REIHE INFORMATIK
TR-2007-006

Loc{lib,trace,eva,ana}: Research Tools for 802.11-based Positioning Systems

Thomas King, Hendrik Lemelson, Thomas Haenselmann, and
Wolfgang Effelsberg
University of Mannheim
– Fakultät für Mathematik und Informatik –
Praktische Informatik IV
A5, 6
D-68159 Mannheim, Germany

Loc{lib,trace,eva,ana}: Research Tools for 802.11-based Positioning Systems

Thomas King, Hendik Lemelson, Thomas Haenselmann, Wolfgang Effelsberg
{king,lemelson,haenselmann,effelsberg}@informatik.uni-mannheim.de
Department of Computer Science
University of Mannheim

Abstract

802.11-based positioning systems are a hot topic in research. However, no standardized set of tools has been established to facilitate the research process. In this paper, we contribute our research tools to the community. The benefit for the community is considerable: *(1)* Standardized tools reduce the amount of work each researcher has to spend to build software to collect signal strength samples and process this data. *(2)* The confidence in the correctness of the tools increases because everybody is encouraged to submit bug-fixes. *(3)* A unified evaluation process makes results mutually comparable. *(4)* We hope other researchers contribute to our tools.

1 Introduction

During recent years we have seen considerable improvements in down-sizing computer hardware and in increasing the capacity of rechargeable batteries, as well as the advent of wireless networks for the mass markets. These technologies allow manufacturers to build mobile devices that have a similar performance as desktop computers had several years ago. The benefit of mobile devices can be leveraged by so-called *location-based services*: Applications that act differently depending on the location of the user, or, even better, proactively offer location-dependent information to the user. Location-based services are currently a hot topic in research, and are considered to be a promising market.

Nowadays, the *Global Positioning System* (GPS) [8] is the predominant outdoor positioning system. Whereas GPS works well in many outdoor scenarios, it suffers from obstacles such as skyscrapers creating shielded street canyons, or walls and ceilings blocking the radio signals indoors. A large number of research projects conceived novel positioning techniques (e.g., [4], [17], and [21]). However, all these systems either require specialized hardware or show poor positioning accuracy.

Many recent research activities focus on *IEEE* 802.11-based positioning because almost everywhere, especially in occupied areas of developed countries, 802.11 net-

work infrastructure is available for data communication[1]. Universities, offices and many private homes utilize 802.11 networks to get rid of wires. As a reaction to the proliferation of 802.11, almost all modern mobile devices ranging from smartphones to laptops are shipped with built-in 802.11 network interfaces. 802.11 networks are not only used in indoor scenarios; even outdoors, many universities and coffee shops support nomadic users. Furthermore, 802.11 radio waves tend to travel outside the intended area covering adjacent regions as well. For instance, an access point deployed at a private home is often detectable while passing by [6]. Finally, 802.11-based positioning systems show sufficient positioning accuracy to be useful for a wide range of applications.

The most promising 802.11-based positioning systems utilize the so-called *fingerprint* approach [3]. This technique comprises two stages: An offline training phase and an online position determination phase. During the offline phase, the signal strength distributions are collected from access points at pre-defined reference points in the operation area. They are stored in a table together with their physical coordinates. An entry in this data-set is called a fingerprint. During the position determination phase, mobile devices sample the signal strengths of access points in their communication range and search for similar patterns in the fingerprint data-set. The best match is selected, and its physical coordinates are returned as position estimate.

Recent research in the area of 802.11-based positioning systems has mainly focused on algorithms that compute a position estimate (e.g., [3], [5], [7], [13], [15], and [22]). Although, the authors of these papers compare their own work to existing approaches, they do not share their research tools. Best to our knowledge, this paper is the first to present a set of research tools that explicitly focus on research into 802.11-based positioning systems. In this work, we want to provide the tools to the community hoping that they will be widely used in the future. The benefit for the community is considerable: First, researchers can save time because the software to trace signal strength and process data are already in place. Second, the confidence in the correctness of the tools increases because everyone is able to confer the source code and fix bugs. Third, a unified process of collecting data and processing it makes research results mutually comparable. And finally, we hope other researchers extend our set of tools.

The contribution of this paper is not algorithmic or empiric. Instead, we contribute to the community by giving away our set of research tools to initiate the creation of standardized research tools for positioning systems based on 802.11.

The remainder of this paper is organized as follows. The next section (Section 2) discusses the research methodology. Subsequently, Section 3 presents the research tools. In Section 4 related topics are discussed. The related work is presented in Section 5. Finally, directions for future work and a conclusion are provided in Section 6.

2 Research Methodology

The research methodology of this section has been used by many researchers in the area of 802.11-based positioning (e.g., [3], [13], and [22]), but to our knowledge, no publication is available that discusses this research methodology in detail.

[1] http://www.wigle.net

In the following subsection, the concept of emulation is introduced. It is described why emulation is needed and how it works. Subsequently, the research process is described in depth.

2.1 Emulation

Many questions addressed by researchers cannot be solved analytically because they are too complex. In most cases, a simulation of the scenario in question can be used to derive answers. Especially in the network research community, simulations are often carried out to compare different algorithms and to understand how they perform on a large scale. The quality of results obtained from simulations can only be as good as the simulation model applied. This is the reason why it is practically impossible to use simulations for 802.11-based positioning system research.

Radio propagation models are in general not precise enough to predict the signal strength of a certain access point at a given position inside an operation area. Especially indoors, radio signals are reflected, diffracted and scattered at objects located around the sender or receiver, or even worse, obstacles exist in the line-of-sight between the sender and the receiver [18]. These signal disturbances cause so-called multi-path signal propagation, meaning that the signal takes different paths to reach the receiver. Typically, the different paths are varying in length, and therefore the same signal arrives multiple times at the receiver at different points in time. The interaction of the different signals causes multi-path fading.

While it is already quite difficult to come up with realistic radio propagation models for static indoor or obstructed outdoor scenarios [18], the bar is raised even higher if movement is involved. However, movement must be considered if we talk about 802.11-based positioning systems in general. The application scenario we usually have in mind is that a moving person or object is interested in its position. Without movement of objects or persons, positioning systems are fairly useless.

To overcome the problem of lacking precise, easy to handle, and easy to compute radio propagation models, the concept of emulation has been widely used in the area of 802.11-based positioning system research. The idea behind emulation is that signal strength samples are taken from a testbed that is integrated into a real-world scenario or recreates the real-world scenario in question. The advantage of this approach is that the signal strength measurements are a result of all the radio propagation phenomena that are difficult to simulate. As almost everything in life, emulation also has its disadvantages as well: As radio propagation is taken as a black box, it might happen that the radio propagation of a certain testbed is a singularity and cannot be taken as general. To alleviate this drawback, researchers often use more than one testbed to proof that their results are generally applicable (e.g., [2] and [3]).

Typically, more signal strength samples than required by a positioning algorithm are taken from the testbed. This approach allows to easily investigate certain scenarios by selecting a subset of the data. Emulation allows to get insights into scenarios that would otherwise take an enormous amount of time to investigate in a real-world experiment.

2.2 Research Process

The process supporting our research methodology can be split into four sub-processes:

- *Definition of the testbed*
- *Data collection*
- *Emulation and comparison*
- *Verification*

It is important to note that the sub-processes are not executed independently. Instead, each sub-process relies on the results of previously performed sub-processes. So, it may become clear during the execution of a certain process that a mistake has been made during the execution of an earlier process. In such a case, a feedback loop is required, and the erroneously executed process as well as all subsequent processes have to be repeated. The research process including the sub-processes and the feedback loops are illustrated in Figure 1. Feedback loops are indicated by dashed arrows. To keep the figure concise only feedback loops between two adjacent sub-processes are illustrated, even that feedback loops span over the entire process.

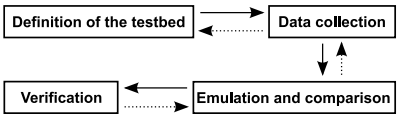


Figure 1: Research process including the sub-processes and the feedback loops.

What follows is a discussion of each sub-process, and the sets of research tools that support each step are named. A detailed description of our research tools is provided in Section 3.

Definition of the Testbed This process is usually the starting point for research in the area of 802.11-based positioning systems. Typically, an operation area is defined inside a building or spread over a campus, depending on the purpose of the research. Sometimes, even larger areas are selected (e.g., [15]). After the testbed is defined, the reference points for the fingerprints have to be selected. For this, an equally-spaced grid is often used, or reference points are selected at locations of interest inside the operation area. To be able to emulate the online determination phase, so-called online points have to be selected as well. Online points are often randomly selected or collected from locations where users are expected to appear.

For this process, a tool to randomly select points inside the operation area is part of *Locana* (see Section 3.5).

Data Collection The reference points and online points are utilized in this step. At each point the signal strength of access points in communication range and if needed additional information from different sensors are sampled for some time. Typically,

more than 100 samples are collected to be able to create various scenarios with the same parameter set.

Loctrace (see Section 3.3) provides an application called *Tracer* to collect signal strength samples at reference points and online points. The data is stored in so-called trace files to be used in the subsequent step.

Emulation and Comparison Trace files created in the previous step are used for emulation and comparison of positioning algorithms. To emulate different scenarios, it should be possible to turn specific objects in the trace file on or off. For instance, to investigate how certain positioning algorithms behave in case of different numbers of access points cover the operation area, it should be possible to virtually switch off and on each access point. Additionally, to be able to compare the positioning algorithm under research with the state-of-the-art algorithms, these algorithms should already be in place.

For this, we provide *Loceva* (see Section 3.4). Loceva consists of two parts: A management part to enable emulation and a positioning algorithm part that contains the state-of-art algorithms.

Verification All results, not only the final results but also the intermediate results, should be verified to make sure mistakes and errors are recognized at an early stage. Each result of each sub-process should be verified for its soundness. Furthermore, to make sure research results apply not only in emulated scenarios but also to real-world environments, a real-world evaluation should be performed as soon as the emulation results are satisfying.

To make verification of intermediate results easy, we implemented a whole bunch of tools. These tools are grouped together in the *Locana* package (see Section 3.5). Additionally, to facilitate real-world experiments we provide *Loclib* (see Section 3.2).

3 Research Tools

This section describes in detail our set of research tools for 802.11-based positioning systems.

3.1 System Overview

In this section, we first provide an abstract overview of the entire set of tools and afterwards discuss their implementation.

3.1.1 Architecture

We start by discussing our software components and continue with the hardware components.

Software Components We created a set of research tools to support each step of our research methodology. The tools are grouped according to their functionality and we created a pool of shared software modules to allow reuse of source code between different groups. Furthermore, this pool allows us to keep the dependencies between the different groups low and manageable. This approach lead us to the following groups of research tools:

- *Loclib*: Loclib is a connector between applications and sensor hardware. Its task is to collect data from the sensor hardware and pre-process it for further usage. On the application side it offers two types of front-ends: The well-known Location API [16] to access position estimates, and so-called handlers that provide access to sensor-specific data (e.g., signal strength values of neighboring access points). On the sensor hardware side, it communicates directly with hardware drivers to access sensor information that would otherwise be hidden. Loclib focuses not only on 802.11, it also contains a GPS part that is able to talk to NMEA-0183-enabled[2] GPS devices as well as a digital compass for obtaining direction and inclination information.

- *Loctrace*: Loctrace mainly consists of one application. This application gathers data offered by Loclib and stores it in a file.

- *Loceva*: Trace files generated by Loctrace are used by Loceva to evaluate different kinds of positioning algorithms. A large number of state-of-the-art positioning algorithms are supported by Loceva. Loceva contains a lot of filters and generators to set up different scenarios and enable emulation.

- *Locana*: Locana visualizes the results computed by Loctrace and Loceva. This helps verifying that the data traced by Loctrace is complete and sound. Intermediate results of Loceva can also be visualized. This is a great means to verify that these algorithms are working as they are supposed to do.

Additionally, we created two groups of shared software:

- *Locutil1*

- *Locutil2*

After this brief description of our set of tools, Figure 2 depicts how this tools support the research process. It shows the same shape as Figure 1 but this time the boxes are named according to the set of tools that do the job for the particular sub-process.

Locana provides tools to define the testbed; Loctrace contains tools for the data collection. The sub-process emulation and comparison is supported by Loceva. Locana as well as Loclib provide tools to allow the verification of the research results.

In Figure 3 the overall architecture is depicted. It shows that Locutil1 and Locutil2 are shared by almost all groups of research tools. Only Loclib requires Locutil1 but not Locutil2. The difference between these two utility packages is that Locutil1 is implemented using Java J2ME and Locutil2 by using J2SE. Please see Section 3.1.2 for a

[2] http://www.nmea.org/pub/0183

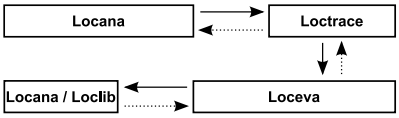


Figure 2: This figure shows how our research tools fit into the research process.

description of the implementation and an explanation of the abbreviations. Loceva and Locana rely only on Locutil1 and Locutil2 but not on Loclib because direct interaction with sensor hardware is not required by these sets of tools.

As depicted in the figure, we distinguish between library and application. A library is a piece of software that cannot be run by itself whereas an application can be invoked by the user to accomplish a specific task. Applications are usually a combination of different libraries and additional source code. Loclib, Locutil1 and Locutil2 are libraries that cannot be run in standalone mode. Loctrace, Loceva and Locana are sets of applications that rely on these libraries.

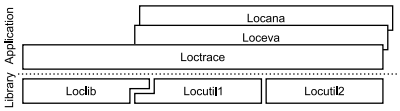


Figure 3: Software architecture of Loclib, Loctrace, Loceva, Locana, Locutil1, and Locutil2.

Hardware Components The only group of tools that requires specific hardware is Loclib. The Linux Wireless Extensions are utilized by Loclib to access data provided by 802.11 wireless LAN network cards. These extensions work well with all 802.11 network cards that are supported by Linux. Furthermore, Loclib contains a few model-specific options that are only working with Intel Pro/Wireless 2200BG network cards. For instance, we implemented passive scanning [9] as described in the IEEE 802.11 standard into the driver of this network card. Additionally, we implemented a novel scanning technique called Monitor Sniffing [11] exclusively for this driver.

3.1.2 Implementation

Our research tools are mainly implemented in Java [19]. For most tools we used the Standard Edition (J2SE), whereas we implemented most parts of Loclib by using the Micro Edition of Java (J2ME) in the Connected Device Configuration (CDC). We selected Java as the main programming language for two reasons:

- Java is a modern object-oriented application programming language.
- Java is platform independent.

The first reason allows us to rapidly implement our tools and applications. The platform independency of Java enabled us to write parts of Loclib in such a way that

it runs on small mobile devices (e.g., a cellphone or a PDA) as well as on desktop computers.

To get access to driver-specific information we implemented a part of Loclib in C++. To be more precise, we developed a library to control the wireless LAN network card in a close manner by directly accessing its driver interface. So far, this library works only on Linux but can be easily ported to Microsoft Windows or other operating systems. The Java Native Interface (JNI) is used to invoke functions of C++ and for transferring data from the C++ domain to the Java domain.

In total, our research tools consist of more than sixteen thousand lines of Java code and more than 1500 lines of C++ code.

3.2 Loclib

As already mentioned in the previous section, Loclib interconnects sensor hardware and applications by gathering sensor-specific data and converting it into location information if required. Loclib is organized in three layers:

- *Sensor-specific data collection layer*
- *Data conversion layer*
- *Location application programming interface layer*

Figure 4 shows the software architecture of Loclib and the aforementioned layers. The sensor-specific data collection layer gathers data from different sensor hardware. At the moment, Loclib is able to retrieve data from 802.11 wireless LAN network cards, NMEA-enabled GPS receivers, and digital compasses. To collect data, drivers can be accessed, or if it is feasible Loclib communicates directly with the sensor in question. For instance, digital compasses are directly queried, as well as NMEA-0183 devices. In case of 802.11 network cards, the signal strength of access points in communication range is retrieved from the driver. The data collected from the sensor-specific data collection layer is usually forwarded to the data conversion layer for further processing. However, the data can also be directly accessed by using so-called handlers. Handlers are pre-defined interfaces to allow applications such as Loctrace to access sensor-specific data.

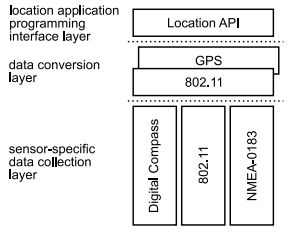


Figure 4: Software architecture of Loclib.

The data conversion layer is responsible for converting data provided by the sensor-specific data collection layer into a position estimate that can be utilized by the Location API. To accomplish this, so far, GPS or 802.11-based positioning algorithms (see [7] and [13]) can be used. The data conversion layer is able to switch between 802.11-based positioning and GPS-based positioning if one of the sources fades out and the other is still operational. If both positioning services are available, GPS is preferred. Should both sources of position information be unable to deliver the required data an error code is returned instead of a valid position.

The location application program interface layer implements the well-known and widely used Location API to deliver location estimates to applications. This is a standardized way to hand over location information. Especially on mobile devices, this is a wide-spread approach to support location-based services.

3.3 Loctrace

Loctrace contains only one application: *Tracer*. Tracer is used to collect the data required to create fingerprint databases. To achieve this goal, Tracer is build on top of Loclib and directly retrieves sensor-specific data (e.g., the signal strength of access points within communication range in an 802.11-based wireless network). It contains a graphical user interface (GUI) to make the configuration process easy to handle (e.g., select a scanning mode and the scanning device). Various parameters such as the number of scans or the delay between two consecutive scans are also configurable through the GUI. If the trace process is started, a histogram pops up in the bottom part of Tracer showing the access points within communication range and their corresponding signal strength distributions. A screenshot of the Tracer GUI is depicted in Figure 5.

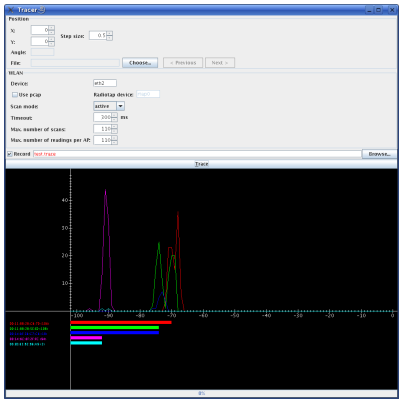


Figure 5: *Tracer* is part of Loctrace and mainly used to trace signal strength values of access points in communication range.

The data collected from Loclib is stored in a human-readable trace file and contains lines that adhere to the following format:

```
t="Timestamp"; pos="RealPosition", id=\\
```

```
"MACofScanDevice"; degree="orientation";\\
"MACofResponse1"="SignalStrengthValue",\\
"Frequency","Mode","Noise"; ... \\
"MACofResponseN"="SignalStrengthValue",\\
"Frequency","Mode","Noise"
```

The \\ are actually not part of the format and are only included to indicate an artificial line break. These line breaks are just added to display the format in a space-limited column. The following describes the meaning of each element of the format:

- *t*: time-stamp in milliseconds since midnight, January 1, 1970 UTC
- *pos*: the physical coordinate of the scanning network device
- *id*: MAC address of the network device used for scanning
- *degree*: direction of the user carrying the scanning device in degrees (only set if a digital compass is available)
- *MAC*: MAC address of a responding 802.11 peer (e.g., an access point or a device in adhoc mode) with the corresponding values for signal strength in dBm, the channel frequency, its mode (access point = 3, device in adhoc mode = 1), and noise level in dBm.

Trace files generated by Tracer are a major building block for our overall research process. These files can be used by Loceva to evaluate and emulate different positioning algorithms and scenarios. Furthermore, the traces can be displayed for visual inspection by tools of the Locana package. Finally, these traces can be used as an offline fingerprint database during normal operation of an 802.11-based positioning system.

3.4 Loceva

Loceva is the place where almost all the research happens. It contains only one application but this tool is the playground for positioning algorithms. Loceva comprises two parts: a *management* part that handles emulation and selection of different scenarios, and an *algorithm* part that contains positioning algorithms.

To make it easy to evaluate and compare algorithms currently under research, Loceva contains a management part that enables emulation in general and allows to easily select different kinds of scenarios. For this, Loceva utilizes trace files created with Loctrace to emulate a specific scenario. Such an emulated scenario can then be used for a comparison of different positioning algorithms. This makes sure that differences in the results are based on the positioning algorithms and not on the environment that changed over time in a way beneficially for one particular algorithm.

The creation and management of various scenarios is enabled by so-called filters. Filters create different scenarios by disabling or selecting different objects of a trace file. For instance, a MAC filter artificially switches off access points even if they have

been part of the trace file. Another example is the position filter that disables different reference points of the fingerprint database based on their coordinates.

The positioning part contains various positioning algorithms to make it easy to compare newly envisioned algorithms with state-of-the-art ones. The following list shows the positioning algorithms that are part of Loceva. The list is grouped by the research projects that have invented them:

- *RADAR*: Nearest neighbor(s) in signal space [3], k nearest neighbor(s) in signal space [2]
- *PlaceLab*: K nearest neighbor(s) p unknown, Ranking [6]
- *Rice*: Histogram [14], Gaussian [7]
- *Horus*: Horus [22]

Although the main focus of Loceva is on positioning algorithms, it also contains a few continuous user tracking algorithms:

- *RADAR*: Viterbi-like algorithm [2]
- *Rice*: Tracking [7]
- *HORUS*: Horus [22]

After selecting a certain scenario and positioning algorithm, Loceva computes the position error that would have occurred in this setting. The position error is defined as the Euclidean distance between the actual position of the user and the position estimate calculated by the algorithm. At the end of each emulation the average position error is printed, and a graph showing a cumulative distribution function of the position error (as shown in Figure 6) is generated. Such a graph can be used to compare the position accuracy of different positioning algorithms by determining the median, 95^{th} percentile and so on.

Additionally, Loceva can be enabled to create a file that contains a log of intermediate results computed by the selected positioning algorithm. This log can be used with Locana to analyze the behavior of the positioning algorithm in question.

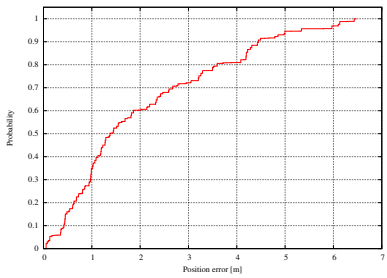


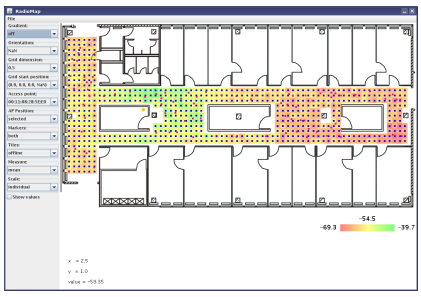
Figure 6: A cumulative distribution function of the position error generated by Loceva.

3.5 Locana

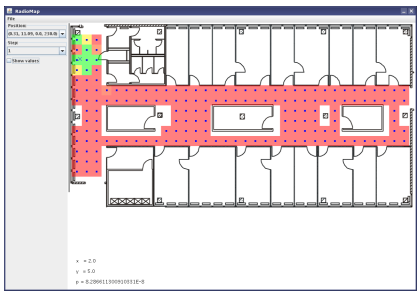
A whole bunch of tools are grouped together in the Locana package. Locana contains many small tools that are supposed to perform special jobs. Most of these tools verify the output of Loctrace and Loceva, or list a certain object of a trace file. For instance, a tool called *AccessPointLister* prints out all the access points and how often they have been heard for a given trace file.

However, Locana contains also a powerful tool named *RadioMap*. RadioMap offers two modes of operation: *loctrace* and *loceva*. The former mode visualizes trace files generated by Loctrace. This feature is mainly used to visually investigate a fingerprint database. For each reference point and access point the number of readings, the average signal strength and its standard deviation can be displayed. The same can be displayed for online points as well. Furthermore, the grid dimension and starting point of the grid of reference points can be varied. A screenshot of RadioMap running in loctrace mode is shown in Figure 7(a).

As previously mentioned, Loceva is able to optionally generate a file that logs intermediate results of positioning algorithms. Such a log file can be displayed in loceva mode of RadioMap. This helps to better understand how the selected positioning algorithm works, and to verify that the implementation works as it is supposed to. Figure 7(b) depicts a screenshot of RadioMap running in loceva mode.



(a) RadioMap showing the signal strength distribution for a particular access point over a testbed. Additionally, the online points are marked by magenta markers, and the reference points are painted in blue.



(b) The probability that the user is located at a particular point is shown for each reference point. The real position of the user is marked by a cross.

Figure 7: *RadioMap* visualizing data generated by Loctrace and by Loceva.

4 Discussion

In this section, we discuss topics that do not fit into the structure of the paper but should be mentioned anyway.

It is extremely important to verify intermediate results because otherwise it is very difficult to figure out what is going wrong if the research process continues. For instance, it makes no sense to work with trace files that are erroneous because what ever

will be the result in the long run is useless. This is a reason why we spend reasonable amounts of time and efforts in creating tools to verify and visualize results. The time spending is paid back as soon as the tools are in place and speed up the research work.

In this paper, we mainly focus on research of algorithms to determine the position of a user. However, the research methodology and our tools are not limited for this task. We used our tools also to answer the question of how fingerprint data-sets can easily and automatically be distributed to mobile devices and at the same time respect the limited storage capabilities of these devices [10]. This shows the potential of our tools to cover the whole research area of 802.11-based positioning systems.

We offer researchers who do not need their own testbed a shortcut by providing two sets of trace files (see [12]). These trace files allow them to directly start with their research work without going through the time-consuming and tedious process of setting up a testbed and collecting data.

5 Related Work

The famous network simulator ns-2 [1] targets mainly the simulation of TCP, routing, and multicast protocols over wired and wireless networks. However, ns-2 also contains an emulation module that enables it to send packets to and receive packets from a real-world network. While ns-2 focuses mainly on network protocols it is not suitable for 802.11-based positioning systems research because its emulation module does not support signal strength information of received packets.

One of the largest research projects in the area of positioning systems in the last few years is PlaceLab [15]. PlaceLab focuses not only on 802.11-based systems, instead it tries to gain position information from all possible sources of beacons (e.g., Bluetooth [15], GSM [20]). PlaceLab provides software that enables users to easily setup a positioning service. However, the tools needed to do research have not been released by the project management.

Bahl et al. first introduced the concept of fingerprinting to 802.11-based positioning systems in 2000 [3]. Although, they talked about their research methodology and the implementation of their tools in the paper and a subsequently published technical report [2], they never released their tools to the public.

Researchers at Rice University undertook two research projects to investigate 802.11-based positioning systems [7] [14]. Both projects focused on algorithms to compute the location of a mobile device while being carried around by a user. For the evaluation of the novel algorithms, two experiments were conducted. Although the authors of these papers talked about their research methodology, they neither talked about details of their implementation, nor were their tools released.

The Horus project is one of the projects in the area of 802.11-based positioning systems that recently gained considerable coverage in the news [22]. While the authors talk about their research process they refused to release any tools regarding Horus because they filed a patent application that covers algorithms invented during their research for Horus.

6 Conclusions and Future Work

In this paper, we have described our research tools and our research methodology as we use them for our research work in the area of 802.11-based positioning systems. We first introduced our research methodology and then discussed each of our research tools. We contribute our research tools to the community to initiate the creation of standardized tools. Researchers benefit from standardized tools because they can focus on the actual work and not on re-inventing the wheel (e.g., re-implementing state-of-art positioning systems for comparison). Furthermore, standardized tools make results more creditable because tools will be less erroneous over time. Finally, a standardized process of data collection and processing makes results more comparable.

Currently, we are working on documentation that covers the source code as well as our research methodology in more detail. We try to make the documentation as user-friendly as possible. For this, we utilize a Wiki so that the documentation can be extended and updated easily. This is especially important if other researchers begin to contribute to our project.

Availability

The research tools, as presented in this paper, are available under the terms of the GPL on our website [12].

Acknowledgments

We would like to thank Alexander Biskop and Andreas Färber for their help implementing the tools. Furthermore, the authors acknowledge the financial support granted by the *Deutsche Forschungsgemeinschaft* (DFG).

References

[1] The ns-2 network simulator. Website: `http://www.isi.edu/nsnam/ns/`.

[2] P. Bahl and V. N. Padmanabhan. Enhancements of the RADAR User Location and Tracking System. Technical report, Microsoft Research, 2000.

[3] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proc. of the IEEE InfoCom*, 2000.

[4] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. EasyLiving: Technologies for Intelligent Environments. In *Proc. of the International Symposium on Handheld and Ubiquitous Computing*, 2000.

[5] P. Castro, P. Chiu, T. Kremenek, and R. R. Muntz. A Probabilistic Room Location Service for Wireless Networked Environments. In *Proc. of the UbiComp*, 2001.

[6] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy Characterization for Metropolitan-scale Wi-Fi Localization. In *Proc. of the ACM MobiSys*, 2005.

[7] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. Practical Robust Localization over Large-Scale 802.11 Wireless Networks. In *Proc. of the ACM MobiCom*, 2004.

[8] E. Kaplan and C. Hegarty, editors. *Understanding GPS: Principles and Applications*. Artech House Incorporated, second edition, December 2005.

[9] T. King. Passive Scanning not supported? ipw2100 developer mailinglist: `http://sf.net/mailarchive/forum.php?thread%5Fid=27023011\&forum%5Fid=38938`, July 2006.

[10] T. King, T. Butter, M. Brantner, S. Kopf, T. Haenselmann, A. Biskop, A. Färber, and W. Effelsberg. Distribution of Fingerprints for 802.11-based Positioning Systems. In *Proc. of the MDM*, 2007.

[11] T. King, T. Haenselmann, S. Kopf, and W. Effelsberg. Overhearing the Wireless Interface for 802.11-based Positioning Systems. In *Proc. of the IEEE PerCom*, 2007.

[12] T. King and S. Kopf. Loclib - A Location Library. Website: `http://www.informatik.uni-mannheim.de/pi4/lib/projects/loclib/`, January 2007.

[13] T. King, S. Kopf, T. Haenselmann, C. Lubberger, and W. Effelsberg. COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses. In *Proc. of the ACM WiNTECH*, 2006.

[14] A. M. Ladd, K. E. Bekris, A. Rudys, G. Marceau, L. E. Kavraki, and D. S. Wallach. Robotics-Based Location Sensing using Wireless Ethernet. In *Proc. of the ACM MobiCom*, 2002.

[15] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proc. of the IEEE PerCom*, 2005.

[16] K. Loytana. JSR 179: Location API for J2ME - Final Release 2. Website: `http://www.jcp.org/en/jsr/detail?id=179`, March 2006.

[17] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. of the ACM MobiCom*, 2000.

[18] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, second edition, December 2001.

[19] Sun Microsystems, Inc. Java Technology. Website: `http://java.sun.com/`, 2007.

[20] A. Varshavsky, A. LaMarca, J. Hightower, and E. de Lara. The SkyLoc Floor Localization System. In *Proc. of the IEEE PerCom*, 2007.

[21] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.

[22] M. Youssef and A. Agrawala. The Horus WLAN Location Determination System. In *Proc. of the ACM MobiSys*, 2005.