A multimodal distributed architecture for indoor localization

M.A. Martínez, F.J. Villanueva, M.J. Santofímia and J.C. López
Computer Architecture and Networks, School of Computer Science,
University of Castilla-La Mancha, Ciudad Real, Spain.
Email: [miguela.martinez, felix.villanueva, mariajose.santofimia, juancarlos.lopez]@uclm.es

***Abstract*—The indoor location problem, despite being a hot topic for the research community, it is still an issue with great room from improvement. The poor results reported by the use of isolated location technologies are leading current research efforts to work in the combination of different technologies. This work presents an object-oriented distributed architecture that, supporting different location technologies, is capable of providing a multimodal location system. The main contribution of this work is therefore the proposal of a system in which the information retrieved from different location technologies can be combined in order to provide high level services, such as user device tracking or *watch areas*.**

I. Introduction

People location for indoor environments is the cornerstone of higher level applications such as real-time tracking, user activity recognition, user and robot navigation or target-of-interest monitoring, just to name a few. The goodness of the results provided by those high level applications are therefore dependent on how precisely people can be located. In this regard, the literature review yields different approaches to people location, ranging from those that cover mathematical aspects for estimation purposes [1], to those that resort to available indoor technologies [2] such as WiFi[3], 802.15.4[4], RFID[5], or Bluetooth[6].

Relevant research is also being done in providing *ad-hoc* solutions to the positioning problem through different techniques, such as augmented reality[7]or audio[8].

However, last advances in indoor location are being achieved by approaches based on the composition of the information retrieved from different location technologies, improving therefore the resultant indoor location system [9][10]. The authors of this work strongly believe that efforts must be addressed on this direction, at least until a more accurate technology appears on the scene. This work has been mainly motivated by the need to overcome the lack of precision of the current positioning technologies. To this end, the range of technologies used to gather location data must be broaden, rather than being just constrained to a single or a small group of technologies.

The multimodal location architecture proposed in this paper is characterized for being technology independent, scalable and auto-configurable. Adopting the object-oriented philosophy, this work provides a high level interface in order to offer location services to external systems in a technologically-transparent way.

The remainder of this article is organized as follows. Section II presents the foundation of the location-based system (LBS) proposed here. Section III describes the proposed architecture. Finally, Section IV presents the conclusions drawn from this work along with the future work directions.

II. A distributed Location Based System

The main drawback of current location systems is that positioning is based in the information retrieved from single location technologies, rather than in the combination of several (cooperating) technologies. Furthermore, those systems provide location services to third-party applications by means of monolithic modules in which the location technology cannot be decoupled from the service itself.

For the sake of minimizing the coupling between the location technology and the service itself, this work proposes the definition of two roles, as known: the *Location Event Provider (LEP)* and the *Location Event Consumer (LEC)*. The former supports the technology-dependent role while the latter is technology independent. The adoption of such a decoupled approach entitles LEP to propagate location events to any LEC.

Nevertheless, role separation it is neither the silver bullet for the LBS, since challenges such as the heterogeneity problem arises. The differences among the location events provided by the distinct technologies makes unfeasible to compose information unless that some homogenization tasks are performed upon such events.

For the sake of homogenization, we have followed an implementation based on the Mobile Location Protocol[11] standard adapted to incorporate the tenets of the object-oriented paradigm. This standard defines geometrical shapes, interfaces to manage the location information and the location concept, that is to say, the location event format.

Additionally, the wide variety of systems that would be using the LBS should be also taken into account. It is therefore necessary to offer a high level interface that allows different LBS users to transparently obtain location information. The idea behind the use of a high level interface is to provide a unique way of managing and dealing with location events. It is also desirable to consider supporting different operative systems, programming languages, mobile devices, distributed components, etc. Both challenges -a common interface and heterogeneous software- are to be faced through the use of

a CORBA-like object-oriented middleware[12]. Adopting a strategy based in physically decoupling the LEP from the LEC allows LBS to provide a common interface to any third-party system while at the same time abstracting it from the software heterogeneity.

The distributed nature of the proposed system poses some key issues such as scalability and fault tolerance. In this regard, a service discovery protocol (SDP) has also been designed and implemented providing the means to compose and replicate the LBS services.

The main contribution of this work is a technology-independent and scalable LBS, designed on the basis of the aforementioned challenges and the named solutions proposed to address them.

III. ARCHITECTURE

The multimodal indoor location system is composed of two logic subsystems: the *Location Event Provider (LEP)* -which implements the location event provider role- and the *Location Service (LS)*. The LS, besides from implementing the LEC role, it also provides highly elaborated services in contrast to the raw ones provided by LEPs.

The LEP is a technology-dependent subsystem intended to detect the presence of those devices involved in the location task (via Bluetooth, WiFi, audio, augmented reality, etc). The LEP is also in charge of propagating location events, using the MLP format, to those other services or systems interested on such events. There should be a LEP for each supported technology, and therefore the role of the service discovery protocol is essential for managing purposes.

The interfaces used by the LEP, in Listing 1, have been implemented in slice (the interface specification language for the used middleware). Through the *MLP.LocationListener* interface the LEP propagates the location events while the *MLP.LocationAdmin* interface allows the LEC to subscribe to the LEP. In other words, it can be said that the LEC uses the *MLP.LocationListener* interface in order to retrieve location events while the LEP manages the LEC interests in receiving location events by using the administration interface *MLP.LocationAdmin*. Moreover, a new interface, based on the use of generic properties, has been designed in order to deal with arising desires to subscribe to a certain type of location events.

The property-based behavior is inspired in the CORBA Property Service[13], and it enacts the system flexibility; for example it is possible to subscribe to a particular LEP that satisfies specific conditions such a concrete resolution in a specific area.

Listing 1. Partial MLP slice definition

```
interface LocationListener {
 idempotent void locateReport(Position pos);
 idempotent void locateSeqReport(PositionSeq pos);
 idempotent void locateRangeReport(PositionSeq pos);
};

interface LocationAdmin extends
 LocationDataProvider {
 void addListener(LocationListener* listener);
 void removeListener(LocationListener* listener);
};

interface LocationAdminProps extends LocationAdmin {
 void addListenerWithProps(
  MLP::LocationListener* listener,
  PS::Properties properties)
   throws PS::UnsupportedProperty;
};
```

These interfaces support the role distinction. However, further challenges need to be faced in order to provide a transparent way of dealing with location events. The prosed solutions are presented underneath.

A. High level interface

In order to support the first issue -to provide a high level interface to other systems- we have created the *LocationService* (LS). The LS is a LEC service interested in receiving location events regarding a particular area. LS propagates events according to a specific semantics.

Location events look similar in any LBS. We are adopted a structure according to the definitions stated in the MLP standard. The *Position* structure is made of three fields, where the first field -*msid*- determines the type and identifier of the event; the second field -*time*- defines the event timestamp, and the last field -*shape*- represents the area where the user has been detected. The *shape* field has also been defined accordingly to the MLP standard.

The LS can be seen as an additional LEP that encapsulates and manages the location events, and provides them for other LEC. Specifically, the LS offers a common interface, as described in Listing 2, for all the LEPs whose area intersects with the LS area. So, the LS subscribes to all the LEPs whose areas are covered by the LS area. Federation and composition, as it will be explained later on, allow us to build LSs with the capabilility of covering any area of a physical infrastructure (building, campus, etc.) by composing low-level LS (room, floor, etc.).

Listing 2. Location Service slice definition

```
interface LocationService extends
MLP::LocationDataProvider{

 void federate(MLP::LocationListener* lsListener,
               PS::Properties properties)
   throws PS::UnsupportedProperty;

 void unfederate(MLP::LocationListener* lsListener);

 MLP::Location getLocation(LS::DeviceProfile device)
  throws UnknownIdentifier;

 void trackingDevice(LS::DeviceProfile device,
                     MLP::LocationListener* listener)
  throws UnknownIdentifier, InvalidProxy;

 LS::DeviceProfileSeq usersIntoArea(MLP::Shape area)
  throws UncoveredArea;

 void watchArea(MLP::Shape area,
                MLP::LocationListener* listener)
  throws UncoveredArea;
};
```

It should be highlighted that the LS may receive location events generated by different technologies. Therefore, it is

necessary to merge the different user positioning events in just one. In order to do so, the LS receives events from different LEPs and performs the geometric intersection between all of them (it is worth noting, once again, that a location event, for being MLP-compliant, is described as a geometrical shape). If the intersection is a null set, the LS propagates the event whose technology is more accurate. In any other case, the LS propagates the event that results from the intersection.

Additionally, location event management is enhanced with some semantic knowledge that supports event filtering by specific areas or from particular devices, allowing device tracking. Providing semantic knowledge also entails the LS to determine the identity of those users located at a specific area, and to federate several LSs. In order to illustrate the federation capability we can consider several LSs at the building floor level (one for each floor) and one LS at the building level, which is actually a federation of floor level LSs. Moreover it is possible to configure the federation, using the properties mechanism. For example a federate system could only be interested in the events coming from a specific technology, while another could need all the raw location events.

B. Composition mechanism

To support the aforementioned geographical semantics, the LS needs to know where LEPs are placed and the area they represent. We assume that each LEP knows their covered area. At this point the LS needs to discover (via the Service Discovery Framework[14]) the LEPs that represent areas that intersect with the one covered by the LS.

To support area definition the standardized *Well Known Text* (WKT)[15] format has been used. Therefore, each LEP needs a shape property defined in its configuration file using the WKT format. On the other hand means to deal with the Service Discovery Protocol are needed. For this purpose some middleware features have been used. In this particular case, the ZeroC Ice[12] middleware has been considered, because it provides an efficient publish/subscribe event service called *IceStorm*, and the capability of creating a grid of computers remotely manageable through the *IceGrid* service.

The composition mechanism, which allow LS to find those LEP located in its area, is carried out in two different ways.

1) LEP discovery: In this method the LS starts the composition mechanism, therefore finding the LEPs. The LS looks up the LEPs (Figure 1) whose areas are covered by the LS. To do so, the LS creates an event channel to retrieve the search results. Next, the LS sends a look up message that specifies the area, the specific type of server to be discovered (a LEP) and the response channel. When the LEPs receives the discovery event they check their represented area and reply to the LS consequently. Finally the LS subscribes to the location event channel of the corresponding LEPs.

2) LEP announcement: Through this method the LEP notifies its presence to the system, and the LS can subscribe to it (Figure 2). When a LEP starts to run, it announces itself to the advertisement channel. The LS was subscribed to that channel and consequently receives the LEP advertisement. The LS asks

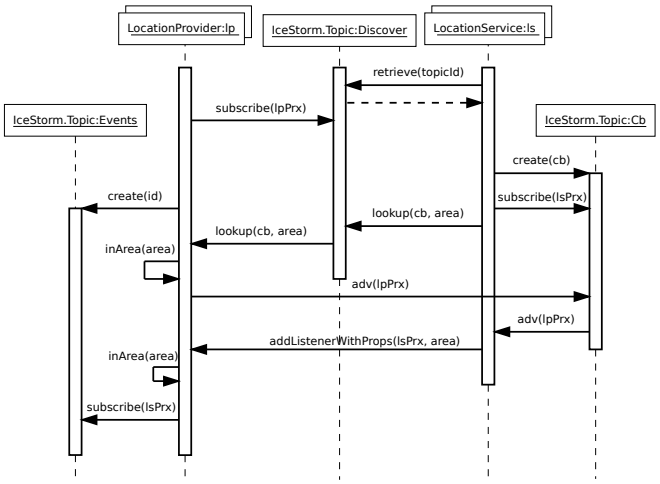

Fig. 1. Sequence diagram of the LS discover of LEPs

to be added to the LEP location event channel. The LEP checks the LS covered area and subscribes the LS consequently. When the LS is subscribed to the LEPs location event channels it will receive the location events. The LS will then propagate these events using the *LocationService* interface.

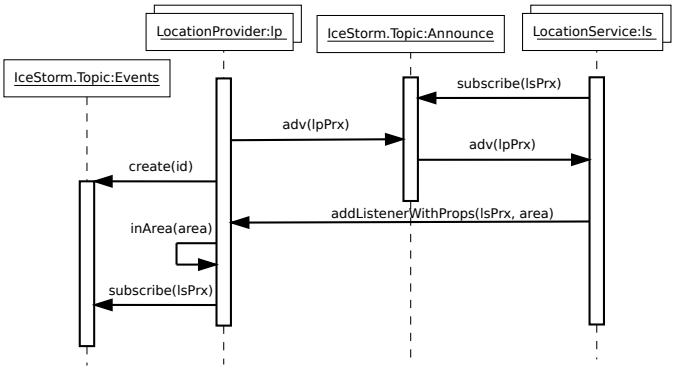

Fig. 2. Sequence diagram of LEP announcement

It should be noted that the discovery process uses properties in order to localize the LEPs. That is, the LS sends the "look up" using a property which represents a shape (in MLP format). However, it may use other properties to restrict the search.

C. Multiple technology identifiers

The LS can operate in two modes. In the first mode, the LS can dispatch the location events in a semantic way. In this mode the LS does not carry out a batching procedure. It can detect the event area, technology, and identifier and propagate them to the interested LEC -the LECs that subscribed via *trackingDevice* or *watchArea* methods-. In the second mode, the LS carries out a batching procedure in order to apply technology merging algorithms, and only propagates the estimation result. This operation mode presents a new handicap: it is necessary to create one common device identifier which supports the association of all the different technology identifiers.

In order to tackle this issue we define the *LS::DeviceProfile*. This class has an attribute which describes a dictionary that defines entries as pairs technology (as key) and identifier (as value). Thus the system can manage the different devices of each system user.

The aforementioned approximation can only partially solve the problem. It provides the necessary way to manage several technology identifiers. However, a mechanism that allows the retrieval of the device profile with a single technology identifier is also need. For this purpose we design the *LS::ProfileResolver* (Listing 3) which offers resolution methods, and the *LS::ProfileResolverAdmin* that offers administration methods such *bind*, *rebind*, *unbind* and *binds* (recover all the binding device profile).

Listing 3. LS::ProfileResolver definition

```
interface ProfileResolver {

 LS::DeviceProfile resolve(string tech, string id)
   throws NotFoundProfile;

 LS::DeviceProfileSeq resolveSeq(
  LS::StringDict techIdDict)
   throws NotFoundProfile;
};
```

Thanks to the *Profile Resolver*, the LS can carry out the batching procedures. In fact, if the LS works in this mode it must know the *Profile Resolver* service, since otherwise the LS will not start. In case of location events without associated devices, for example, motion sensor based LEP, these localization events are labeled with an "anonymous" label, and will be used by the system to improve the accuracy.

The proposed architecture provides some features which are very interesting from the distributed systems perspective, helping to deal with complexity and offering reliability and efficiency.

The composition mechanisms are complementary in the sense that LEPs have to implement both the announcement mechanism in order to notify the system about their presence, and the discovery mechanism so as to allow other services to find them. Additionally, the LS also has to listen to the announcements published in the system, in order to subscribe to LEPs. Previously, LEPs needs to be sought in the covered area.

Using both mechanisms the LSs are always updated. In this way, some fault tolerant methods can be implemented, since replication mechanisms can be implemented in a transparent way: if two LSs covering the same area are started at same time the will find the same LEPs and subscribe to them. Hence, both LSs will receive the same location events and therefore will have the same state.

Event filtering is an additional interesting system feature. So, a LS can subscribe to the different LSs deployed just to receive a specific type of events (e.g. estimated events). This behavior allows in general to reduce the number of location events propagated to higher levels, and, together with the federation flexibility it be can used to efficiently handle a complex hierarchy of spaces.

IV. Conclusions

This paper describes a multimodal indoor location system able to combine several positioning technologies so as to enhance location accuracy. To this end we propose an object-oriented distributed architecture which is characterized by its scalability, technology independence an flexibility.

There are two crucial aspects in the system. On the one hand, the use of a middleware places an abstraction layer in between the services and the location technologies. On the other hand, the use -and implementation- of standards increases the integration capabilities of the system, so as to be reused or to adopt other technologies.

The future works are focused on the implementation of a simulator -taking advantage of the modularity and independence of the different system components- in order to measure the algorithm location precision. We also are evaluating common-sense reasoning engines so as to estimate positions based on enhanced knowledge about how the world works.

References

[1] F. Seco, A. Jimenez, C. Prieto, J. Roa, and K. Koutsou, "A survey of mathematical methods for indoor localization," in *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, 2009, pp. 9 –14.

[2] D. Zhang, F. Xia, Z. Yang, L. Yao, and W. Zhao, "Localization technologies for indoor human tracking," *CoRR*, vol. abs/1003.1833, 2010.

[3] Inc. Cisco Systems, "Wi-fi location-based services 4.1 design guide," Tech. Rep., May 2008.

[4] P. Barsocchi, S. Lenzi, S. Chessa, and G. Giunta, "Virtual calibration for rssi-based indoor localization with ieee 802.15.4," in *Communications, 2009. ICC '09. IEEE International Conference on*, 2009, pp. 1 –5.

[5] T. Sanpechuda and L. Kovavisaruch, "A review of rfid localization: Applications and techniques," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on*, vol. 2, May 2008, pp. 769 –772.

[6] A. Salazar, "Positioning bluetooth reg; and wi-fi trade; systems," *Consumer Electronics, IEEE Transactions on*, vol. 50, no. 1, pp. 151 – 157, Feb. 2004.

[7] H. Hile and G. Borriello, "Positioning and orientation in indoor environments using camera phones," *Computer Graphics and Applications, IEEE*, vol. 28, no. 4, pp. 32 –39, 2008.

[8] T. Nishimura, Y. Nakamura, H. Tomobe, T. Kurata, T. Okuma, and Y. Matsuo, "Location estimation using auditory signal emitted and received by all objects," in *Networked Sensing Systems, 2007. INSS '07. Fourth International Conference on*, 2007, p. 295.

[9] M. Papandrea, "Multimodal ubiquitous localization: a gps/wifi/gsm-based lightweight solution," in *World of Wireless, Mobile and Multimedia Networks Workshops, 2009. WoWMoM 2009. IEEE International Symposium on a*, 2009, pp. 1 –3.

[10] O. Vinyals, E. Martin, and G. Friedland, "Multimodal indoor localization: An audio-wireless-based approach," in *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, 2010, pp. 120 –125.

[11] Open Mobile Alliance, "Mobile location protocol," version 1.2.1, 2004.

[12] M. Henning and M. Spruiell, *Distributed Programming with Ice*, Revision 3.4, 2010.

[13] Object Management Group, "Property service specification," version 1.0, 2000.

[14] F. Villanueva, D. Villa, M. Santofimia, F. Moya, and J. Lopez, "A framework for advanced home service design and management," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 3, pp. 1246 –1253, 2009.

[15] Open Geospatial Consortium Inc., "Opengis implementation standard for geographic information - simple feature access - part 2: Sql option," Candidate Version 3.1, 2010.