WLAN Location Determination without Active Client Collaboration

Stefan Kraxberger
Graz University of Technology
Graz, Austria
stefan.kraxberger@iaik.tugraz.at

Guenther Lackner
studio78.at
Graz, Austria
guenther.lackner@studio78.at

Udo Payer
University of Applied Sciences
Graz, Austria
udo.payer@campus02.at

***Abstract*—Location determination of devices in wireless networks has been around for some time. Due to the fact that most approaches are based on client collaboration, they are not suitable for being integrated in security related components as intrusion detection or access control systems. In this paper we propose an approach that works without any client collaboration and even awareness by the client. Further on, our architecture is intended to be deployed on standard state-of-the-art IEEE 802.11g access points which operate using open source firmware as OpenWRT. We use these modified access points to collect, process and relay *received signal strength* (RSS) values of all communicating clients. These values are then used for trilateration using a location determination algorithm on a centralized server. We also present a working prototype, its promising results and possible further enhancements of our approach.**

***Index Terms*—location determination; wireless networks, trilateration; intrusion detection; access control**

I. Introduction

Due to the nature of free air propagation, which all radio communication is based on, wireless computer networks may be easy accessible for potential attackers. Infrastructure bound wired solutions do have an advantage in terms of security over wireless ones, due the inevitable physical connection to the target network. The possibility to determine the physical location of any connected client would result in a significant gain of the achievable security level.

Manifold location determination methods exist. Unfortunately many of them are not suitable for IEEE 802.11 based networks and their typical field of application [1]. Further on, most of them require special purpose hardware as for example GPS receivers or cell phones [2]. Other classical approaches like run-time based triangulation are very problematic to implement because of the limitations of timing devices in wireless network hardware, although it might be possible with future IEEE 802.11n compatible chipsets. Due to this fact, solutions based on the analysis of *received signal strength* (RSS) values are widely regarded as best practice for WLAN device location determination. For examples refer to [3] [4].

Further, location determination methods can be distinguished between client and network based methods. Client based methods usually require some kind of agent software installed on the client and a server component that collects data from these agents. They also calculate their approximate position [5] directly on the client. The later work without any collaboration from the clients. This can be seen as a significant advantage over the client based methods.

Client based solutions are very well elaborated considering the accuracy of their positioning algorithms. One solution was proposed by Robert A. Malaney from the University of New South Wales [6]. Collected RSS values from the system are compared with a previously computed database of RSS fingerprints, in order to get the current location. However, as mentioned before, client cooperation is limiting the usage of location determination in a security system. Attackers would be able to spoof RSS values, and fake their positions. Therefore, an approach which isolates the gathered position information completely from the cooperation of the client is the preferred solution. The method we present in this paper works absolutely without any client collaboration and notice, and is therefore well suited for a security related application like an intrusion detection or access control system. All data collection tasks are performed by the network infrastructure itself, which is achieved by modifying the open source firmware of IEEE 802.11 access points.

This work is organized as follows. First we introduce the architecture of our solution and how we implemented the data gathering and distribution process. Thereafter, we provide some background information on position determination and trilateration. Finally, we present results obtained from a real-world test environment.

II. Architecture

This section describes the architecture of our approach and how it can be used to provide valuable information for intrusion detection systems, access control mechanisms or any other component which enforces security policies as proposed by [7]. The idea behind the positioning system is to detect the location of a possible client within a wireless network. All higher layer communication with the network is based on the devices MAC address. All MAC addresses appearing in the network are placed on a list of connected devices. This is done by modified access points (AP), which directly communicate with a centralized component placed on a server connected to the backbone of the network. These APs, we call them drones, collect all frame headers that contain the RSS values of the corresponding transmission and submit them, together with the MAC address of the belonging device to the central server. Based on these values, the server computes a RSS fingerprint of the devices and compares it against entries from a pre-calculated database of RSS fingerprints. For an accurate determination of the client's position, the server needs RSS values from at least three drones in communication range of the client.

The server can now inform the security systems of the location of all clients in order to allow them to enforce security policies regarding the location of accessing systems.

A. *Positioning*

Figure 1 shows an example network consisting of three APs and one client. This represents the minimal configuration to get accurate information to determine the client's location. An easy way to collect the RSS values would be to execute a `iwlist` (in case of a Linux operating system) scan on the client. But since we don't have collaborating clients, this approach is not possible. Thus, we use the drones to obtain the signal strength values which can be used to determine the location of the client. For this purpose all drones need to be in a listening mode and collect all received frames.

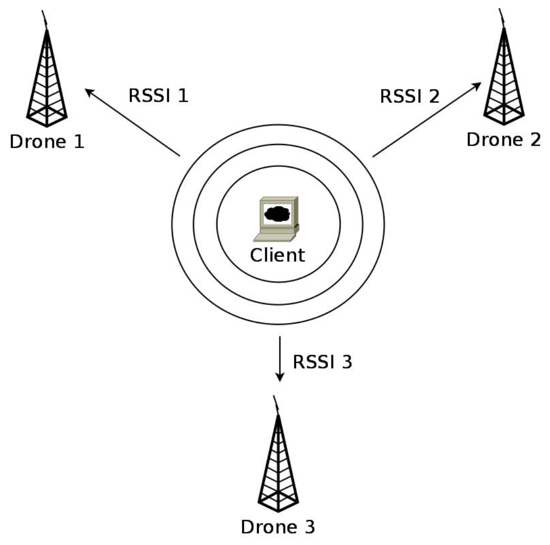


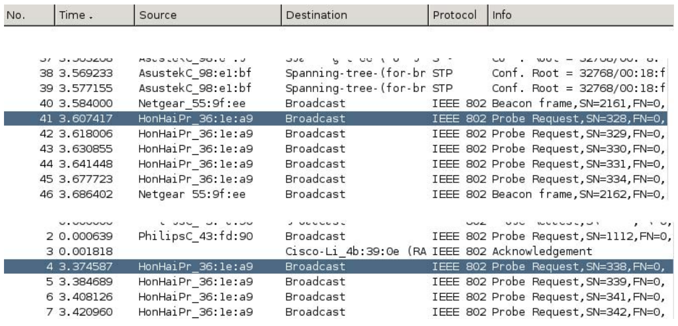
Fig. 1. Wireless network configuration

B. *Obtaining signal information*

As a first step, it is necessary to capture transmissions from the communicating client and store them for later analysis. This is done through the use of the `tcpdump` tool, which is based on the `pcap` library. We capture the data frames from the client with the following command.

```
tcpdump -i network-interface -s 1500 -w out.pcap
```

Due to the fact of memory limitations on the drones, it is necessary to filter the captured information. `tcpdump` allows to filter the received packets using combinations of various criteria so that only relevant parts are captured and stored for further processing. In figure 2 we can see the RSS value and the location in the frame.



```
No.  Time .    Source              Destination            Protocol  Info
  38 3.569233  AsustekC_98:e1:bf   Spanning-tree-(for-br  STP       Conf. Root = 32768/00:18:f
  39 3.577155  AsustekC_98:e1:bf   Spanning-tree-(for-br  STP       Conf. Root = 32768/00:18:f
  40 3.584000  Netgear_55:9f:ee    Broadcast              IEEE 802  Beacon frame,SN=2161,FN=0,
  41 3.607417  HonHaiPr_36:1e:a9   Broadcast              IEEE 802  Probe Request,SN=328,FN=0,
  42 3.618006  HonHaiPr_36:1e:a9   Broadcast              IEEE 802  Probe Request,SN=329,FN=0,
  43 3.630855  HonHaiPr_36:1e:a9   Broadcast              IEEE 802  Probe Request,SN=330,FN=0,
  44 3.641448  HonHaiPr_36:1e:a9   Broadcast              IEEE 802  Probe Request,SN=331,FN=0,
  45 3.677723  HonHaiPr_36:1e:a9   Broadcast              IEEE 802  Probe Request,SN=334,FN=0,
  46 3.686402  Netgear 55:9f:ee    Broadcast              IEEE 802  Beacon frame,SN=2162,FN=0,

   2 0.000639  PhilipsC_43:fd:90   Broadcast              IEEE 802  Probe Request,SN=1112,FN=0,
   3 0.001818                      Cisco-Li_4b:39:0e (RA  IEEE 802  Acknowledgement
   4 3.374587  HonHaiPr_36:1e:a9   Broadcast              IEEE 802  Probe Request,SN=338,FN=0,
   5 3.384689  HonHaiPr_36:1e:a9   Broadcast              IEEE 802  Probe Request,SN=339,FN=0,
   6 3.408126  HonHaiPr_36:1e:a9   Broadcast              IEEE 802  Probe Request,SN=341,FN=0,
   7 3.420960  HonHaiPr_36:1e:a9   Broadcast              IEEE 802  Probe Request,SN=342,FN=0,
```

Fig. 2. Received signal strength on different routers obtained via Wireshark

The stored packets are thereafter sorted by their source MAC address using `tcpdump` in order to enable further processing of the data with custom perl scripts. The relevant fields of the stored packet information are the source MAC address, the MAC timestamp and the RSS value.

```
tcpdump -e -c 1 -i prism0 -xx -s 1500 ether
      src XX:XX:XX:XX:XX:XX
```

The time and signal strength values are translated into the decimal system. Because the RSS values can strongly variate, we calculate an average over every 10 samples. This in addition reduces the amount of data transmitted and reduces the chance of traffic congestions. After calculating these average values, they are sent to the server. A major challenge, was to balance the memory consumption of the framework due to the limited resources on the drones. Simultaneous data collection and processing was not possible, hence after receiving a packet it is processed directly and all other incoming packets during the time needed for processing are dropped. The results obtained within our testbed show that this method has adequate performance.

C. *Router-Drone communication*

To perform the trilateration, we need to transmit information gathered at the drones to the central server. In our prototype we used the UNIX tool `netcat` to perform this task. The server listens on an open port for any incoming data from the drones. All communication can be regarded as one-way since there is no need for the server to send anything to the drones but transmission protocol

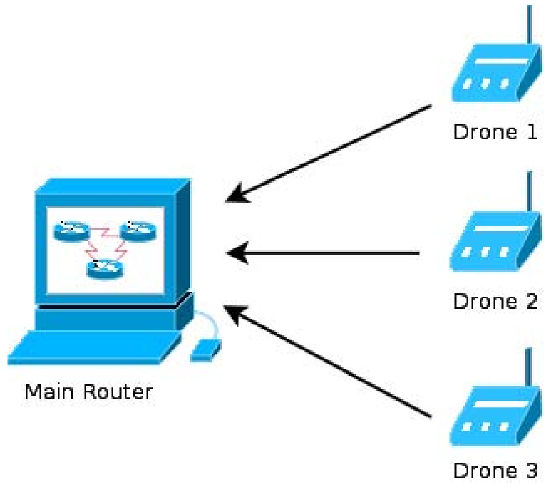


Fig. 3. Server-Drone Communication

related data for `netcat`. Due to the fact, that the traffic between the drones and the server is at a very low level, no sophisticated collision control mechanisms need to be applied.

Figure 3 shows the schematic layout of the server-drone communication. Due to the fact that there are many interferences like signal reflections from walls and attenuation caused by moving objects, we need to compare measurements of the same timestamps from the same clients collected by the different drones. Without synchronization, the time deviation is to large to find packages that belong to the same transmission. Therefore, a method to synchronize the clocks on the drones is absolutely necessary. For this task we used the `openntpd` tool.

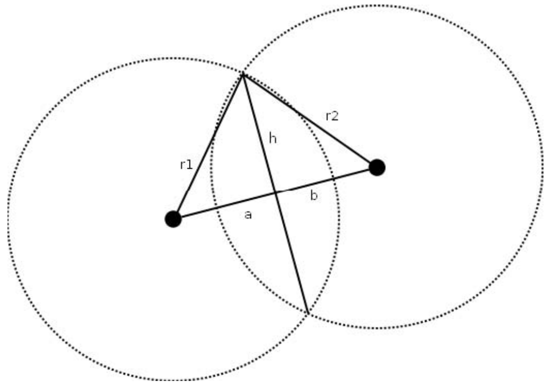


Fig. 4. Intersection of two circles

D. Trilateration

As said before, we apply the method of trilateration for position determination. To calculate the position of a wireless network client we intersect three circles with the signal power as radius. The computation itself is done in two steps. For the intersection, we first need the distance to the center of two circles. The values defining the position of the first center is given as $x1$ and $y1$. The second center is defined with $x2$ and $y2$. In the next step we use formula 1 to calculate the distance.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

In figure 4 the intersections of two circles are shown graphically. The two triangles are defined in equation 2 and 3.

$${r_1}^2 = a^2 + h^2 \tag{2}$$

$${r_1}^2 = b^2 + h^2 \tag{3}$$

With the distance d and the two radii, the value a can be calculated in equation 4.

$$a = \frac{({r_1}^2 - {r_2}^2 + d^2)}{2d} \tag{4}$$

With the value a, the value for h can be easily computed with formula 5.

$$h = \sqrt{{r_1}^2 - a^2} \tag{5}$$

This is repeated three times, so we get the 6 different intersections as shown in figure 5. The dotted line shows the circle, which is ignored at each step. Unfortunately, in common wireless networks with standard hardware only very noisy signal strength measurements can be obtained. As we can see in figure 6 there are positions where the distance has no relation to the measured values. This two-way-ground propagation model was obtained by Appel et al. and presented in [8]. Additionally, due to disturbing sources like walking humans and walls that attenuate the signal, we cannot get a signal that has full relation to the distance.

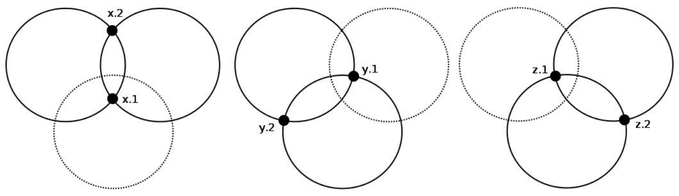


Fig. 5. Three different-intersections

For the trilateration algorithm, we create three circles withs radii representing the RSS values from at least three drones belonging to the same client transmission. Figure 7a shows the trilateration approach based on signal strength values. These circles will intersect only if the values measured at the different drones do not deviate from each other. For example, if we measure a wrong value for RSS 1 this system will not intersect and will return a failure. In figure 7b the doted circle is drawn with the wrong signal strength. As one can easily see there is no intersection which covers all three RSS values. Due to the fact that these errors occur frequently, another way needed to be found to calculate the location correctly. For that reason we use and iterative process. We divide the radius of each circle by a special ratio, the so called *signal ratio*.

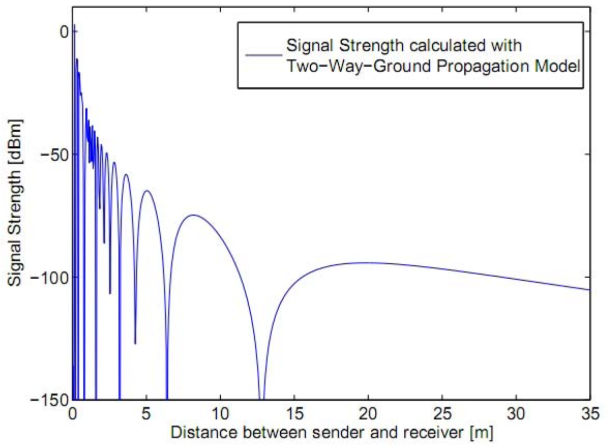


Fig. 6. Signal Strength calculated with Two-Way-Ground Propagation Model

Figure 7c shows the three growing circles. As one can easily see, there exists a small triangle in the middle where all three circles intersect. We have introduced a specific variable which is responsible for the width, each circle is increased at each iterative step. If this value is to big, the computation will lead to a large triangle. If it is to small, there will be a large number of iterations necessary.

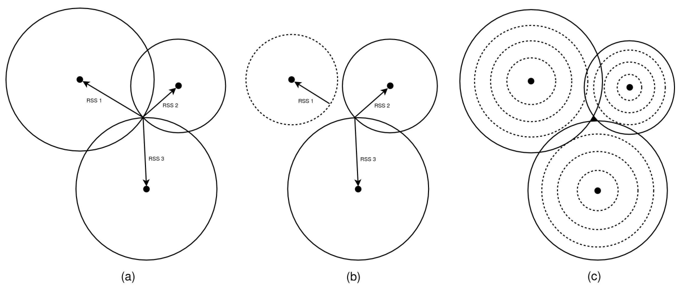


Fig. 7. (a) Optimal trilateration (b) With one noisy signal (c) Interactive trilateration

Within each iteration, we get 6 different intersection points as shown in figure 5. Without noise, three of this points would be at the same position. Unfortunately, this is not possible in the system as each intersection has a correct and an erroneous point. We define the result of each intersection with the variables x, y and z and their possible values as 0 for the erroneous and 1 for the correct position. Now, the possible combination can be calculated with formula 6.

$$Possible combinations = 2^{Number of intersections} \tag{6}$$

With each combination, we draw a triangle and obtain seven wrong and one correct arrangement. To find the correct triangle, we compare the distance between the points of the different intersections. Looking at figure 5, we can see that the distance between x_1 and y_1 is smaller than the distance between x_2 and y_1. Therefore, we calculate:

$$abs(x_1 - y_1), abs(x_1 - y_2), abs(x_2 - y_1), abs(x_2 - y_2)$$

The smallest distance of these four calculations returns the two correct points. In the given example, these points are x_1 and y_1. The same procedure is then repeated for the z values. To achieve a certain level of redundancy, this is also done with the x and y values. The area in the middle is no triangle with straight lines. To use a standard formula to find the balance point and minimize the failure, we try minimize the area in the center. Therefore, the iteration will repeated until a certain threshold is reached.

In our prototype, the results are displayed on the server (which can also be installed on one of the drones) via a web interface. For the web server, the `lighttpd` package was used, because it uses less than 100 kB of memory on the router. The used working version is `1.4.11-1` and it is enhanced with the `lighttpd-mod-cgi` module. For the visualization, we use a small PHP function which allows the user to insert a target client's MAC address to determine its location. This address is than written to a text file which is used by the server script. The script sends this address to the different drones and waits until enough signal strength values are received. After the trilateration of this data, the server stores the position information into another text file. The PHP processor then takes the information and creates a image file which contains the position of the client in relation to a specified layout of the building in which the wireless routers reside. Further on, this information can be used to decide if the location of the client is within the required area in order to get access to specific services.

III. Results

In figure 8 the resulting relation between the measured received signal strength and the calculated distance for one specific indoor configuration is outlined. We have done several measurements with the client at different locations. The resulting relation between signal strength and distance is exponential as one can easily see from the figure, a fact that is not surprising. Using this relation we can calculate the position of the client in relation to the known positions of the drones.

A. Hardware and Tools

There are two different devices used in our test environment. The first one are the drones, which collect the signal values at different locations. The second one is a more powerful device (i.e. a server), which combines the collected information and calculates the position. The

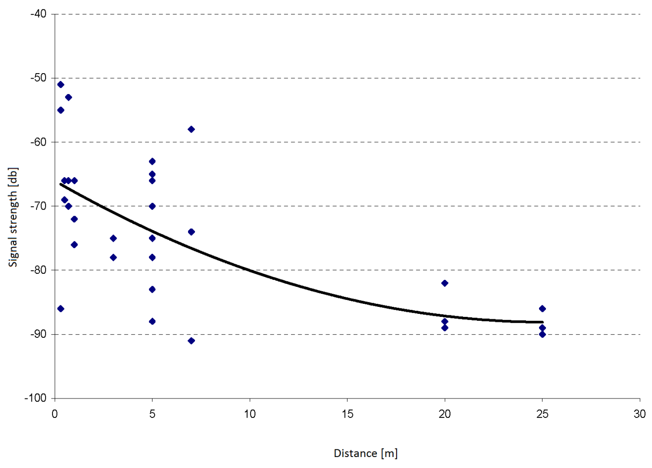


Fig. 8. Measurements of a wireless device with the framework

Fig. 9. Linksys WRT54GL

presentation of the results is handled by a web server able to execute PHP scripts.

For the drones three Linksys WRT54GL v1.1 have been used which are shown in figure 9. The total memory of each router is 4 MB NAND flash memory on a single chip and 16 MB of RAM memory. The CPU runs at 200 MHz. These devices are used to collect the RSS values at different locations. The server runs on an ASUS WL-500G Premium Router. This device operates with a 8 MB Flash storage, 32 MB of RAM memory and runs a CPU with 266 MHz. The router collects the different values from the drones and calculates the position. Additionally, it runs a web server to show the position in a web interface.

The standard firmware of the router does not support the necessary functionality needed for the application. Therefore, an `OpenWRT` OS is installed. Because we need additional software on each of the Linksys Router, we used the OpenWrt White Russian 0.9 Micro Firmware. The software can be found on the OpenWRT web site. There are different versions for the ASUS and the Linksys routers available.

The used tools are `lighttpd` as web server, `lipcap`, `wl` and the `tcpdump` for the signal gathering and the `microperl` package to execute the source code.

IV. Conclusion

In this paper we propose a position determination framework for wireless networks which can be used with standard state-of-the art wireless routers. Contrary to other approaches, our framework does not require any active collaboration of the client. In these other methods, the client is usually performing the data collection of the signal strength values and is transmitting them to a centralized server. An attacking clients is obviously able to alternate the values and pretend to be at a different location. Our proposed approach is measuring the RSS values directly at the network infrastructure without any interaction of the client. The clear advantage is that the client cannot easily fake the values. As already mentioned, in the current state of development, our approach lacks of accuracy due to the effects of signal propagation. But it is possible to improve the accuracy using statistical methods and artificial intelligence mechanisms like self-organizing maps or neural networks.

V. Acknowledgements

The work reported in this article was supported by the European Commission through project SECRICOM, FP7, contract no. FP7-SEC-218123.

References

[1] L. F. M. de Moraes and B. A. A. Nunes, "Calibration-free wlan location system based on dynamic mapping of signal strength," New York, NY, USA, pp. 92–99, 2006.

[2] A. Smailagic and D. Kogan, "Location sensing and privacy in a context-aware computing environment," *Wireless Communications, IEEE*, vol. 9, no. 5, pp. 10–17, Oct. 2002.

[3] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *IEEE INFOCOM*, Mar 2000, pp. 2:775–784.

[4] A. Taheri, A. Singh, and E. Agu, "Location fingerprinting on infrastructure 802.11 wireless local area networks location fingerprinting on infrastructure 802.11 wireless local area networks," in *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks.* Washington, DC, USA: IEEE Computer Society, 2004, pp. 676–683.

[5] M. Youssef and A. Agrawala, "Small-scale compensation for wlan location determination systems," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 3, March 2003, pp. 1974–1978 vol.3.

[6] R. A. Malaney, "Securing wi&#45;fi networks with position verification&#58; extended version," *Int. J. Secur. Netw.*, vol. 2, no. 1/2, pp. 27–36, 2007.

[7] A. Peres, R. F. Weber, P. A. R. Torres, and R. D. Vecchia, "Ieee 802.11 wireless location and network security mechanism through fingerprint, triangulation and dynamic obstacle identification," in *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing.* New York, NY, USA: ACM, 2009, pp. 1459–1463.

[8] P. Appel and P. Ebinger, "Entfernungsschätzungen basierend auf funksignalstärkemessungen für die angriffserkennung in manets," in *D-A-CH Security 2008. Proceedings : Bestandsaufnahme, Konzepte, Anwendungen, Perspektiven*, 2008, pp. 249–261.