

## A distributed framework for location sensing systems

Patrick RÖDER<sup>1</sup>, Mario HOFFMANN<sup>2</sup>, Matthias RITSCHER<sup>3</sup>

<sup>1</sup>Fraunhofer – Institute Secure Telecooperation, Darmstadt, Germany, e-mail : Patrick.Roeder@web.de

<sup>2,3</sup>Fraunhofer – Institute Secure Telecooperation, Darmstadt, Germany,  
e-mail : (Mario.Hoffmann, Matthias.Ritscher)@sit.fraunhofer.de

**Abstract** - Location sensing systems became very popular and widely used in the recent years. Many places are already equipped with a great variety of location sensing hardware running specifically tailored software to use it. According to their focused nature, different systems are incompatible to each other. The development of a highly available, context-aware location service raises the challenge of integrating a large quantity of location sensing systems, consequently the service complexity increases. In order to maximize the usefulness and flexibility of already existing location sensing systems as well as to enable a service development separated from special positioning issues, we take advantage of a modular, 4-layered framework design, which hides the specific details of the location sensing system. The client-server architecture ensures that even clients with limited processing power and/or storage capacity are able to participate in the system. The framework features an abstract uniform access to the variety of sensors managed by “Location Servers”<sup>1</sup> and a model of hierarchal zones to describe maps of the environment. Location Servers communicate with each other to exchange information about maps and tracked objects, allowing clients to roam between different location systems. Sensor fusion enhances the accuracy and positioning probability.

### 1. Introduction

Imagine traveling to a company, you are working with to visit your business partner Mr. Miller in his office. As you leave the train at the destination city, your smart phone guides you the way to the proper exit of the train station using its wireless system for location sensing. Outside the station, there is no specific location sensing system, so the cell phone confines itself to use the positioning information provided by the cellular network that it is connected to anyway. Finally, having arrived at the huge office building where Mr. Miller works, your phone switches to a combination of Wireless LAN and Bluetooth proximity location sensing to guide you to Mr. Millers office.

To make this vision possible we developed an appropriate framework that is capable of handling the required techniques and provides uniform location information, which is independent of the actual positioning systems used.

### 2. Requirements

We define a few requirements, to get an idea of what properties the framework should have.

#### 2.1. Flexibility

Due to the pervasiveness and variety of current location systems, extensions to and adaptations of the framework should be easy. To achieve this goal it is common practice in computer science to utilize a layered, modular design. Moreover, the design should be simple and well documented to enable other developers to adapt the framework bottom-of-the-line.

---

<sup>1</sup> This term describes the server of a location sensing system

## **2.2. Support of all kinds of sensors**

We want to create a framework, which you can use, wherever a system for location sensing is required. Therefore, it is necessary to support all kinds of location sensors. In particular, there should also be no difference between sensors for outdoor and indoor usage, enabling seamless transitions between different location sensing techniques.

## **2.3. A distributed system**

Every organization or company with an own location sensing system (typically indoor) is seen as a unit administrated in its own domain with database entries about items in the range of the covered area. In this inherently distributed system, the proposed system must provide methods to exchange information between different units.

## **2.4. Platform independency and no proprietary software**

The aspired framework has to be in many different computing environments. Therefore, it is necessary to have a platform independent software basis and programming language. For the same reason we do not contemplate to use proprietary software or frameworks.

## **2.5. A flexible model for describing maps**

A flexible model for representing maps is required, suitable as well for the description of the indoor architecture of buildings as for defining outdoor areas like parts of a city. By means of this location model, containment questions like “Who is in the same room, building or part of the city?” have to be answered. In addition to this type of question, the representation model should provide answers to questions like: “Where is the next restaurant?“, not about containment, but the environment and objects nearby.

## **2.6. Security and privacy**

Many users adherently link location-sensing systems to the vision of being watched by “big brother”. To build up a trustworthy and reliable system, security and privacy mechanisms for identification and access control to the system are badly needed. The communication channel and the persistent storage of sensitive data must be protected using secure cryptographic algorithms.

## **2.7. Making use of the infrastructure of our institute**

Not a point of general interest, but important for us, is using the existing infrastructure of our institute, namely the coverage of a wireless LAN network and the fact that many doors are equipped with RFID tag readers which are managed by a central server. Lately many of our stationary computers have been equipped with Bluetooth modules to communicate with printers and other devices on the users desktop. These stationary devices and the RFID readers are used build up an indoor positioning system based on the proximity location sensing technique. Moreover, we use the signal characteristics of the WLAN and GSM cellular networks for positioning.

## **3. State of the art**

Today there is a variety of monolithic location sensing systems all using just a single type of sensor and specialized for a specific task [1]. There is the need for a system which handles more than one location sensing system and which is not restricted to a special field of usage. The location stack [2] partially meets our requirements, as it features a 7-layered design with the capability of handling several different sensors combining their readings using dedicated fusion algorithms. Roth [3] also introduces a hierarchical, self-organizing location-sensing infrastructure, which separates the positioning issues from the location based service duties to provide a location information generation independent of the actual location system. To comply our requirements there are several reasons for creating an own system, which is partly based on the considerations of other research projects, but should feature hierarchal maps and the distributed aspect of Location Servers, briefly described in the following.

## 4. Approach

Storing location information in a gigantic database with a huge number of clients is not an appropriate solution for a system that retrieves location information. On the one hand the amount of raw data is not capable to handle, on the other hand, this type of data is traditionally in the hand of distributed information databases. In our approach, we take advantage of this configuration by using a server at each organization or area, for database administration and location aggregation. All calculations and maintaining the databases are carried out by so-called Location Servers, which usually have more computational and memory resources at their disposal than the typical mobile device (mobile phone or PDA). The client is needed solely to report readings from various location sensors and to perform queries.

We use a hybrid location model of physical and symbolic location information. Physical locations are needed for routing purposes and distance measurements, whereas symbolic locations are more easily interpretable both by humans and by services built on top of our service. We use three-dimensional hierarchal maps on the server side to derive a symbolic location from a physical one.

### 4.1. Client – Server components

We take advantage of a client/server approach with mobile clients on one side and Location Servers on the other.

Location Servers handle the maintenance of the databases as well as the fulfillment of database queries or complex calculations like sensor fusion. These multiple and distributed servers (one per area or organization) exchange data among one another to keep their databases up to date.

Mobile clients with low computational resources and memory limitations like mobile phones or palmtop computers use the server to perform complex computations for them.

### 4.2. Distributed Location Servers

There are two main reasons for the necessity of intra Location Server communication. On the one hand the servers need to exchange data like content, maps or information about tracked objects, because their database entries and sensor range only incorporates their covered area.

On the other hand, the settlement of these systems is just in local administration. To exchange information static lists, which include an enumeration of other Location Servers, being updated manually when a new system arises, are sufficient at the current prototype state. But in the future, there should be a self-organizing structure including a scalable, Web Service based approach for intra-Location Server connection.

If a service queries the Location Server for position information of a tracked object e.g. a client, the Location server can ask other Location Servers whether they have the desired information in case it does not have it itself. Tracking users that roam in the coverage area of “foreign” Location Server is done by a structure similar to the GSM cellular network [7]. In this context it has to be mentioned that every tracked object has a reference to its dedicated “home server”<sup>2</sup>. All other servers appear to be “foreign” Location Servers the client roams to.

This approach requires globally unique client IDs to obtain the home server of a tracked object. This could be carried out by a central database service, matching client IDs with the appropriate home Location Server. Because of scalability and privacy issues, such a solution is not suited for a hierarchical and distributed system like ours. The client IDs in our approach are structured similar to email-addresses, in a human readable format.

This ID will be derived from the user’s real name, which needs to be unique within an organization. The client IDs becomes globally unique with the symbolic location of its “home server” as a postfix, e.g. `patrick.roeder@fraunhofer-sit.darmstadt.de`. Here the term “fraunhofer-sit.darmstadt.de” is not a DNS-Domain, but the symbolic location information of our institute’s Location Server. When a client connects to a foreign server, this system creates a new entry for the client in its visitor database and informs the client’s home server that the client roams to its coverage area.

---

<sup>2</sup> Location Server of the organization or area, where the client initially registered

The home server creates an entry in the home database noting that the client is now being tracked by the foreign system. All tracking requests for the client of the home system will be forwarded to the foreign system.

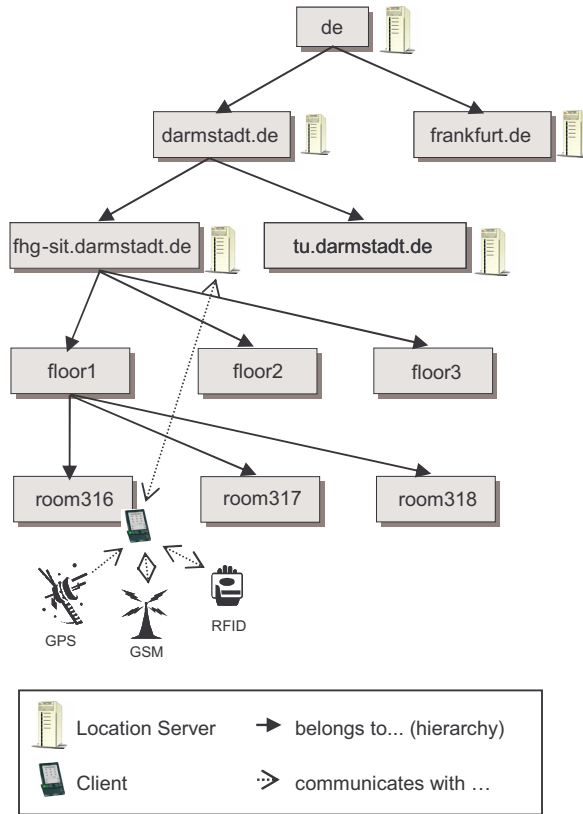


Figure 1: Logical hierarchy of the framework

We illustrate this with a short example. You want to receive a phone call for your co-worker John Smith who is currently visiting a partner institute where he only brought his PDA. You use the automatic phone call routing service, a service that uses the location information provided by our platform, where the service asks Johns home Location Server for John’s current phone number. The Location Server consults its database that John is connected to the system of the partner institute. The server forwards the request to the other system and asks for the position and the enclosing room where John currently is. The home server also requests the phone number of the phone in that room. This number is returned to the phone routing service, which then automatically calls John.

## 5. Location model

Our Location Server approach provides two different types of positions: physical and semantic positions, in a globally unique representation. We support these types of the location information due to the fact that miscellaneous location-based applications or services call for different location types. Indeed, positioning systems provide different types of location data, not considering the location information required by the application. On this account, the Location Server performs the translation between absolute and semantic coordinates.

### 5.1. Hierarchal zone model

Symbolic positions provided by our platform use the zone algebra, which is based on a model of hierarchal organized zones to describe the real world. The model is inspired by the hybrid location model [6], but modified to that effect that there is no special syntax for describing positions. The zones have the shape of a right prism, i. e. a polyhedron with some polygon as its top and bottom face lying on top of each other, so that the other faces are all rectangular. The heights of the top and bottom face are specified in meters above the ground. To model outdoor areas we use zones with a bottom face below the earth's surface and a top face residing in the ionosphere. In our opinion, the assumption that zones have this special form does not severely limit the modeling of real world infrastructures and district areas, since the zone model is not intended to be used to describe city maps or comparable. This is out of the scope of our framework and should be handled by a separate application using our service as a basis.

### 5.2. Zone algebra

The zones are organized in a hierarchal manner, so that one can answer containment questions like finding who is on the same floor, in the same building or on the same campus area. In addition to this structure of zones, sets of zones can be defined to put semantics on the abstract zone data structure, for instance merging the rooms in a building that belong to the same workgroup.

With the aid of these zone definitions and the succeeding transformations, the platform can provide physical and semantic location information from any location input. On the one hand, the transfer of physical position into semantic location information is carried out by finding the smallest enclosing zone through the Location Server, since it holds all information about the shape and vicinity of zones in its coverage area. The symbolic position output is structured similar to the Domain Name System (DNS) notation, e.g. room316.floor1.fraunhofer-sit.darmstadt.de, which enables service-side query analysis like

```
if (pos.semantic.endsWith("darmstadt.de"))
    System.out.println("We are in Darmstadt");
```

On the other hand semantic positions are converted to physical areas described by tuples of points using the Location Server zone database.

By the hierarchal setup of zones in combination with the possibility to build up sets of zones we achieve a feasible framework according to our requirements.

## 6. Software architecture and Implementation

Strongly inspired by the location stack [2] we use a 4-layered structure for our framework, to make the framework less complex and to reduce the interaction between separate layers. Figure 2 illustrates the architecture.

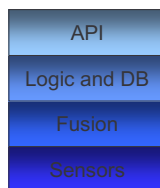


Figure 2: Schematic diagram of the layered architecture

### 6.1. Sensor layer

This layer describes the abstract concept of a location sensor. The layer is divided into two modules: sensor driver and sensor data abstractor. The former provides low-level software that gathers raw sensor data from

the corresponding sensor hardware. We cannot model the two modules as individual layers, because the interface between them depends on the type of described sensor.

These sensor drivers typically run in same process as the client Location Server software, in fact they can run on any device in the network to aggregate sensor data. All sensors are treated like mobile clients, enabling the usage of the same software module stack for client-side, mobile or network-fixed sensors. In contrast to other considerations [2], the sensor layer also converts the raw sensor data to a standardized representation, which is accessible by the next layer. The native sensor data is sent to a sensor abstractor, which runs in Location Server process.

These abstractors optionally have access to a database to query for instance the physical or semantic location of a RFID-Tag reader or a Bluetooth host. The standardized representation of actual positions is obtained using sets of tuples including the estimated location, the probability distribution and the quantity of the uncertainty based on a Gaussian distribution. Locations can comprise the estimated physical location of the tracked object or sets of physical positions. Forming sets of possible locations allows us to model very complex situations like the characteristics of the WLAN systems we want to use. Each point in the set features its own probability distribution around the point.

The appropriate shape of the probability distribution depends on the location sensor in use. A fair approximation for different location sensing systems is a radial probability distribution, i. e. one that is spherically or circular symmetric. This assumption suits some location sensors, e.g. the position uncertainty of the GPS system can be represented in form of a circular area. It is less justified for location sensors such as cellular networks, WLAN, Bluetooth, RFID or IR-signals because the corresponding wireless signals are attenuated or blocked by obstacles forming arbitrary shapes. Nevertheless, we use this standardized representation based on the maximum propagation distance of the wireless signal and the maximum detection area and report it to the higher levels, to perform plausibility checks.

## 6.2. Fusion layer

The fusion layer takes data from all sensors in the uniform format used by the sensor layer and calculates a single resulting position. We can use many different algorithms, depending on the requirements we have. Since its modular characteristic with a well-defined interface, it is easy to exchange the fusion algorithm.

Algorithms with very high precision and high requirements in computing power are available, as well as simpler and faster algorithms, which can handle a much higher number of simultaneous tracked clients.

An example for such an algorithm is to calculate a weighted average (weighted by probability) of the reported positions. More dedicated algorithms keep a record of the recent positions and the average moving speed of the client. When the fusion algorithm examines a set of possible new locations, all these are compared to the average moving speed of the client. Positions where the client would have traveled at high velocity inside a building, while moving just before at walking speed are unacceptable. Bayesian filters [4] are a class of algorithms, which can handle this computation very efficiently.

To enhance the accuracy of a history-based tracking, a graph can be stored, which describes the possible routes of a client. This takes into account that humans usually do not walk across walls or on top of rivers, instead of doors and bridges. The graph, which describes these possible routes, is called a Voronoi graph [5].

## 6.3. Logic and database layer

The logic and database layer maintains databases for all tracked objects and their environment. It can store the actual semantic or geometric position and the history of locations (if desired) of the tracked objects. Contextual information like names, room numbers and other properties are also located at this layer. Digital representations of the environment are stored as a hierarchal zone model, whereas the logic layer can look up a geometric or semantic position in the zone database to get additional symbolic information like the room or area in which the position is located.

## 6.4. API layer

The API layer describes the interface of our service used, to obtain position and contextual information as well as basic relationships like containment, and proximity, which we implemented as common and

expandable set of features. The API is used both for client/server communication and for communication among the servers.

We decided to implement our service as a Web Service [6], since they offer an interface, which is machine and human readable. Using Web Services, you can perform queries by using an arbitrary web browser or SOAP client, available on almost every computing platform. Moreover, you can build other services like the previously described telephone call routing system on top of our service using the well-defined Web Service interface, which you describe in a format that can easily be parsed by machines.

Web Services bring their own discovery and lookup mechanisms, so we do not need to think of own mechanisms. You can register your service in a central entity, the UDDI (Universal Description, Discovery and Integration) registry, which is part of the Web Services architecture, to make it available for interested clients.

## **7. Service discovery and lookup**

You need the network address of the location service to use it. Using a static list of Location Servers with their corresponding address is unfeasible in a highly dynamic system with mobile clients that roam between many different systems. Moreover, you have to decide whether foreign clients may use your system.

### **7.1. Public or private service?**

Your organization can decide whether to offer the location service only locally for their staff or to take part in the global network of Location Servers. Taking part in the global system makes it possible for others to locate the clients of your organization. You can compare this to having a locally restricted email system or using the global internet email system.

### **7.2. Private Location Service**

Clients have to retrieve the network address of the location service if they enter a new building or organization to use the service. Therefore, every organization runs a private, internal UDDI registry, where you register the location service among the other services, which you have built on top of the location service. Clients within your organization can lookup the address of the location service using this private registry. It is a common practice to announce the address of the local registry using the DHCP protocol [9].

### **7.3. Public Location Service**

If your organization wants to take part in the global system, you must register your service in the public worldwide UDDI registry. The network address of the Location Service and the physical location of your organization is registered. The symbolic location (e. g. "fraunhofer-sit.darmstadt.de") serves as a unique name for your organization. Other clients perform either a query by that name if they are looking for this particular service or a query by physical location if they are looking for services around them.

### **7.4. Locating foreign clients**

To locate a foreign client you need know its client ID, for example "patrick.roeder@fraunhofer-sit.darmstadt.de". Since there is always one dedicated server, which is responsible for a specific client, you need to find the network address of the server corresponding to the client ID.

To get the network address you do a query in the global UDDI registry with the part after the @-sign of the client ID. In our example, you perform a query for "fraunhofer-sit.darmstadt.de" and get the network address of our Location Service. Having the address of the Location Service, you can perform a query for the location of the client "patrick.roeder" and finally get his physical location.

## **8. Security aspects**

The development in the field of mobile data transmission technologies and wireless devices combined with attractive mobile services gave rise to a new set of threat scenarios and additional security requirements. In

this context, it is important to recognize that offering and integrating such services for wireless devices requires a carefully balanced security model. In addition to the form of logging connections and their duration in the Internet, the use of location-based services also makes location information potentially available to third parties.

So dealing with security aspects in the field of the location based services, questions in terms of authentication, access control, data transmission security, and privacy protection have to be answered.

In this platform we put emphasis on the privacy protection aspects: A role based access control to the location information for services, which limits the access to the user's physical location, seems to be feasible, where the user himself can authorize a service to locate him/her. The service uses the location, but does not publish it (compare example in chapter 4.2 of routing a telephone call).

Encryption is needed to prevent eavesdropping, whereas mutual authentication, which uses a shared secret between the two parties, or using a public key infrastructure, is applied to prevent spoofing of client IDs. With the help of secure middleware, end-to-end security is achieved without the necessity for encryption performed in the network infrastructure. In addition to the security aspects of access control and the communication channel, the client-side application has to guarantee a persistent reading, transmission and possibly storage of the sensitive sensor data.

## 9. Conclusions

We presented a distributed, modular framework for location sensing systems, which enables a uniform location information interface for context-aware services. The framework is capable of maintaining hierarchal structured maps and exchanging this information with similar systems. Clients can roam in foreign location sensing systems and inform their home server about the local Location Server they are currently connected to. The framework can handle miscellaneous location sensing systems including multiple sensors and may be used indoor and outdoor with seamless transitions between the different states.

## References

- [1] Jeffrey Hightower and Gaetano Borriello, "A Survey and Taxonomy of Location Systems for Ubiquitous Computing", University of Washington, *Technical Report UW-CSE 01-08-03*, 2001
- [2] Jeffrey Hightower "The Location Stack: A Layered Model for Location in Ubiquitous Computing", University of Washington, 2002
- [3] Jörg Roth, "Flexible Positioning for Location-based Services", University of Hagen, 2003
- [4] Dieter Fox et al, "Bayesian Filters for Location Estimation", University of Washington, 2003
- [5] Lin Liao et al, "Voronoi Tracking: Location Estimation Using Sparse and Noisy Sensor Data", University of Washington, 2003
- [6] James Snell et al, "Programming WebServices with SOAP", O'Reilly, 2001
- [7] C. Jiang and P. Steenkiste, "A hybrid Location Model with Computable Location Identifier for Ubiquitous Computing", Carnegie Mellon University, Pittsburg USA, 2002
- [8] J. Eberspächer and H. J. Vögel, "GSM - Global System for Mobile Communication", *Teubner Verlag Stuttgart*, 1997
- [9] IETF Network Working Group, "Request for Comments: 2131, Dynamic Host Configuration Protocol", Bucknell University, 1997