Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 109C (2017) 1110–1115

Procedia Computer Science

www.elsevier.com/locate/procedia

The International Workshop on Smart Cities Systems Engineering (SCE 2017)

# Beacon-based context-aware architecture for crowd sensing public transportation scheduling and user habits

Danilo Cianciulli[a], Gerardo Canfora[a], Eugenio Zimeo[a,*]

[a] *University of Sannio, Department of Engineering, Benevento, 82100 - Italy*

**Abstract**

Crowd sourcing and sensing are relatively recent paradigms that, enabled by the pervasiveness of mobile devices, allow users to transparently contribute in complex problem solving. Their effectiveness depends on people voluntarism, and this could limit their adoption. Recent technologies for automating context-awareness could give a significant impulse to spread crowdsourcing paradigms. In this paper, we propose a distributed software system that exploits mobile devices to improve public transportation efficiency. It takes advantage of the large number of deployed personal mobile devices and uses them as both mule sensors, in cooperation with beacon technology for geofecing, and clients for getting information about bus positions and estimated arrival times. The paper discusses the prototype architecture, its basic application for getting dynamic bus information, and the long-term scope in supporting transportation companies and municipalities, reducing costs, improving bus lines, urban mobility and planning.


Peer-review under responsibility of the Conference Program Chairs.

*Keywords:* crowdsourcing; mobile crowd sensing; smart transportation; beacons; context-awareness

## 1. Introduction

Crowdsourcing is a relatively recent concept that generally means the act of taking a task usually performed by a single agent and outsourcing it by making an open call to a large group of people[1]. However, it is hard to give a closed definition, since its meaning goes under constant evolutions as new applications appear, causing lack of consensus and a certain semantic confusion[2]. As an instance, the wide spread of smartphones enables the so-called *mobile crowd sensing*[4], where users transparently contribute to complex problem solving[3]. The continuous evolution of technology makes it available small and cheap devices that ease the implementation of context awareness, reducing the burden on users (who have not to explicitly provide data) and promoting a wider adoption of sensor-based mobile apps. The *context* is any information that can be used to characterize the situation of an entity[6], that is a person, place, or object relevant for the application. It can be defined as the set of variables that characterize the *external state* of an application, and include every attribute that characterizes a user and/or the (smart) environment the application interacts with[7].

* Corresponding author. Tel.: +39 0824 305538 ; fax: +39 0824 50552.
*E-mail address:* eugenio.zimeo@unisannio.it


Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2017.05.451

In line with this evolution, we discuss the design principles for implementing a distributed software system that exploits mobile crowd sensing and context-awareness to improve public transportation services, usability, and cost effectiveness. The key idea is to take advantage of the large number of deployed personal devices and to use them as mule sensors of a peer-to-peer infrastructure where the same devices can ask for information about bus positions and estimated arrival times. In particular, by using common smartphones and cheap Bluetooth-based devices, our work aims at avoiding complex and expensive hardware infrastructures, giving citizens the opportunity to help making their cities smarter.

The core solution mainly provides citizens with the ability to get current bus positions, taking into account dynamic environment conditions, but also aims at supporting transportation companies and municipalities in reducing costs, improving bus lines and urban planning. The main goals our system aims at are:

- *Public transportation monitoring*: that is collecting data such as usage statistics of buses and stations, time series with current position of buses, and anomalies detection;
- *Citizens profiling*: that is collecting information about users and devices using the system;
- *Support for user's travel planning*: the ability to query the system for information about the current position of specific buses, average travel times between stations, and estimated times of arrival.

The goals above pave the way for a large variety of long-term goals that may include (a) *public transportation optimization*, that is the possibility to implement actions that aim at reducing costs or improving service performances by identifying under- or over-used routes, (b) *citizens' habits detection*, that is mining information about habits to be used in various contexts (e.g. targeted advertising), and (c) *travel recommendations*, that is helping citizens to schedule their routes with hints about best-fitting paths and alternative options in case of need.

The remainder of the paper is organized as follows: in section 2, we explore related work about mobile-based crowdsourcing and context-aware applications; in section 3, we present our architecture for crowdsourcing public transportation information; in section 4, we discuss the main problems related to the implementation of context detection; finally, in section 5, we conclude the paper and outline future work.

## 2. Related work

Mobile crowd sensing enables numerous large-scale applications[4]. Papers on this topic are typically distinguished in three main categories, namely environmental, infrastructure, and social[8], with both implicit and explicit human participation[5]. The work presented in this paper falls in the infrastructure applications category that, generally, involves the measurement of phenomena that are related to public infrastructure, such us measuring traffic congestion, road conditions, parking availability, outages of public works and real-time transit tracking[8]. Several examples can be found when speaking of mobile crowd sensing applied to transportation, such as parking guidance systems[10], or ITS (Intelligent Transportation Systems) oriented solutions, in which the goal is to replace expensive sensors on vehicles and street infrastructures with mobile crowdsourcing, yet enabling safe driving experiences enriched with the awareness of car accidents and traffic information[9]. In mobile crowd sensing applied to smart transportation, Bluetooth-based beacons can be used to make mobile devices aware of the surrounding environment. This technology, thanks to its low costs, enables a wide spectrum of applications[12], mainly in the context of indoor localization systems, such as cell-based localization enabled by intelligent placement[13], or proximity-aware systems for the interaction with nearby objects, such as artworks in museums[14]. However, despite several mobile crowd sensing solutions for public transportation have been proposed[15,16], no relevant work exists that makes use of Bluetooth-based beacon technologies.

## 3. Beacon-based context-awareness

The approach that we propose in this paper is based on three main concepts: *crowdsourcing*, *mobility* and *context-awareness*. The first concept is tied to the wide availability of personal devices that can be used as sensor nodes; the second one, mobility of people, transforms these devices in mules (mobile sensors) able to collect information while moving; the third one aims at automating the acquisition of context information through a cheap technology able to assign an electronic identity to physical objects that can be broadcasted to mobile devices entering a specific range.

According to the main concepts above, our solution is delivered through a dedicated App for mobile devices, connected with a remote server for coordination and data collection by means of an API (e.g. a RESTful API over HTTP protocol). Fig. 1 shows a high-level abstraction of components and relations. By using the App, citizens implicitly signal the position of buses when they are in their proximity, or aboard. This is made by a background service of the App that is able to detect the presence of a bus and then to signal its geographic coordinates to a server by means of a well-defined API. The same App may be used to query the server for current positions of buses and stations a user is interested in.

Thus, the client App has two independent main features that might be separated in two different applications: 1) it serves as a sensor for the system that needs information about current buses positions and travel statistics (from now on the *sensor App*), and 2) it is used by citizens to seek for buses positions, average travel times, estimated times of arrival, help on planning, and so on (from now on the *query App*).

One important point of the system is the context identification: (1) how can we detect the presence of buses and stations, and (2) how can we identify them once detected? An answer to both questions is to use Bluetooth-based beacons, which allow Bluetooth devices to broadcast and receive tiny and static pieces of data within short distances. It consists of two parts: a broadcaster (the beacon itself), that is always advertising its presence passing its ID through a radio packet, and a receiver, in our case the sensor App, that detects such packets. The identities of buses and stations are encapsulated in IDs that are transmitted via Bluetooth. However, we need to distinguish between beacons received from stations and those received from buses. There are essentially two alternatives: (1) query the server to know if a given beacon is related to a station or to a bus, or (2) use beacon-specific data fields as a discriminating factor.

According to Apple guidelines for iBeacons [17,18], the beacon payload contains editable static data such as a 16 byte *UUID* field that represents the particular application, and 2 bytes *Major* and *Minor* fields that allow to categorize the purpose of the beacon. These considerations lead us to use the $2^{nd}$ choice, and reserve the *UUID* for identifying the public transportation provider, while using the *Major* as bus/station discriminating number, and the *Minor* as bus/station identifier.

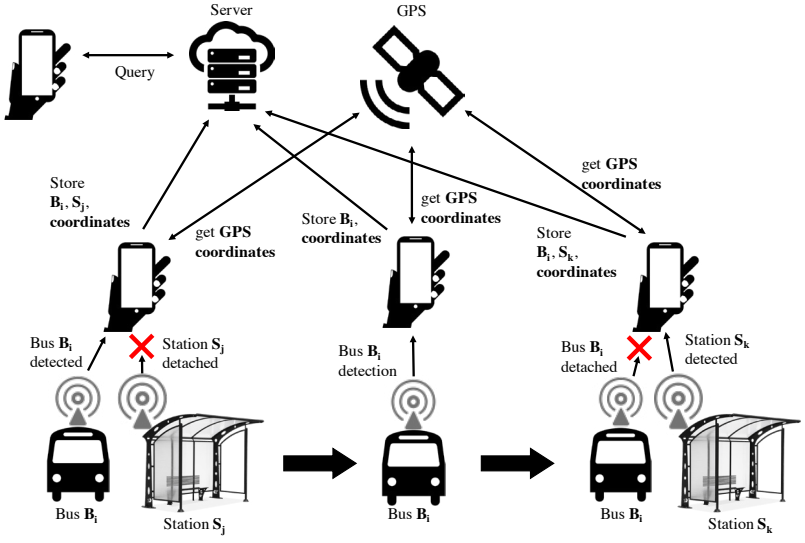
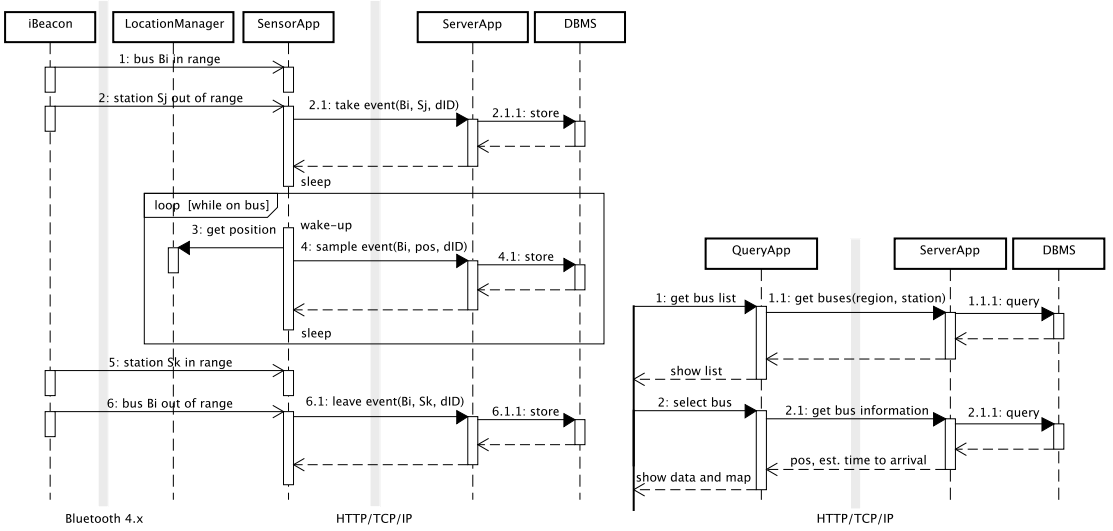



Fig. 1. The App can be used in two ways. (1) In the upper-left corner it works as a query application that is used as a support for citizens to retrieve information such as current buses positions and travel times. (2) In middle and bottom sections, it is used as a sensor application that is able to detect when a citizen takes a bus $B_i$ from a station $S_j$ (left), to signal current bus $B_i$ position during the travel (center), and to detect when the citizen leaves the bus $B_i$ at a station $S_k$ (right).

Fig. 2(a) shows a high-level conceptual sequence diagram that depicts how the sensor App interacts with Bluetooth-based beacons and the server. When the App detects a beacon $B_i$ associated with bus $i$ and, contextually, loses beacon $S_j$ associated with station $j$ (messages 1 and 2), this means that the user boarded the bus identified by beacon $B_i$ at the station $S_j$. As a consequence, a *take event* message is sent to the server, containing information about the station, the bus, and the device (message 2.1). The App is not required to send the bus position immediately, since this information is directly related to the station position; thus it sleeps for a given time interval.

After this sequence of messages, we assume that the citizen is on board, therefore, when the App wakes-up, it periodically sends the current position by means of *sample event* messages (message 4): this allows the system to know, time to time, where the bus is. Even if we assumed that the device makes use of GPS information, here we used a more abstract *Location Manager*, that is a SO-specific component that takes care of retrieving the current positions from different sources (GPS, Wi-Fi networks, phone cells, etc.) for a higher precision (message 3). Finally, when the App detects a beacon $S_k$ associated with station $k$ and, contextually, loses beacon $B_i$ of the bus $i$ (messages 5 and 6), then it means that the user got off bus $i$ at the station $k$. Therefore, a *leave event* message is issued to the server (message 6.1) containing information about the station, the bus, and the device.




(a) Interactions between beacons, sensor App, and server. (b) Query App use case example.

Fig. 2. High-level conceptual sequence diagrams. Note how they are vertically divided by communication channels.

On the other hand, Fig. 2(b) shows a high-level conceptual sequence diagram that depicts a possible use case of the query App. In this example, the query App looks for buses within a given geographic region or that reaches a given station, thus, after a selection, it shows where the bus is along with the estimated time of arrival for a given station.

We mainly focus on the sensor side of the system, more specifically on the part of the server that manages event messages, since the business logic of the whole system is based on data collected by means of event messages depicted in Fig. 2(a).

The prototypes of server and sensor Apps make use of a RESTful API to communicate each other. On the one hand, the server is developed with mixed Java frameworks (Spring and Java EE), and exposes a simple Web-based GUI for management purposes, such as registering buses and stations along with beacon UUIDs; on the other hand, the sensor App is an Andoid-specific application. The whole system follows a three-tier architecture with a distributed application logic and it is ready for performance scaling, but it has several requirements that must be met: (a) Bluetooth and GPS modules are required on smartphones, and (b) users need a billing plan with their telco providers that enables Internet access on devices. These requirements are commonly met by any modern device and user profile, thus they are not considered as limitations.

## 4. Context detection

In this section, we address the main problem of correctly detecting bus and user contexts with the help of both client and server logics. Fig. 3 shows a simplified domain data model. It is centred around the concept of *Device* that has an optional *User* associated with (users, in fact, are not required to be registered in order to use the system). A *Device* resource is created when a new one is detected through the device ID (its IMEI in our case) received via the API, to which a *User* resource is associated when a user logs in with.

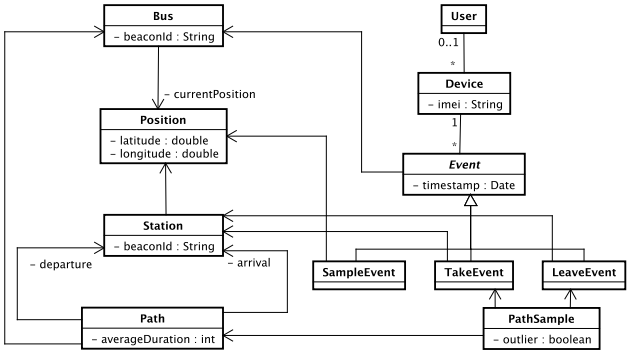



Fig. 3. UML class diagram that shows the simplified domain data model.

Three types of events have been identified and modelled as a specialization of an abstract *Event* that has references to *Bus* and *Device*. They are *TakeEvent*, which is created when the sensor App signals that a user has boarded a bus, *SampleEvent*, which is created when the sensor App signals that a user is in the proximity or on board of a bus, and *LeaveEvent*, which is created when the sensor App signals that a user has got off a bus. Thus, *SampleEvent* takes into account the position, while *TakeEvent* and *LeaveEvent* refer to a station that is the departure for the former, and the arrival for the latter.

Crucial for the model are *Station* and *Bus* entities: both are related to *Position* entity, but for the former it is a static information (since stations do not move), while for the latter it is a dynamic information that represents the current position, which is updated with a spatial average when any of the three events takes place. In this domain data model, any route that has distinct departure, arrival, and bus defines a distinct *Path*. A path is detected when the same device generates a take event followed by a leave event for the same bus: this information is captured by the *PathSample* entity. When a path sample is detected, then we check if a *Path* with the same departure, arrival, and bus exists: if so, we update the path information (such as average travel time) with the new data, otherwise we create a new path where we set the average travel time as the time difference between the related leave and take events.

To adapt the average travel time for *Path* entities to changes in traffic conditions, we make use of an exponentially weighted moving average (EWMA) updated time to time as new path samples are detected:

$$EWMA_k = (1 - \alpha) \cdot EWMA_{k-1} + \alpha \cdot sample_k$$

where $sample_k$ is the sample we are using to update the average, and $\alpha$ is a smoothing factor in the range $(0, 1)$ that allows to give more or less importance to newer samples.

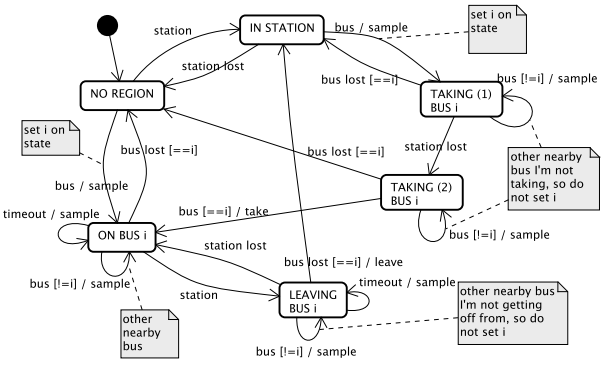



Fig. 4. Client-side context detection: state diagram, where *bus* and *station* transitions represent the detection events of buses and stations beacons, whereas *bus lost* and *station lost* transitions identify beacons detachments.

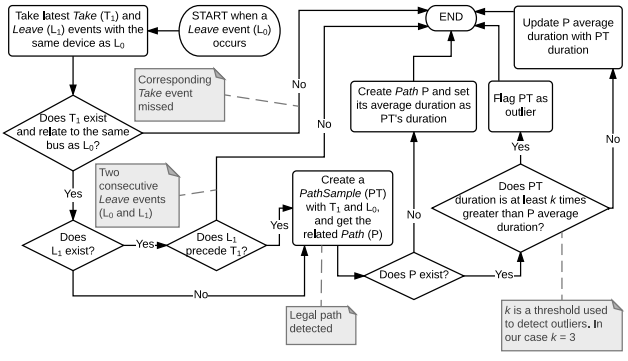



Fig. 5. Server-side context detection: flowchart showing the algorithm used to update average times, and to detect legal paths and outliers as new Leave events occur.

Fig. 4 shows the state diagram of the sensor App, useful to detect user and bus contexts, and that is responsible of correctly generate coarse-grained events (*Take*, *Sample*, and *Leave*) based on fine-grained ones (the detection or the loss of beacons). When Take, Leave and Sample events are generated and communicated to the server, this stores them and tries to detect a path instance each time a new Leave event occurs. This leads to the need of identifying

illegal paths that may occur when some events are not captured or have been lost for any reason. Fig. 5 shows the algorithm used to detect legal path samples that are then used to update the average travel time.

## 5. Conclusions

The paper presented a system to transparently capture information such as users that take and get off buses at given stations, as well as historical time series of buses positions, enabling interesting big data analysis. Currently, the system is able to support users by informing them about bus positions and, when enough data will be collected, accurate travel information based on statistics, such as estimated times and recommendations. We are also able to relate captured events to people that bind their identity to devices they are using, thus enabling habits detection.

The proposed approach has several notable advantages: (a) companies and municipalities have to sustain almost no infrastructure costs for implementation and maintenance, and (b) the system is able to learn by itself as new routes are introduced. Human intervention is only required for beacons placement and their association to buses and stations.

To stimulate a wide adoption of the application for collecting useful usage data, some actions are needed: (1) give users a valuable support in order to greatly facilitate their experience with public transportation, and (2) make use of incentive techniques [19,20], which we will address in future work.

## Acknowledgements

This work has been partially supported by the GAUSS national research project (MIUR, PRIN 2015, Contract 2015KWREMX), and POR Regione Campania FSE 2007/13, 2014/20 - CUP F82I15000470002.

## References

1. Howe, Jeff. “Crowdsourcing: How the power of the crowd is driving the future of business” *Random House*, 2008.
2. Estells-Arolas, Enrique, and Gonzlez-Ladrn-De-Guevara, Fernando. “Towards an integrated crowdsourcing definition.” *Journal of Information science* 38.2 (2012): 189-200.
3. Chatzimilioudis, Georgios, et al. “Crowdsourcing with smartphones.” *IEEE Internet Computing* 16.5 (2012): 36-44.
4. Ma, Huadong, Dong Zhao, and Peiyan Yuan. “Opportunities in mobile crowd sensing.” *IEEE Communications Magazine* 52.8 (2014): 29-35.
5. Guo, Bin, et al. “From participatory sensing to mobile crowd sensing.” *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on.* IEEE, 2014.
6. Abowd, Gregory D., et al. “Towards a better understanding of context and context-awareness.” *International Symposium on Handheld and Ubiquitous Computing.* Springer Berlin Heidelberg, 1999.
7. Furno, Angelo, and Zimeo, Eugenio. “Context-aware composition of semantic web services.” *Mobile Networks and Applications* 19.2 (2014): 235-248.
8. Ganti, Raghu K., et al. “Mobile crowdsensing: current state and future challenges.” *IEEE Communications Magazine* 49.11 (2011).
9. Ali, Kashif, et al. “Crowdits: Crowdsourcing in intelligent transportation systems.”*Wireless Communications and Networking Conference (WCNC), 2012 IEEE.* IEEE, 2012.
10. Chen, Xiao, et al. “Crowdsourcing for on-street smart parking.” *Proceedings of the second ACM international symposium on Design and analysis of intelligent vehicular networks and applications.* ACM, 2012.
11. Misra, Aditi, et al. “Crowdsourcing and its application to transportation data collection and management.”*Transportation Research Record: Journal of the Transportation Research Board*2414 (2014): 1-8.
12. Gast, Matthew S. “Building applications with IBeacon: proximity and location services with bluetooth low energy.” *O’Reilly Media, Inc.*, 2014.
13. Chawathe, Sudarshan S. “Beacon placement for indoor localization using bluetooth.” *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on.* IEEE, 2008.
14. He, Zhiqiang, et al. “A proposal of interaction system between visitor and collection in museum hall by iBeacon.” *Computer Science & Education (ICCSE), 2015 10th International Conference on.* IEEE, 2015.
15. Farkas, Kroly, et al. “Participatory sensing based real-time public transport information service.” *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on.* IEEE, 2014.
16. Heiskala, Mikko, et al. “Crowdsensing-based transportation services - An analysis from business model and sustainability viewpoints.” *Research in Transportation Business & Management* 18 (2016): 38-48.
17. “Proximity Beacon Specification.” *Apple Developers.* https://developer.apple.com/ibeacon/
18. “Getting Started with iBeacon.” *Apple Developers.* https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf
19. Yang, Dejun, et al. “Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing.” *Proceedings of the 18th annual international conference on Mobile computing and networking.* ACM, 2012.
20. Jaimes, Luis G., et al. “A survey of incentive techniques for mobile crowd sensing.” *IEEE Internet of Things Journal* 2.5 (2015): 370-380.