

# OWLPS: A Self-calibrated Fingerprint-Based Wi-Fi Positioning System

Matteo Cypriani, Philippe Canalda, and François Spies

FEMTO-ST — UMR CNRS 6174,  
DISC Department,  
University of Franche-Comté, France  
`{surname.name}@univ-fcomte.fr`  
<http://www.femto-st.fr/>

**Abstract.** Owl Positioning System (OwlPS) is an indoor positioning system based on the IEEE 802.11 radio network (Wi-Fi). Since 2004, our team OMNI develops and experiments various techniques (both from the literature and from our own work) for indoor and outdoor positioning. We mainly exploit signal strength fingerprinting and indoor propagation models, helped by information such as the building's map, the mobile's path, etc. The latest version of the system (v1.2) includes a self-calibration mechanism, that avoids the time-consuming manual fingerprinting phase and allows taking into account dynamically the changes of the environment (human, climatic, etc.) when computing the location of the mobile terminals.

## 1 Introduction

In order to have good positioning accuracy indoors, one can develop a system using line-of-sight methods such as light (visible or not: infrared, laser...) or ultrasound. However, if there are obstacles between the sender and the receiver, it is required to use a medium that is able to go through them, such as radio waves. All the techniques based on radio signals have a relatively bad accuracy in heterogeneous environments such as buildings, because of the many obstacles that modify the waves' characteristics due to physical phenomena such as reflection, absorption, refraction and diffraction. Much work has been conducted in the last few years to define a radio-based positioning technique which would accurately estimate the position even in such heterogeneous environments.

Two techniques are commonly used to build an indoor positioning system based on radio waves:

- propagation models, sometimes adapted for indoor environments [1,2], along with geometrical methods such as trilateration or multilateration;
- fingerprint of the signal strength (SS) in the deployment area [3].

The propagation model-based systems are very fast to deploy but the positioning accuracy is weak. The fingerprinting method is quite slow to deploy, because

it requires one to build a cartography of the SS in the deployment area before localising mobile terminals; however, the positioning accuracy can be quite good, depending on the building complexity, the fingerprint meshing and the positioning function.

We chose to work mainly with fingerprinting-based methods, which give good results. The weak point of these approaches is the duration of the repository construction, but it seems also to be the easiest task to automate; we propose such an automation method in Section 2.4.

This paper is organised as follows: Section 2 presents the OwlPS platform in detail, then we briefly explain the deployment made at the CIAMI Living Lab during the EvAAL contest in Section 3; in Section 4 we present and discuss the results obtained after the competition.

## 2 The OwlPS Platform

Owl Positioning System is a Wi-Fi-based, infrastructure-centred<sup>1</sup> positioning system developed at the University of Franche-Comté, which implements several positioning algorithms and techniques. We first present its architecture and its deployment process, then the positioning algorithms implemented, and finally an explanation of the self-calibration mechanism.

### 2.1 Architecture

The OwlPS architecture, summarised in Fig. 1, is composed of several elements:

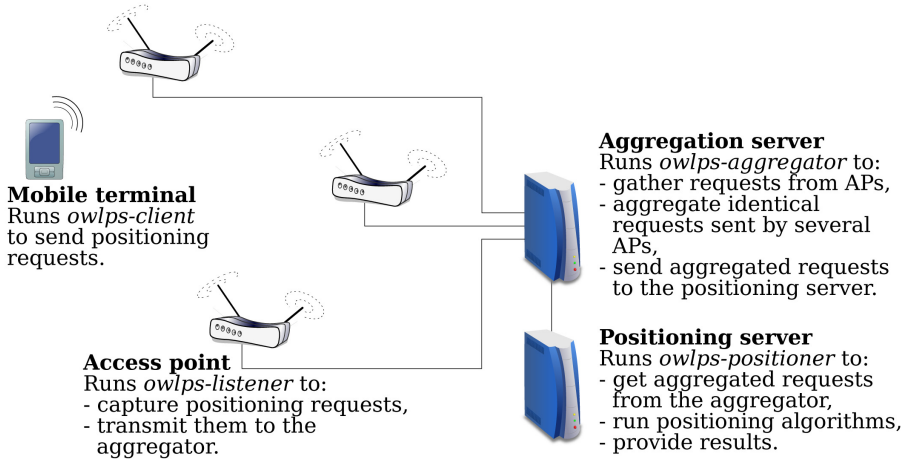
- **Mobile terminals**, such as laptops, PDAs, cell phones, hand-held game consoles, etc., which are equipped with Wi-Fi cards. These run the *owlps-client* software.
- **Access points (APs)**, which capture the frames of the Wi-Fi network in order to receive any positioning request transmitted by the mobiles. These run the *owlps-listener* software, which uses the *pcap* library to capture the IEEE 802.11 frames. The SS values are extracted from the *Radiotap* [4] header of each frame, therefore the network interface’s driver must support *Radiotap*<sup>2</sup>. It is possible to have as many APs as desired: as long as they are only listening to the radio network, they do not cause any interference.
- **The aggregation server**, to which the APs forward the captured positioning requests; its task is to gather and format these requests. It runs the *owlps-aggregator* software.

---

<sup>1</sup> In an infrastructure-centred architecture, the elements of the infrastructure do the measurements and compute the positions of the mobile terminals, as opposed to a mobile-centred architecture in which the mobile terminals measure and compute their own positions.

<sup>2</sup> On Linux, only a few drivers such as *ipw2200* or *MadWifi* used to support *Radiotap*, but nowadays, thanks to the new *mac80211* infrastructure of the Linux kernel, more and more drivers are *Radiotap*-enabled [5].

- **The positioning server** (or computation server), which computes the position of each mobile from the information forwarded by the aggregation server, thanks to the *owlps-positioner* software.



**Fig. 1.** Hardware and software architecture of OwlPS

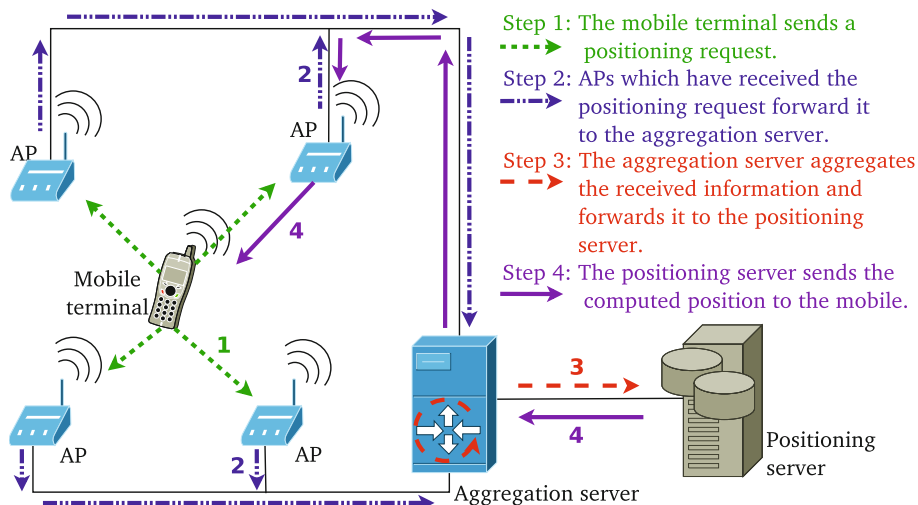
All the modules are implemented in C, except *owlps-positioner*, which is written in C++. The system is designed and tested on GNU/Linux-based platforms<sup>3</sup>. The *owlps-client* module is not mandatory, it can be replaced (for instance on Java-based cell phones) by any software able to send a UDP packet following the adequate data format.

Of course, a single machine can run several software modules; in general the aggregation and computation modules are installed on the same host. Except for *owlps-positioner*, the memory footprint of the modules is low enough to run on most embedded hardware<sup>4</sup>. The hardware requirements of *owlps-positioner* depend on the number of mobiles and APs, and on the positioning algorithm selected. With a low workload (e.g. one mobile and six APs), it can run on a low-end PC without difficulty.

With a high number of capture APs, it is possible to have more than one aggregation server, each group of APs being configured to send the captured

<sup>3</sup> The *owlps-client* and *owlps-aggregator* modules also build on BSD platforms. Some parts of the network-related code of *owlps-listener* are Linux-specific, so it would require a few adaptations to work on another operating system. *owlps-positioner* builds on any UNIX-like platform with GCC 4.4 or later and the Boost libraries.

<sup>4</sup> Around 27MB of virtual size (1MB of resident memory) for *owlps-aggregator* and 2.7MB (1.2MB resident) for *owlps-listener*.



**Fig. 2.** Four-step process of the mobile's position resolution

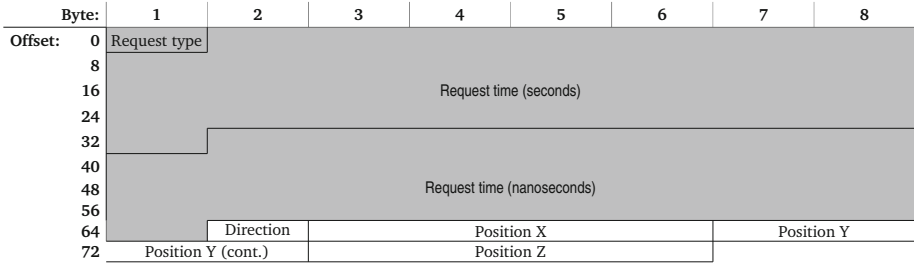
positioning requests to a given aggregation server. However, it is currently not possible to have more than one positioning server in a single deployment area.

Fig. 2 summarises the four steps of the mobile's position resolution:

1. The mobile submits a positioning request to the infrastructure. This request consists of a group of identical UDP packets containing the local time on the mobile terminal; when used to calibrate the system, it also contains the current coordinates of the mobile. Fig. 3 describes the binary format of the request packet.
2. Each AP capturing the positioning request extracts the corresponding SS. Then it transmits to the aggregation server a UDP packet containing the received mobile information, the received SS, the timestamp of reception on the AP, and the mobile and AP MAC addresses.
3. The aggregation server receives the positioning requests forwarded by the APs. It gathers those corresponding to the same couple {mobile MAC address, request timestamp} and forwards them to the positioning server.
4. The positioning server analyses the information received from the aggregation server and computes the mobile's position; the result is then sent to the mobile, or processed in another way. Fig. 4 shows the various ways the positioning server can provide the computed position to a user or another software module.

## 2.2 Deployment

The deployment of the system is pretty straightforward, and includes the following steps:



**Fig. 3.** Binary format of a positioning/calibration request’s packet sent by the mobile, as of OwlPS 1.2. The grey fields are always present; they correspond to a positioning request (65 bytes). If the packet contains all the fields, it is a calibration request (78 bytes). All the fields must be encoded in the network’s endianness (big endian).

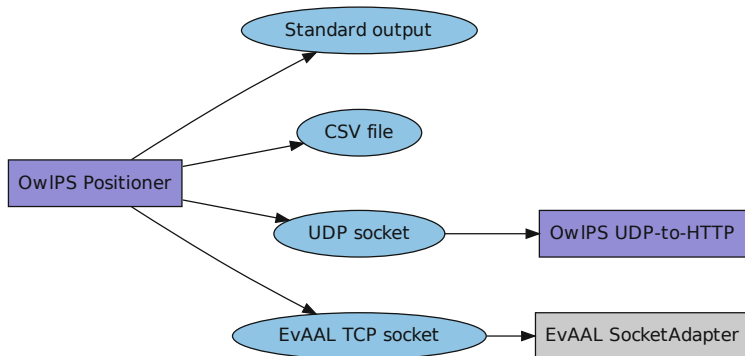
1. Deployment of the APs into the area. They must be able to communicate with the aggregation server, either through a wired or wireless network.
2. Description of the hardware characteristics in the configuration files of the positioning server: antenna gain, transmitted power, operating radio frequency, coordinates of the fixed elements.
3. Description of the size and topology of the deployment area – optional, depending on the algorithm used.
4. Manual off-line calibration (fingerprinting) – only if self-calibration is not to be used (see Section 2.4).

### 2.3 Implemented Positioning Algorithms

When running the system, one must also choose at least one positioning algorithm amongst those implemented in the positioning server. The algorithms implemented in OwlPS 0.8 were described briefly and compared in [6]. Here is a quick summary of the algorithms implemented in OwlPS 1.2:

- Nearest-neighbour in Signal Strength (NSS), based on RADAR [3], which is a simple cartography-based algorithm.
- Trilateration using the propagation formula proposed by Interlink Networks in [1].
- Trilateration using the FBCM [2,7] (*Friis-Based Calibrated Model*), which adapts the propagation formula to better match the deployment area’s characteristics, thanks to a minimal calibration.
- Basic FRBHM [8,7,9] (*FBCM and Reference-Based Hybrid Model*), that is a combination of the NSS and the FBCM which allows to adapt dynamically the propagation formula to the characteristics of the room where the mobile terminal is supposed to be.

Since recent work has been mostly centred on the self-calibration, the support for the Viterbi-enabled algorithms (NSS with Viterbi-like [10], Discrete and Continuous FRBHM [7,9]) was dropped as of OwlPS 1.0.



**Fig. 4.** Output formats supported by OwlPS Positioner v1.2, including the output developed for the EvAAL contest to connect with the `SocketAdapter` program provided by the organisers. The OwlPS UDP-to-HTTP module is a minimalist HTTP server that takes as input the results provided by the positioning server on a UDP socket, and provides the last position of each mobile request; we use this module to develop a Javascript monitoring program based on Google Maps.

## 2.4 Self-calibration

The OwlPS 1.2 release implements a self-calibration (or auto-calibration) mechanism that allows the system to be operational within a few minutes after its deployment. Since the self-calibration is a continuous process, it also guarantees that the system is aware of the modifications that occur in the radio environment.

For instance, if the number of people present in the building changes, if those people move within the building, if furniture is moved, if the weather changes, etc., the system will take into account the changes in order to maintain accuracy. On the other hand, with static calibration, the positioning error can raise dramatically if the environment changes. Moreover, the auto-calibration process is quick enough to allow the system to be aware of the short term modifications of the environment, for example a door that is opened or closed, someone walking through a corridor, etc.

When self-calibration is activated, the aggregation server sends the APs regular round robin orders to transmit an auto-calibration request. When receiving such an order, an AP transmits a positioning request, as if it were a mobile terminal, which will be intercepted by the other APs. The request is then processed the usual way: it is transmitted by the APs to the aggregation server and, once aggregated, to the positioning server.

The positioning server is then able to build a matrix  $S$  of the SS received by each AP of index  $j$  from each AP of index  $i$ . If we note  $s_{Tx,Rx}$  the strength of a signal from a transmitter  $Tx$  to a receiver  $Rx$ , and  $n$  the number of deployed APs, the matrix  $S$  is defined as:

$$\forall i, j \in \mathbb{N}, i \neq j, i \leq n, j \leq n : S_{i,j} = s_{AP_i, AP_j} \quad (1)$$

An example of such a matrix is given in Fig. 5, for four APs.

Tx Rx	<b>AP<sub>A</sub></b>	<b>AP<sub>B</sub></b>	<b>AP<sub>C</sub></b>	<b>AP<sub>D</sub></b>
<b>AP<sub>A</sub></b>	-	-21	-60	-51
<b>AP<sub>B</sub></b>	-23	-	-52	-73
<b>AP<sub>C</sub></b>	-64	-55	-	-17
<b>AP<sub>D</sub></b>	-49	-70	-19	-

**Fig. 5.** Sample matrix of the SS received by each AP from each other, in dBm. As shown, the signal is not necessarily symmetrical, i.e. the SS received by  $AP_A$  from  $AP_B$  can be different than the SS received by  $AP_B$  from  $AP_A$ .

From the APs' matrix  $S$ , and the APs' description file (containing their coordinates and hardware characteristics, see Section 2.2), the system builds a geographical matrix, called  $G$ . Each element  $(i, j)$  of this matrix represents a spatial coordinate  $(x, y)$  of the deployment area and contains an extrapolation of the real SS values of  $S$ , that is the SS received from a virtual mobile terminal  $M$  located in  $(x, y)$  by each of the APs:





$$G_{i,j} = \{s_{M,AP_1}; s_{M,AP_2}; \dots; s_{M,AP_n}\} \quad (2)$$

Note that  $x$  and  $y$  are the geographical coordinates (real numbers) of the virtual mobile  $M$ , but this position is stored in the matrix as the element  $(i, j)$  ( $i$  and  $j$  are integers). Fig. 6 gives an example of a geographical matrix.

As stated above, to create the element  $(i, j)$  of the matrix  $G$ , the positioning server will generate one SS per AP. To generate the SS received by  $AP_A$  from  $M$ , it first chooses, amongst all the other APs, the two that have the more acute angles with the coordinates of  $AP_A$  and  $M$ . In Fig. 7, the coordinates of the APs and virtual mobile are the following:

- $AP_A$ : (1, 10),
- $AP_B$ : (7, 10),
- $AP_C$ : (1, 1),
- $AP_D$ : (7, 1),
- $M$ : (6, 8).

Thus, the angles formed by  $M$ ,  $AP_A$ , and the other APs are the following (in the angle notations, we shorten  $AP_X$  to  $X$ ):

10		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD		
9		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	
8		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	
7		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	
6		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	
5		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	
4		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	
3		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	
2		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	
1		SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD	SSA SSB SSC SSD		
		1	2	3	4	5	6	7

**Fig. 6.** Sample geographical matrix of the deployment area, with four APs in the corners. Each element of the matrix represents a geographical coordinate in the area. The physical distance between two elements, in horizontal and vertical axis, is set by the administrator when the system is deployed, depending of the area's topology. For the sake of clarity, we will consider here that two elements are separated by a distance of 1 m both horizontally and vertically, but this distance can be set independently in each axis.

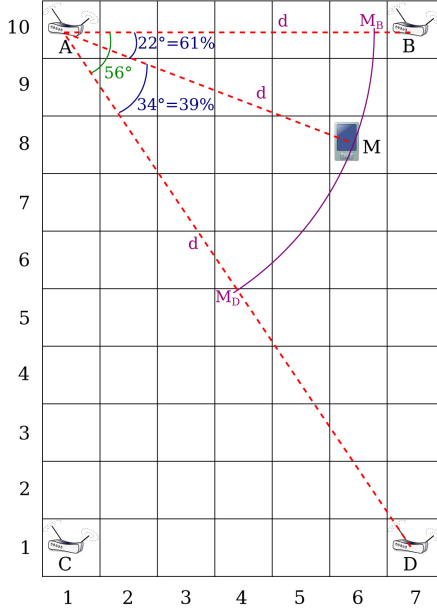
- $\widehat{MAB} \simeq 22^\circ$
- $\widehat{MAC} \simeq 68^\circ$
- $\widehat{MAD} \simeq 34^\circ$

Here we have  $\widehat{MAB} < \widehat{MAD} < \widehat{MAC}$ , therefore the two nearest APs in angle are  $AP_B$  and  $AP_D$ .

A weight is then attributed to the two selected APs: the one with the most acute angle will receive a higher weighting in the computation of the SS. The reference angle  $\widehat{BAD}$  is the sum of  $\widehat{MAB}$  and  $\widehat{MAD}$ , which equals  $56^\circ$ .  $\widehat{MAD}$  ( $34^\circ$ ) is approximately 61% of  $56^\circ$ , therefore we attribute a weight  $W_B = 61\%$  to  $AP_B$ , and  $AP_D$  is given a weight  $W_D = 100 - 61 = 39\%$ .

Once the two reference APs are selected and weighted, the SS can be computed. The principle is to evaluate the quality of the link between the considered





**Fig. 7.** Selection and weighting of the reference APs to compute the SS received by  $AP_A$  from the virtual mobile  $M$

AP ( $AP_A$  in our example) and the reference APs ( $AP_B$  and  $AP_D$ ), and to use the weights to estimate the quality of the link between  $AP_A$  and  $M$ . To evaluate the link quality, the Friis transmission equation is used. The base equation in dBm is shown in Equation 3 (the losses at the receiver and transmitter are ignored); we then demonstrate that we can write the equation in such a way that the Friis index  $N$  is computed in function of the other parameters (Equation 5).

$$Pr = Pt + Gt + Gr + 20 \log \lambda - 20 \log (4\pi) - 10 N \log d \quad (3)$$

$$10 N \log d = Pt + Gt + Gr - Pr + 20 \log \lambda - 20 \log (4\pi) \quad (4)$$

$$N = \frac{Pt + Gt + Gr - Pr + 20 \log \lambda - 20 \log (4\pi)}{10 \log d} \quad (5)$$

Where:

- $Pr$ : power gathered on the reception antenna (dBm);
- $Pt$ : power sent to the transmission antenna (dBm);
- $Gr, Gt$ : reception and transmission antennas' gains (dBi);
- $d$ : distance travelled (metres);
- $\lambda$ : wavelength (metres);
- $N$ : Friis index (also called “path loss exponent”).

Let  $d$  the distance between the considered AP ( $AP_A$ ) and  $M$ . Let  $M_X$  a virtual mobile located at a distance  $d$  from  $AP_A$ , in the direction of  $AP_X$ . We use Equation 5 to compute the Friis index  $N_B$ , respectively  $N_D$ , for the link  $AP_B \rightarrow AP_A$ , respectively  $AP_D \rightarrow AP_A$ . As stated in the section 2.2, we already know, from the file describing the APs' hardware, the parameters  $Pt$ ,  $Gr$ ,  $Gt$ , and  $\lambda^5$ ; the distance  $d$  between the two APs is computed thanks to their coordinates, declared in the same configuration file, and  $Pr$  is the SS received by  $AP_A$  from the other AP, read from the matrix  $S$ .

Then, thanks to Equation 3, we compute the SS that would be received by  $AP_A$  from  $M_B$ , respectively  $M_D$ , using the Friis index  $N_B$ , respectively  $N_D$ . The final SS for  $M$  as received by  $AP_A$  is the mean of the SS of  $M_B$  and  $M_D$ , weighted according to the previously computed weights of  $AP_B$  ( $W_B$ ) and  $AP_D$  ( $W_D$ ):

$$SS_M = \frac{SS_{M_B} \times W_B}{100} + \frac{SS_{M_D} \times W_D}{100} \quad (6)$$

The same process is repeated to generate the SS for each other AP at this position (in our example:  $AP_B$ ,  $AP_C$  and  $AP_D$ ). Once a SS has been generated for each AP, the coordinates of  $M$  are updated and the next element of the matrix  $G$  is created. In the end,  $G$  is filled with generated ‘‘calibration measurements’’, that can be used by algorithms like the NSS, as a real (manual) calibration would be used. Fig. 8 summarises the auto-calibration process.

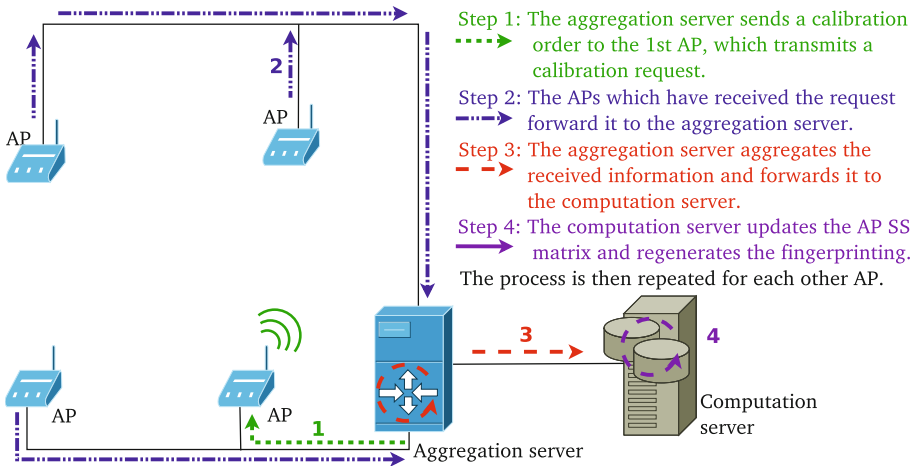
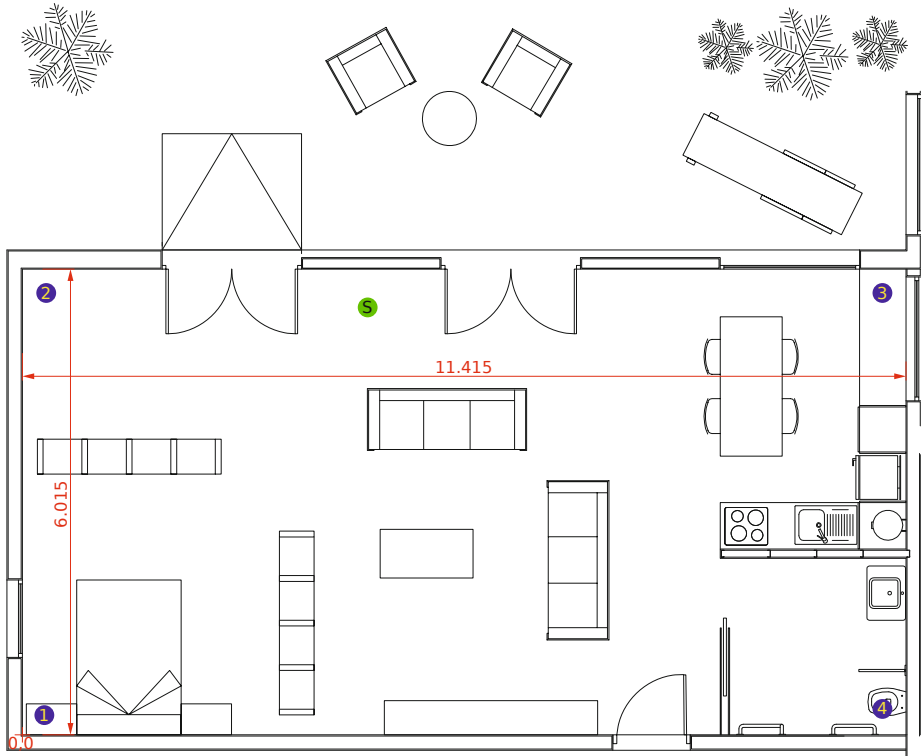


Fig. 8. Summary of the auto-calibration process

<sup>5</sup> The wavelength  $\lambda$  is computed from the propagation speed of the radio signal (we use the speed of light in vacuum,  $c$ ), and its frequency  $f$ :  $\lambda = \frac{c}{f}$ .

### 3 Deployment at the CIAmI Living Lab

We describe in this section the deployment of OwlPS at the CIAmI Living Lab in Valencia, during the EvAAL competition. We deployed four Wi-Fi APs (Fonera 2.0<sup>6</sup> running the embedded Linux distribution *OpenWrt*), one in each corner of the evaluation area (see Fig. 9). The APs include a *Radiotap*-enabled Atheros Wi-Fi chipset, configured with the *MadWifi* tools [11].



**Fig. 9.** Plan of the CIAmI Living Lab, with the four Foneras (blue numbered circles) deployed in each corner of the area: (1) 0.30,0.30; (2) 0.30,5.70; (3) 11.11,5.70; (4) 11.11,0.30. The server (positioning and aggregation) is represented by the green “S” circle (approximate position: 4.50,5.50).

The aggregation and positioning software modules are both installed on a Lenovo Thinkpad X200<sup>7</sup> running Debian GNU/Linux. The `SocketAdapter` program provided by the organisers is used to send the results to the EvAAL evaluation software. The located device is another Fonera, powered by a small battery.

<sup>6</sup> Atheros AR2315 running at 180 MHz, 32 MB RAM, 8 MB storage.

<sup>7</sup> Intel Core 2 Duo P8400 running at 2.26 GHz, 2 GB RAM.

All the modules communicate through an IEEE 802.11 ad-hoc network; this is possible because the Wi-Fi interfaces of the APs support running several modes (here ad-hoc and monitor) simultaneously. If the Wi-Fi interfaces were single-mode only, we would have needed to use the wired (Ethernet) network, or to add a second Wi-Fi interface to the APs<sup>8</sup>.

The self-calibration is activated, with one auto-calibration request every 320 ms, i.e. each AP sends a request every 1280 ms. The positioning server computes the results using the NSS algorithm.

## 4 Competition Results' Discussion

The scores obtained by OwlPS during the EvAAL 2011 contest are given in Table 1.

**Table 1.** Scores obtained by OwlPS and average of the scores of the six competitors. As a reminder, the EvAAL scores range from 0 to 10.

Criterion	OwlPS	Average
Accuracy	1.3653	4.2311
Availability	9.4337	6.5147
Installation complexity	8.4733	5.8822
User acceptance	6.5	6.2083
Integrability in AAL	1	5.5833
Final score	4.85	5.0067

Undoubtedly, the main strengths of OwlPS are its high positioning rate and its quick deployment procedure, hence the scores in the *availability* and *installation complexity* criteria. There is not much to say about the availability of the system; the maximum score was not reached most likely because of a few packet losses, or maybe some desynchronisations between OwlPS and the evaluation system.

The initial deployment of the system in the living lab took only 7 minutes with one single operator, which would have allowed a score of 10 in *installation complexity*. Unfortunately, a software bug appeared in the first run of the evaluation, causing the results provided by the positioning server to be completely aberrant. Patching the software to fix the bug took approximately 10 more minutes, which of course were added to the deployment time, for a total of about 17 minutes, therefore the score was lowered.

OwlPS requires the person being localised to carry a Wi-Fi-enabled device. During the competition, this device was a Wi-Fi router with a belt clip, powered by a small battery. In a real-life deployment, a device such as a smartphone would be used, which has been judged quite acceptable by the evaluation committee. The drawback of this system is that it requires several APs to be deployed across

---

<sup>8</sup> The Fonera 2.0 has a USB port, in which we could plug an additional USB Wi-Fi interface.

the user’s house, but this has not been seen as a major drawback. The score of *user acceptance* obtained is just above average.

The two serious flaws of OwlPS are its *accuracy* and its *integrability*. OwlPS is an experimental research project, and is mainly developed by only one programmer; since it is not a commercial product, only what is strictly necessary from a research support point of view is developed. Therefore, it does not conform with most of the criteria defined by the organisers for the *integrability in AAL*.

**Monitoring.** There is no pre-made monitoring tool. It would be possible (and actually quite easy) to use well-known monitoring tools such as Nagios, deploying SNMP agents on the APs and servers. However, it is true that it is not a turnkey solution, and a monitoring configuration could be provided to help the system administrator.

**Documentation.** The OwlPS source code is essentially self-documented; in the positioning server code, Doxygen-style comments are used to allow the generation of a documentation in a more readable format (HTML,  $\LaTeX$ , man pages, etc.). A rather complete user documentation exists for OwlPS 0.8 [12] (in French), but it is not totally up-to-date with OwlPS 1.2.

**External library.** There is no software library allowing one to easily receive and interpret the positioning results in an external program. However, the positioning server can provide the results in several ways (as shown in Fig. 4), and it is quite easy to add new output formats; as an example, the code of the `OutputTCPSocketEvaal` C++ class added in the positioning server to participate in the contest is only 130 (real) lines long. Moreover, the CSV format used in the CSV and UDP outputs can be easily parsed with any programming language. We plan to add the support of XML for the output, so that it would be even easier to implement a parser, thanks to an XML schema.

**Open Source.** Finally, the OwlPS code is not public; it was planned to release it under a free software license before the summer of 2011 (i.e. before the competition), but the process is currently on hold since we are engaged in a technology transfer program.

The *accuracy* score is computed by averaging the scores of the two phases of the evaluation.

**Phase 1:** the positioning system has to guess the area of interest (AOI) where the user is standing; OwlPS guessed correctly in 27.31% of the cases, which gives a score of 2.731 (see Table 2).

**Phase 2:** the user goes through a “random” path, and the system outputs an estimation of its position in real time, the evaluation criterion being the euclidean distance error of the estimated position; for OwlPS, the global 75th percentile of error is above 4 metres, which gives a score of 0 (see Table 3).

**Table 2.** Accuracy results for the phase 1 (areas of interest). Number of times the AOI guessed by OwlPS Positioner matches or does not match the AOI where the user is.

Matches	Misses	Total
222	591	813
27.31%	72.69%	100%

**Table 3.** Accuracy results for the phase 2 (random path). Mean error (in metres), standard deviation and 75th percentile for the three routes of the second test of the phase 2. For each route, the first two column show the error in X and Y, and the third column the euclidean distance in the plan. The “Global” columns are for all the values in the three routes taken together.

	Route 1			Route 2			Route 3			Global		
	X err.	Y err.	Err.	X err.	Y err.	Err.	X err.	Y err.	Err.	X err.	Y err.	Err.
Mean	2.23	2.58	3.63	3.61	1.03	3.94	2.56	1.85	3.43	2.95	1.65	3.70
Std.	1.58	1.29	1.62	2.22	1.13	2.19	1.78	1.37	1.80	2.04	1.39	1.96
75th	2.63	3.84	4.63	5.22	1.05	5.22	3.71	3.41	4.83	3.76	2.98	4.89

If the accuracy of OwlPS is far from perfect, we can say that such results make the system pretty usable in most indoor positioning scenarios. With a mean error below 4 metres, and a 75th percentile of error below 5 metres, the room in which the user is can be determined without error in most cases. In addition, the standard deviation is relatively low, with a maximum error, all three routes taken together, of 9.44 metres. It is true however that this precision is not sufficient in scenarios such as activity recognition, in which the system needs to estimate the position with a precision of at least one metre, to be able to determine in which area of a room the user is and what he is likely to do; an estimation of the user’s orientation would also help in such scenarios.

## 5 Conclusion

In this paper, we presented the main features of OwlPS, our Wi-Fi-based, infrastructure-centred, indoor positioning system. Its most interesting features include a scalable and flexible architecture, the use of standard and low-cost hardware, and above all a fast deployment and a low cost of maintenance thanks to its self-calibration mechanism.

The positioning software module is designed to be modular, so it is easy to implement additional positioning techniques, output formats, etc. It is also able to generate results for several positioning algorithms from the same input data, so it is really simple to compare objectively the results.

With a final score of 4.85, OwlPS is just below the average of the EvAAL competitors. There is a huge progression margin in the *integrability* and *accuracy* criteria. The former would require some work to make the system more turnkey: new output formats, software packages to ease the installation, comprehensive

documentation to help the administrator during configuration, tools to monitor the infrastructure devices, etc.

We identified that the accuracy problem comes from the similarity algorithm of the nearest neighbour function. Indeed, during the positioning phase the positioning server chooses in the SS cartography the point that is the most similar to the current SS measurement from the mobile terminal. But it often happens that several points of the cartography are considered as similar, even though their geographical coordinates are far from each other. We are currently working on improving this function, by introducing new similarity algorithms based on probabilities.

However, it is to be noted that the self-calibration mechanism fulfils its goal, which is to allow a very quick deployment of the system without degrading significantly the accuracy and the positioning rate. This is highlighted by the good scores obtained in the *installation complexity* and *availability* criteria.

In the current and former deployments of OwlPS, we set up the APs so that they form a convex polygon, in which the mobile terminals are supposed to be most of the time. This is an intuitive way to deploy, but it is not proven to be the best choice. Ongoing work include optimisation of both the Wi-Fi coverage offered to the mobile terminals and the positioning accuracy.

Future work will also bear scalability improvements, for example by allowing several positioning servers to be used in a single deployment, or limiting the network traffic.

## References

1. Interlink Networks, Inc.: A practical approach to identifying and tracking unauthorized 802.11 cards and access points. Technical report (2002)
2. Lassabe, F., Baala, O., Canalda, P., Chatonnay, P., Spies, F.: A Friis-based calibrated model for WiFi terminals positioning. In: Proceedings of IEEE Int. Symp. on a World of Wireless, Mobile and Multimedia Networks, Taormina, Italy, pp. 382–387 (June 2005)
3. Bahl, P., Padmanabhan, V.N.: RADAR: An in-building RF-based user location and tracking system. In: INFOCOM (2), pp. 775–784 (2000)
4. Radiotap website, <http://www.radiotap.org/>
5. Radiotap on Linux Wireless website, <http://linuxwireless.org/en/developers/Documentation/radiotap>
6. Cypriani, M., Lassabe, F., Canalda, P., Spies, F.: Open Wireless Positioning System: a Wi-Fi-based indoor positioning system. In: VTC-fall 2009, 70th IEEE Vehicular Technology Conference, Anchorage, Alaska. IEEE Vehicular Technology Society (September 2009)
7. Lassabe, F., Canalda, P., Chatonnay, P., Spies, F.: Indoor Wi-Fi positioning: Techniques and systems. *Annals of Telecommunications* 64(9/10), 651–664 (2009)
8. Lassabe, F., Charlet, D., Canalda, P., Chatonnay, P., Spies, F.: Refining WiFi indoor positioning renders pertinent deploying location-based multimedia guide. In: Procs of IEEE 20th Int. Conf. on Advanced Information Networking and Applications, Vienna, Austria, vol. 2, pp. 126–130 (2006)

9. Cypriani, M., Canalda, P., Lassabe, F., Spies, F.: Wi-Fi-based indoor positioning: Basic techniques, hybrid algorithms and open software platform. In: Mautz, R., Kunz, M., Ingensand, H. (eds.) IPIN 2010, IEEE Int. Conf. on Indoor Positioning and Indoor Navigation, Session WLAN RSS (Signal Strength Based Methods), Fingerprinting, ETH Zurich, Campus Science City (Hoenggerberg), Switzerland, pp. 116–125 (September 2010)
10. Bahl, P., Balachandran, A., Padmanabhan, V.: Enhancements to the RADAR user location and tracking system. Technical report, Microsoft Research (February 2000)
11. MadWifi website, <http://www.madwifi-project.org/>
12. Cypriani, M., Canalda, P., Zirari, S., Lassabe, F., Spies, F.: Open Wireless Positioning System, version 0.8. Technical Report RT2008-02, LIFC - Laboratoire d'Informatique de l'Université de Franche Comté (December 2008)