the "time synchronization function", through which WLAN devices add timestamps to the received frames. The second method employs a higher resolution timestamp based on the CPU-clock. This timestamp is triggered directly by the hardware interrupt that is generated when sending or receiving a frame. The third and new method reads the clock of the WLAN chipset. Our implementations show how the measurements can be implemented on any WLAN device.

KEYWORDS: WLAN, Positioning, RTT, ToF, COTS, Localization

I. INTRODUCTION

The improvements in Smartphone technology have brought a new wave of applications to the normal user. One of the types of applications that have increased in importance is location-based services. Location based services use the location of a device or individual to provide specific information relevant to the user at said time. The determination of this location is done using different techniques, each providing a different level of accuracy and reliability. In addition, different services require a different accuracy; for example, a weather application might just need to know the city where the user is located, while a navigation application will require a higher granularity of the position.

The use of existing wireless communication infrastructure to implement location techniques leads to several advantages. One of the advantages is lower investment costs as no new wide deployment will be required. In addition, using the existing communication infrastructure will allow the users to keep their own devices. With the right implementation, the essential requirements such as good accuracy, availability and reliability can be covered.

The ubiquity of IEEE 802.11 Wireless Local Area Network (WLAN) devices, the maturity of the standard, and the excellent balance between area of coverage and the amount of infrastructure equipment needed, helps WLAN to main wireless communication system used for positioning. While not all WLAN localization systems work the same, most of these systems take advantage of the received signal strength indicator (RSSI), a measurement of the energy present in a wireless signal which is done by all WLAN hardware. Unfortunately, it is well known that the performance of such systems decrease their accuracy when changes in the positioning environment occur (e.g. people, furniture) [1].

This paper takes a different approach, and concentrates on using the time-of-flight (ToF) of a wireless signal, which is employed to obtain the distance between a WLAN device and a WLAN access points (AP). The main differentiator between our system and the state-of-the-art is that the proposed system uses commercial-off-the-shelf (COTS) hardware, implemented through minimal firmware modifications.

This contribution has two goals: the first goal is to propose a new measurement method, to implement it and to compare it to two other software-based solutions for ranging with ToF in WLAN. The second goal is to show how all three measurement methods can be implemented on any COTS device. The implementation of all three methods has been done on the same platform in order to obtain results that are directly comparable. The modifications made in order to obtain the time of flight are going to be only software-based.

The presented implementations are capable of retrieving the time-of-flight to calculate an accurate and stable position for any WLAN device without using specialized hardware on either side of the communication. We highlight once again that our system can be deployed through a firmware upgrade, bringing an instant added value to any existing infrastructure.

The structure of the document is as follows: section two, we will start mentioning related work that we consider relevant regarding hardware implementations that can measure the ToF using COTS technology.

Section three will explain the inner workings of each of the three measurement methods. The third measurement method is a new proposal presented in this document.

Section four will go into the implementation of each of the measurement methods. All three methods were implemented on the same hardware platform and work simultaneously. That way, a single measurement will result in a measurement using the three methods at one.

Section five are the conclusions, were we discuss the obtained results.

II. Related Work

A. Hardware-Based Solutions

Many proposals can be found in the state-of-the-art that perform measurements at the physical layer using specialized hardware [2], [3], [4], [5].

In addition, some authors have tried to obtain a measurement of the ToF of a wireless signal using minimal hardware changes to a communication device. Those implementations are able to measure this ToF providing an accuracy of around 1 meter. The reader can find the mentioned hardware-based solutions on [6], [7] and [8]. Our own proposal tries to achieve a similar functionality but with only changes to the device's firmware.

B. Software-Based Solutions

There are very few publications proposing a software-based solution to measure the ToF using standard communication hardware. One of the earliest can be found in [9]. We have selected the two we considered more promising to implement them on our test platform [10], [11]. A description of these ranging techniques can be found in section three.

III. Description of Ranging TEchniques

As mentioned before, two of the three different techniques proposed on this paper are based on the ToF measurement methods proposed by [10] and [11]. Strictly speaking, these methods estimate the round-trip-time (RTT) of a wireless signal, which is the time it takes for a wireless frame to be sent and a corresponding response to be received; the RTT represents double the ToF. A RTT measurement is performed on the WLAN Access Point (AP) taking advantage of a IEEE 802.11 data-ACK frame exchange. The AP sends unicast data frames to the mobile device and counts the time until the corresponding acknowledgement frame (ACK) response coming from the mobile device is received. The employed frames are generated by the Medium Access Control (MAC) layer (layer 2 of the OSI-Model); this has the advantage of avoiding the delays introduced by the different layers included on the process.

In order to compare the three different methods presented on this paper, all of them have been implemented on the same platform and can be obtained simultaneously and tested using the same transmission packets. The second reason for implementing them on the same platform is to show the feasibility of an all-software measurement of the RTT on any COTS WLAN hardware.

A. First Method (Siemens Corporate Technology Technique) [10]

This method obtains the RTT using a timestamp introduced on the MAC header of a wireless frame. This timestamp is part of the Timing Synchronization Function (TSF) of the IEEE 802.11 standard, which maintains a coarse synchronization between all members of a WLAN service set. A timestamp is collected locally by a WLAN AP as soon as a data frame is sent; an additional timestamp will be collected when the ACK corresponding to the data frame is received.

The main limitation of using this method is the accuracy of the timestamp. The time resolution of the TSF is 1μs, which corresponds to the time a wireless signal takes to cover 300m. In order to increase the accuracy of the distance estimation, additional statistical methods are used. Through the collection of several measurements and better granularity for the ToF can be obtained. While the simplest way of obtaining a higher resolution out of several measurements is by averaging, more advanced signal processing algorithms are presented on [10]; this paper uses the gauss filter proposed in the reference to calculate the results of the distance estimation.

There are two reasons why this measurement method works. The first reason is the low level of synchronization, which is limited by the TSF. This means that jitter of up to 1μs will affect the raw measurements obtained. The second reason is the presence of other sources of random noise, for example thermal noise or electromagnetic inference from other signals, which will also affect the raw measurements.

Figure 1 shows the measurements that would be obtained with and without any presence of noise. Without noise, the results obtained for a specific distance will not vary from one measurement to another. The result would lead to "steps" which makes it impossible to distinguish between distances close together.

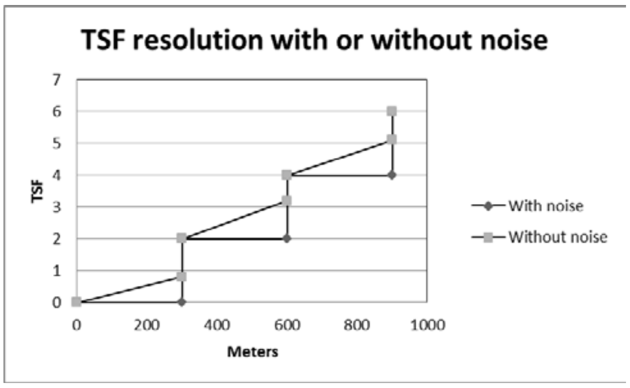


Figure 1. Effect of noise in the distance estimation

With the addition of noise, a simple average would provide the possibility of filling the gaps between the "steps".

While adding noise can help improve the resolution of the distance estimation, too much noise can have a degrading effect on the measurements. There is actually an ideal amount of noise which would produce the best accuracy for the distance estimation.

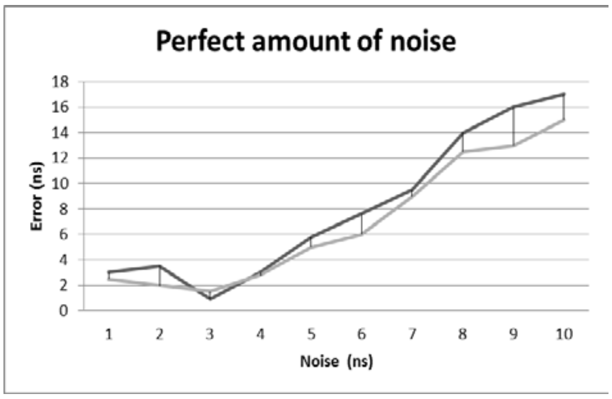


Figure 2. Example diagram of the implementation of the second method

An implementation of this first measurement method has an important limitation: the wireless frames that are transmitted don't contain a timestamp from the TSF. For this reason, it is necessary to have an additional WLAN device receiving all wireless frames; those sent by the AP and those sent by the client. In order to simplify the calculation of the distance, this additional device will be set close together to the AP.

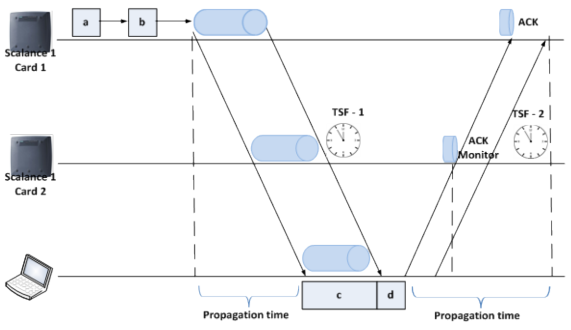


Figure 3. Diagram of the implementation of the first method. a – Packet TX command, b – Kernel, c – Processing Time, d – ACK.

A diagram of the functionality of this measurement method is presented on Figure 3. A wireless data frame is transmitted from an AP towards a WLAN client. This data frame will be received by the additional WLAN device. Once the transmitted packet reaches the receiver, an ACK is sent back to the AP. Before this ACK is sent, there is a processing time which adds a delay that must be taken into account to calculate a distance with good accuracy. The ACK is received by both the AP and the additional WLAN device with a minimal difference on time. The total time (T_{total}) is obtained by taking the time when the ACK was received by the additional WLAN device and subtracting it from the time when the data frame was received by this same additional WLAN unit. The processing time ($T_{process}$) of the WLAN client has to be subtracted too. Equation 1 shoes the steps for obtaining the distance out of this timestamps.

$$d = \frac{c \cdot (\ T_{total} - T_{process} \ )}{2}$$

Equation 1: Calculation of the distance using the obtained timestamps

B. *Second Method (Barcelona Tech Technique)[11]*

The main feature of the second method is that it exploits the higher time resolution that the CPU clock can offer in comparison to the use of the TSF described in the first method. While a typical CPU can easily achieve a resolution of 1ns (1 GHz), the TSF offers only a resolution of 1µs (1 MHz). Additionally, based on the fact that changing the source code of WLAN driver it is required, the necessary hardware interrupts to obtain a RTT measurement can be accessed with relative simplicity through software.

The mechanism to obtain the RTT is to introduce a timestamp when the MAC data frame is transferred by the driver to the WLAN hardware for the transmission and introduce another timestamp when the WLAN hardware receives the ACK frame, which corresponds to said data frame, is received. As with the first method, subtracting both timestamps represents the total time T_{total}. In order to avoid any hardware modification, this solution is implemented capturing the events mentioned above from the driver's code that controls the WLAN interface. The driver used was a specific distribution made by Atheros for Atheros-based WLAN interfaces with a Linux operating system (OS).

The implementation of this second method is presented on Figure 4, starts with the packet transmission order. This order is taken by the system which will automatically pass the request to the driver of the WLAN card. Once on the card driver, the function Ath_tx_start() is called in order to set the packet ready to be sent. Once the packet is prepared, and still inside Ath_tx_start(), the packet is added to the transmission queue using Ath_txqaddbuff(). This queue takes care of sending whenever it is possible. Once the packet can be sent, Ath_hal_tx_start() takes care of starting the process moving the packet to the hal() to physically start transmitting it. Once the packet has been completely sent, ath_intr() throws an interruption. This interruption is thrown in order to work with the sent packet for monitoring purposes. We take advantage of that idea to make the timestamp that later on is going to be used to calculate the RTT. This timestamp is set inside ath_intr(). With the timestamp not only the time is stored but also all the information that later on is going to be needed about the packet in order to correctly classify it. All this information is being stored inside a data structure. Then when the receiver has completely received the packet, it starts processing it, and afterwards the ACK is sent (this processing time, should be almost 0, because the ACK is sent from the physical layer, although an additional delay time is going to be added). Once the ACK has arrived completely to the original transmitter, an interruption is thrown. Once again we take care of capturing the time that this interruption is thrown inside the handler of the interruption (ath_intr()), which is going to be almost the first place in which it will get. After that, the process that the driver follows is no longer relevant for calculating the RTT.

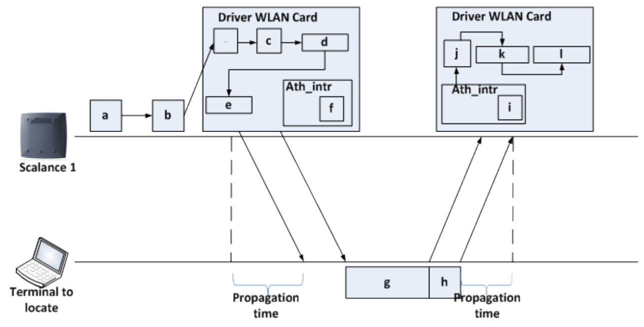


Figure 4. Diagram of the implementation of the second method. a – Packet TX command, b – Kernel, c – Ath_tx_start, d – Ath_txqadbuf, e – Ath_hal_txstart, f – TimeStamp, g – Processing Time, h – ACK, i – TimeStamp, j – Ath_rx_tasklet , k – ieee80211_input , l – ieee80211_deliver_data .

The employed time-base used for the timestamp corresponds to the CPU clock of the AP because it provides very good time resolution and it is accessible from the code of the WLAN card driver. Theoretically, this resolution would achieve ranging errors below one meter; however on the process described before there are delays that introduce error on the accuracy to obtain the time-of-arrival (TOA).

C. *Third Method (Hybrid Technique)*

This method is based on solving the limitation of method 1, which has to use an additional card on Monitor mode in order to obtain the TSF on the transmission. While this third method calculates the RTT using the TSF, it does not obtain this timestamp directly from the packet. Instead it calls a card function, which takes care of obtaining the card clock time. In order to decide, which was the best place within the driver to obtain the card clock, different tests were made. The tests showed that similar results were obtainable, making the timestamp in 2 different places, which are going to be described below.

The third method starts when the order to send the packet is triggered. This order is passed to the kernel, which is going to forward it to the driver. The driver is going to start initializing the packet (ath_tx_start), and once this is prepared is going to add it to the transmission queue (ath_txqaddbuf). Here is where there are the different alternatives used to call the card clock. In the first case (A), we are making the timestamp just before calling the function which is going to pass the packet to the hal() to start sending it. It is known that in this point the packet is ready, and that is already in the queue. The only extra delay that we can have is the time that it takes to the hal() to send the packet. The second place where to put the timestamp (B), is going just after we return from the hal(). The problem here is that it is not really known if when returned from the hal() the packet has already been sent, or the packet is still sending. In both methods, once we have made the timestamp, we take care of storing all the information related with the packet, that we are going to need later on. This data is again stored inside the structure information. After the packet has been received by the receiving hardware, the ACK is sent back. When the ACK is arriving to the initial transmitter, what we are going to do is just read the TSF of this acknowledgment packet. This is possible because this packet is not being transmitted is being received, and in this occasion the TSF is not empty. The good part is that this timing is also going to be a read from the same clock, what make them able to subtract without having to make a synchronization between the transmitting device and the client clocks.

The next figure describes the way the interaction of the functions previously described in a more graphic way.

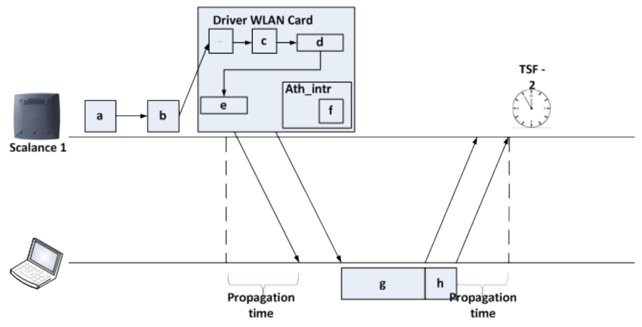


Figure 5. Diagram of the implementation of the third method. a – Packet TX command, b – Kernel, c – Ath_tx_start, d – Ath_txqadbuf, e – Ath_hal_txstart, f – TimeStamp, g – Processing Time, h – ACK.

IV. Experimental Results

A. *Setup*

A prototype with the described ranging techniques implemented has been developed and tested. This prototype takes care of transmitting the packets, obtaining the timestamps for each method as well as executing the statistical modules capable of determining the coordinates for the position of the mobile device. Additionally the system has been tested with a graphic module capable of indicating the position of the mobile device on a 2D map.

The prototype was implemented inside a device designed by Siemens AG which has been on the market for several years for different professional purposes but mainly as a WLAN AP. The kernel version of Linux inside the prototype is RAP2 2.6.15.4. In order to minimize the error due to the interruption management of the OS, the load of the CPU is restricted to the essential Linux OS kernel components. The WLAN card uses an Atheros chipset compatible with an Atheros Driver. The mobile device was implemented using a laptop with an Intel Centrino Pentium M processor 760 (CPU 2 GHz), with an Ubuntu Edgy Eft 6.10 OS.

In order to obtain the highest accuracy, an important decision to make was which type of packets to use, and how many packets to send. Several trials were made, testing different type of packets and different parameters and for their generation including packet size. An important factor in the decision was that many packets per second were required; the faster the measurements arrived the faster the position can be obtained.

Beacons were considered, because they are automatically sent by all AP; however, they are only 10 times per second and, because they are addressed as broadcast frames, they don't receive an answer. A further possibility was to use the association packets forcing the receiver to not accept the association and with that be able to send repeated packets with an answer. The problem was the rate at which those packets are being sent is lower than the one that it would be desired by our system. Finally two different solutions were tested, in order to choose the one which could offer the best results

On the one hand, we tested ICMP packets (more specifically ping packets) and on the other hand empty messages sent through a socket to an open port of the receiver. Both solutions where tested under the same environmental circumstances, using the same amount of packets for several different distances. They were also tested with the three ranging methods exposed on this paper.

In order to decide if the type of packets could be used for ranging purposes, it was checked whether the time that takes to the packet to travel from the transmitter to the receiver and back, grows when the distance is increased.

Finally ICMP pings were proved to be the ones that offered better stability within the three methods described above as well as a lineal grew which granted a correct estimation of distances either short or long. Additionally, simple tools exist for generating this type of packets on different platforms.

Once the type of packets was decided, it was necessary to find the most appropriate amount of packets to send, in order to affect neither the performance nor the accuracy. To do so, using different distances, and a big amount of ICMP packets, the average accumulative was checked to found from which amount of packets it started the time stability. This test was done under the same circumstances as the packet election, and using the three methods exposed on this paper as well. It was proved that the minimum amount of packets to obtain good results was between 1500-3000 which can be obtained with between 10-20 seconds. With those tests it was determined the first point from which there was enough stability in order to get the best accuracy.

As the implementation of the second and first method requires enhancing the driver of the WLAN interface, it was necessary to find the spot which delivered the highest accuracy for the distance estimation. In addition, different programming commands provide different sources for the clock signal (i.e. CPU clock, bus clock), so additional tests were required. In the case of the implementation of the third method, the function that showed the best performance is a specific one found in the Atheros driver, which, of course, makes this timestamp command specific to the driver. On the election of the clock source for the implementation of the method, Linux offers different alternatives which were closely considered. The final election was the function rdtscl(), which returned the lowest 32bits of the CPU Time Stamp Counter (TSC). This counter is saved using 64bits, however, for the measurements needed we are going to have enough with the 32 lowest bits.

An indoor measurement campaigns was carried out in order to prove and define the RTT statistical processing and to test the feasibility and performance of the ranging technique. The measurements were carried out in the corridor in Siemens Corporate Technology in Munich. The test environment was a business complex characterized by challenging conditions such as metallic walls, doors and ceilings, as well as people moving around that blocked the LOS at unexpected intervals. Previous tests and analysis showed us that this blocking of the LOS through people has no statistically identifiable effect on the measurements. Both the mobile station (MS) and the AP are placed 1.5 m off the ground in order to preserve the Fresnel zone. All devices where connected to a power outlet; no battery power was used. The devices were mounted of chairs or tables setup to the height mentioned before..

In the following figure 6, figure 7 and figure 8 we present a histogram of the raw measurements using each of the three measurement methods. Each histogram has been built out of about 15000 measurements for each distance. As mentioned throughout this document, all three methods work simultaneously on the same hardware, which means that only 15000 packets had to be exchanged for each distance. To highlight the effect that distance has on the raw measurements, we present the values corresponding to only three distances.

The horizontal axis shows the RTT of the signal, measured in clock cycles. The duration of a clock cycle is dependent on the clock speed. The vertical axis shows the frequency of the measurements, meaning how often out of 100% the measurements had the value represented in the horizontal axis.

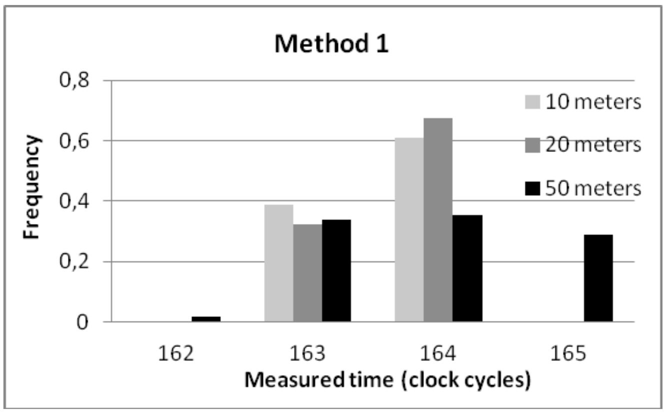


Figure 6. Histogram using the first measurement method

Figure 6 presents the raw measurements using the first method. The histogram of each distance is presented on the same figure, using different shades of grey.

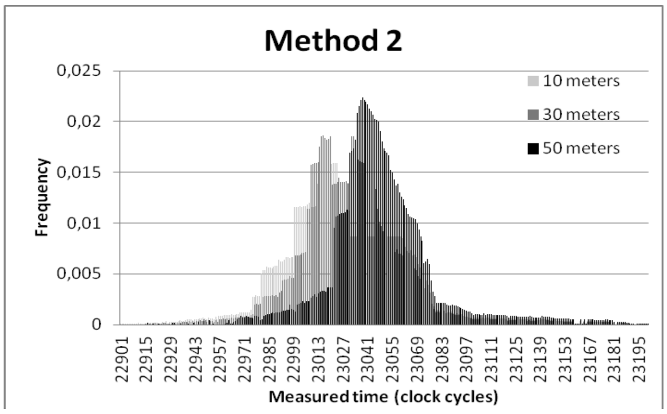


Figure 7. Histogram using the second measurement method

Figure 7 presents the raw data when using the second measurement method. Because of the higher clock resolution of a CPU clock in comparison to the clock of a WLAN device, the amount of values on the horizontal axis are significantly more than the ones in figure 6. For this same reason the range of the vertical axis is lower.

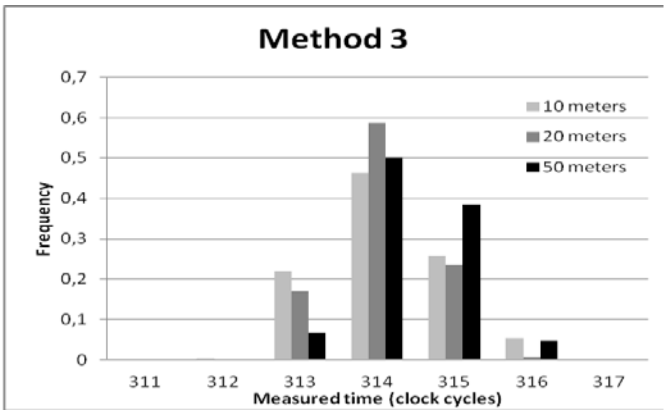


Figure 8. Histogram using the third measurement method

Figure 8. shows the histogram of the raw measurements using the third measurement method. The horizontal resolution, which is directionally proportional to the clock speed, is higher than the one presented on figure 6 and lower than the one seen on Figure 8.

Based on the figures 6, 7 and 8, we have defined a simple filter to crop any outlier outside of the range presented in the figure. That will avoid any disruptive effects from outliers which would correspond to implausible distance values. This is especially important for the second measurement method. As this method uses the CPU clock, which is heavily influenced by the amount and type of tasks running in the OS, outliers corresponding to absurdly high values could sometimes be seen. In the case of the other two measurement methods, the improvements of this simple crop filter were less noticeable.

The main purpose of these three figures is to show that on all three methods the histogram “slides” to the right as the distance increases. The bigger the change in the distance is, the larger the effects. For small distance changes, the effect will be less, down to the point where distances can no longer be told apart. The sliding of the histogram presents a linear relationship between the RTT and the distance, just as expected from the theory. This linear tendency will be seen on the results of the measurement campaign described in the next paragraph.

Next, the performance of the distance estimations will be compared using all the methods. For such tests, our experiences has taught us the importance of using a “blind” process for doing the measurements and processing the data. To achieve this, distances were selected between 0m and 50m using an online random number generator. The distances are 3, 12, 13, 18, 24, 38 and 50 m. In addition, to avoid any systematic effects from the hardware into our measurements (for example the effect of the increase in temperature after the hardware has been in operation for a while), the distances will be measured in random order. No online signal processing was available at that moment, so the raw data was stored to do an offline signal processing. Each position includes about 3000 raw measurements, which take a few seconds to make.

There was a LOS between the MS and the AP at the point when the measurement was done. However there were people present in the hallway during the measurements, which weren’t blocking the LOS.

There were other wireless networks in the building using the same frequency which might have had an effect on our measurements. That is one of the reasons we did thousands of measurements and why we measure with all methods in parallel.

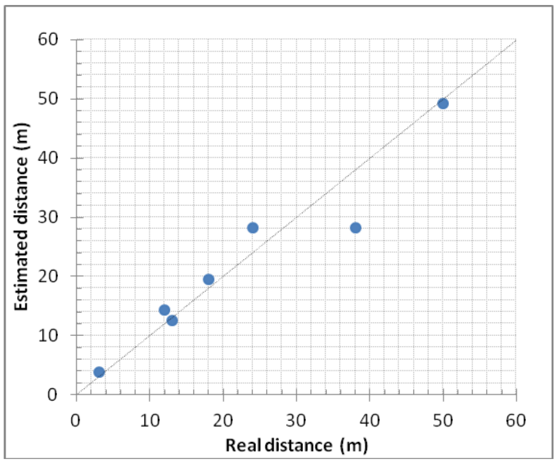


Figure 9. Estimated distance using the first method

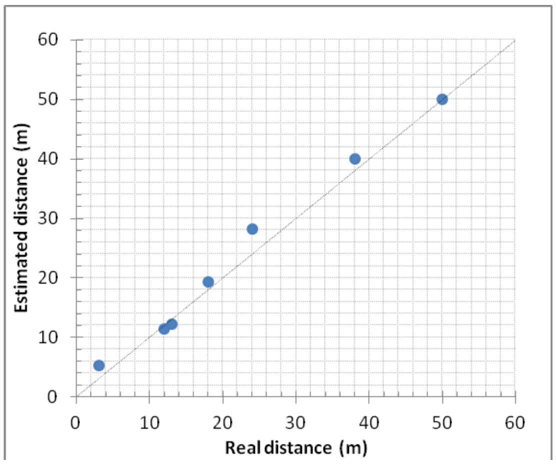
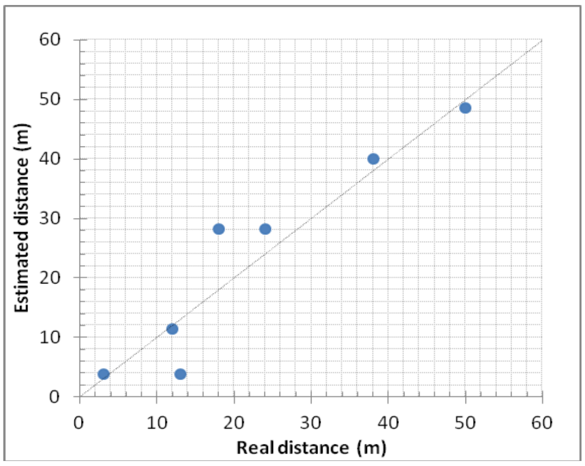


Figure 10. Estimated distance using the second method



Estimated distance using the third method

Based on the results, a clear tendency can be recognized. Even though the results are somewhat similar, the results of the second measurement method show a better estimation of the real distance. The mean error of this second method, in the environment described in the previous section was 1,5 meters. While this is the result of a single measurement campaign, other measurements have shown this accuracy to be typical value.

The first measurement method takes the next spot regarding quality of the results. While it presents an average error of 2.8 meters, there is an important outlier corresponding to the distance of 38 meters. While it would be easy to indicate an environmental change (reflection, or RF interference), we have to mention again that the measurements of all three methods are affected by the exact same environmental conditions. As no noticeable effects can be seen on the same distance for the other measurement methods, this outlier has to be linked to the hardware clocks used. Even further, the third measurement method uses the same clock for measuring the arrival of the wireless frames, leaving the clock which measured the timed at which the frame was sent as the only explanation for the false measurement.

In the last case, the case of the third measurement method, the results obtained show an average error of 4.4 meters. Two outliers were present, which correspond to the distance of 13m and 18m. In this case, the outliers don't correlate to the errors presented using the other measurement methods. This makes us think again that the source or error is not the environment, but the hardware.

V. CONCLUSIONS

In this paper we have presented three measurement methods for achieving a distance estimation using standard WLAN hardware. Each method has specific advantages and disadvantages in their implementation as well as different performance.

To prove their capability of running on standard hardware, we have implemented all three methods to run in parallel on a chosen WLAN AP. Through this implementation, a single WLAN frame exchange will trigger three timestamps, one per measurement method. As the instantaneous environment conditions are the same for all three measurement methods, this will make them directly comparable.

Indoor measurement tests were done in a business complex to show their performance. Out of the three, the second method showed the best performance with a mean error of 1,5 meters. This method uses the CPU clock of the AP to do the measurements, whose clock is the highest out of the three measurement methods. However, the implementation of this method requires access to the source code of the driver of the WLAN hardware used.

The first method presented was next in the performance rating, presenting an average error of 2.8 meters. This method has the advantage that no access to the driver is required. However, this method uses a requires WLAN card.

The third method, which is a combination of the first two, takes the last place with an average error of 4,4m. This method requires only one WLAN card, but it showed more outliers than the rest.

All this results prove the feasibility of implementing a ToF positioning system using existing WLAN infrastructure by just updating the firmware.

REFERENCES

[1] Elnahrawy Eiman, Li Xiaoyan and Martin Richard P.: The limits of localization using signal strength: A comparative study. 1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks. - 2004. - pp. 406-414.

[2] Li, X., Pahlavan, K.: Super-Resolution TOA estimation with diversity for indoor geolocation. IEEE Trans. on Wireless Communications 3(1), 224–234 (2004)

[3] Aassie, A., Omar, A.S.: Time of arrival estimation for WLAN indoor positioning systems using matrix pencil super resolution algorithm. In:

Proc. of 2nd Workshop on Positioning, Navigation and Communication, March 2005, pp. 11–20 (2005)

[4] Ibraheem, A., Schoebel, J.: Time of arrival prediction for WLAN systems using prony algorithm. In: Proc. of 4th Workshop on Positioning, Navigation and Communication, March 2007, pp. 29–32 (2007).

[5] Reddy, H., Chandra, G.: An improved time-of-arrival estimation for WLAN-based local positioning. In: Proc. of 2nd International Conference on Communication Systems software and middleware, January 2007, pp. 1–5 (2007)

[6] McCrady, D.D., Doyle, L., Forstrom, H., Dempsey, T., Martorana, M.: Mobile ranging using low-accuracy clocks. IEEE Trans. on Microwave Theory and Techniques 48(6), 951–958 (2000).

[7] Golden, S.A., Bateman, S.S.: Sensor Measurements for Wi-Fi location with emphasis on time-of-arrival ranging. IEEE Transactions on Mobile Computing 6(10) (October 2007)Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[8] Ciurana M, López D, Barceló F.: SofTOA:Software ranging for TOA-based positioning of WLAN terminals: LoCA 2009, LNCS 5561, pp. 207-221, (2009).

[9] Günther A, Hoene C: Measuring round trip times to determine the distance between WLAN nodes. Technical Report TKN-04-016 / Telecommunication Networks Group ;Technische Universität Berlin. (2004).

[10] Ramirez, Alejandro: Time-of-flight as an information source for positioning. PhD Thesis, (2010)

[11] Ciurana, M., Barceló, F., Izquierdo, F.: A ranging method with IEEE 802.11 data frames for indoor localization. In: Proc. of Wireless Communications and Networking Conference, March 2007, pp. 2092–2096 (2007).