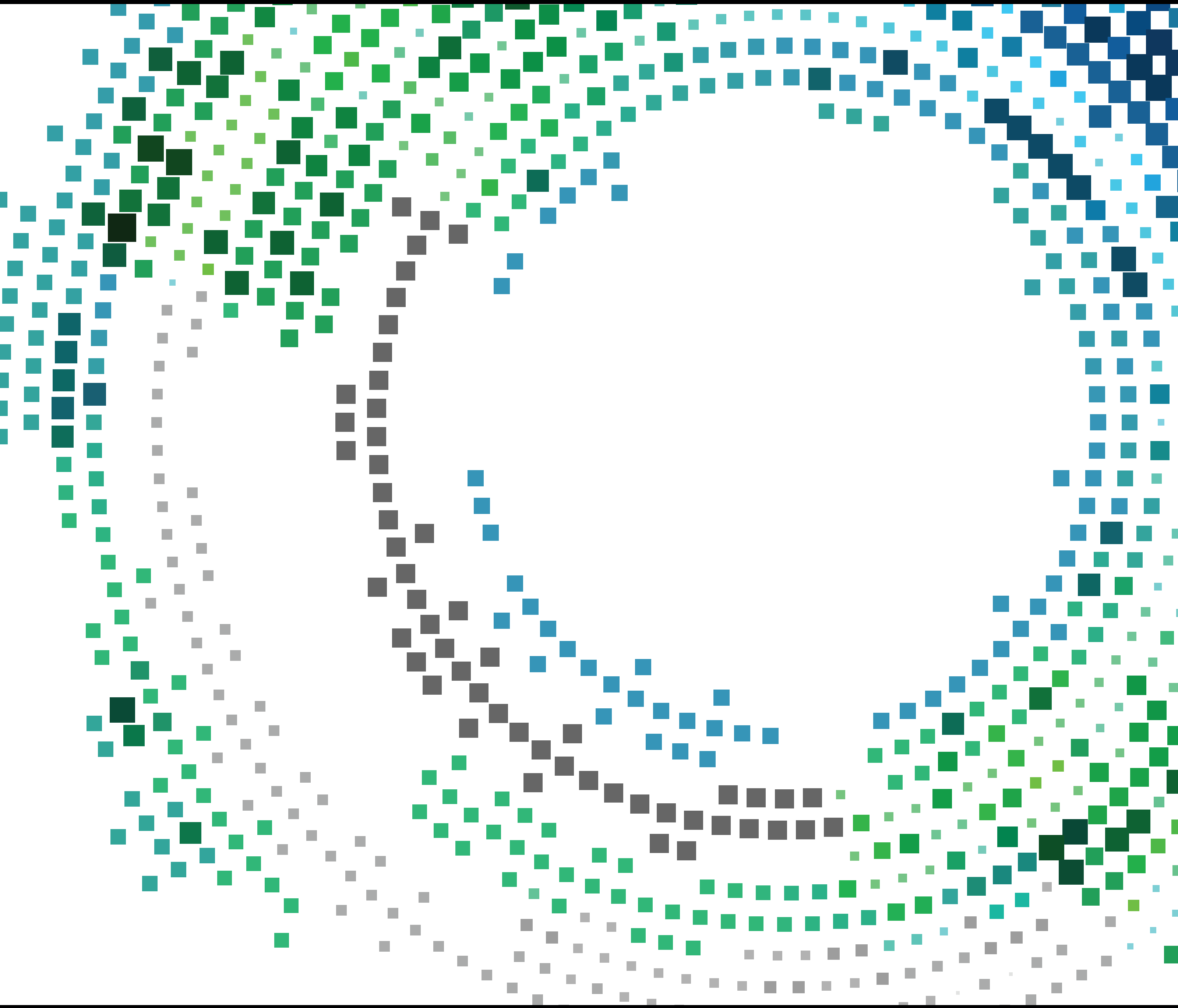
Mobile Information Systems

Crowdsensing and Vehicle-Based Sensing

Guest Editors: Carlos T. Calafate, Celimuge Wu, Enrico Natalizio, and Francisco J. Martínez



Crowdsensing and Vehicle-Based Sensing

Mobile Information Systems

Crowdsensing and Vehicle-Based Sensing

Guest Editors: Carlos T. Calafate, Celimuge Wu, Enrico Natalizio, and Francisco J. Martínez

Contents

Crowdsensing and Vehicle-Based Sensing
Carlos T. Calafate, Celimuge Wu, Enrico Natalizio, and Francisco J. Martínez
Volume 2016, Article ID 2596783, 2 pages

A Survey on Smartphone-Based Crowdsensing Solutions
Willian Zamora, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni
Volume 2016, Article ID 9681842, 26 pages

SenSafe: A Smartphone-Based Traffic Safety Framework by Sensing Vehicle and Pedestrian Behaviors
Zhenyu Liu, Mengfei Wu, Konglin Zhu, and Lin Zhang
Volume 2016, Article ID 7967249, 13 pages

A Service-Oriented Approach to Crowdsensing for Accessible Smart Mobility Scenarios
Silvia Mirri, Catia Prandi, Paola Salomoni, Franco Callegati, Andrea Melis, and Marco Prandini
Volume 2016, Article ID 2821680, 14 pages

Human Mobility Modelling Based on Dense Transit Areas Detection with Opportunistic Sensing
Fernando Terroso-Sáenz, Mercedes Valdes-Vela, Aurora González-Vidal, and Antonio F. Skarmeta
Volume 2016, Article ID 9178539, 15 pages

A Safety Enhancement Broadcasting Scheme Based on Context Sensing in VANETs
Chen Chen, Hongyu Xiang, Canding Sun, and Qingqi Pei
Volume 2016, Article ID 2630439, 17 pages

Representation Learning from Time Labelled Heterogeneous Data for Mobile Crowdsensing
Chunmei Ma, Qing Zhu, Shuang Wu, and Bin Liu
Volume 2016, Article ID 2097243, 10 pages

AirCache: A Crowd-Based Solution for Geoanchored Floating Data
Armir Bujari and Claudio Enrico Palazzi
Volume 2016, Article ID 3247903, 12 pages

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 2596783, 2 pages
http://dx.doi.org/10.1155/2016/2596783

Editorial
Crowdsensing and Vehicle-Based Sensing

Carlos T. Calafate,[1] Celimuge Wu,[2] Enrico Natalizio,[3] and Francisco J. Martínez[4]

[1]*Department of Computer Engineering (DISCA), Universitat Politècnica de València, Camino de Vera S/N, 46022 Valencia, Spain*
[2]*Graduate School of Information Systems, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan*
[3]*Heudiasyc UMR CNRS 7253, Sorbonne Universités, Université de Technologie de Compiègne, CS 60319, 60203 Compiègne Cedex, France*
[4]*Computer Science and System Engineering Department, University of Zaragoza, Escuela Universitaria Politécnica de Teruel, Ciudad Escolar s/n, 44003 Teruel, Spain*

Correspondence should be addressed to Carlos T. Calafate; calafate@disca.upv.es

Received 24 October 2016; Accepted 25 October 2016

Improvements in terms of smart device capabilities and of communication technologies allowed crowdsensing solutions to emerge as a powerful strategy to revolutionize environment sensing, becoming one of the key elements of future smart cities. In particular, smartphones, smartwatches, and other personal gadgets are now endowed not only with significant computing power and different wireless interfaces, but also with an increasing number of sensors able to provide useful information about the user environment, especially when the user is moving around a city. If vehicular mobility is adopted, the sensing capabilities can be further increased by connecting smart devices to vehicles using the On Board Diagnostic Interface (OBD-II) or provided directly by smart vehicles through vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communications. In the future, with the gradual introduction of autonomous vehicles and unmanned aerial vehicles, many more sensing alternatives are expected.

In parallel to these developments, network infrastructure has experienced significant improvements in recent years in terms of both coverage and performance which, combined with advances in cloud computing, allows transmitting, storing, and processing large amounts of data in an efficient manner. Altogether, the potential for generating and analyzing huge amounts of data is remarkable, being able to provide unprecedented information with high levels of spatial and time resolution about any event of interest occurring in a city.

The purpose of this special issue is to address recent advances on mobile sensing, with particular emphasis on novel applications and architectures for crowdsensing and vehicle-based sensing.

The articles contained in the present issue include both reviews and research studies focused on data sensing and processing in situations where either pedestrians or vehicles act as mobile sensors.

As an introduction to the topic, the paper by W. Zamora et al. entitled "A Survey on Smartphone-based Crowdsensing Solutions" provides a survey of smartphone-based crowdsensing solutions that have emerged in the past few years, focusing on different works published in top-ranked journals and conferences. To properly analyze these previous works, they define a reference framework that allows classifying the different proposals under study. Globally, the survey provides useful insight into the broad scope of the crowdsensing area, being of interest to both experts and novices.

Analyzing and optimizing mobility in the urban domain is one of the most relevant contributions that mobile crowdsensing can offer to smart cities. In particular, the ever-increasing capabilities of smartphones and vehicle-mounted devices pave the way for gaining meaningful insight into urban mobility. In this context, the work by F. Terroso-Sáenz et al. entitled "Human Mobility Modelling Based on Dense Transit Areas Detection with Opportunistic Sensing" introduces a novel approach to leverage such paradigm, allowing composing a map of Dense Transit Areas (DTAs) within a city representing some of its mobility features while protecting the privacy of users. Besides vehicles, pedestrian and multimodal paths would also benefit greatly from context data, as well as the information about the whole experience of traveling and wandering the city, including travel planning and payments. The work by S. Mirri et al. entitled "A Service-Oriented Approach to Crowdsensing for Accessible Smart Mobility Scenarios" introduces a prototype of the infrastructure and overall architecture proposed to meet these challenges, describing some of the services that can be provided: path recommendations for wheelchair users and for elderly persons.

In addition to mobility optimization, traffic safety is also a critical issue that should be addressed to minimize the chances of accidents. In this context, the work by Z. Liu et al. entitled "SenSafe: A Smartphone-Based Traffic Safety Framework by Sensing Vehicle and Pedestrian Behaviors" proposes a driving behavior detection mechanism that relies on smartphones to sense events in the proximity and provide alerts to drivers. Their proposal also includes a low-overhead approach for fast data broadcasting, along with a collision estimation algorithm to trigger warnings when dangerous situations are detected. Since broadcasting is a key element for security and context awareness in vehicular ad hoc environments by simplifying direct vehicle-to-vehicle communications, effective information exchanging becomes critical. The work by C. Chen et al. entitled "A Safety Enhancement Broadcasting Scheme Based on Context Sensing in VANETs" addresses such problem by proposing a congestion control scheme for vehicular networks that can dynamically adapt to variable channel occupation to maximize the network throughput while avoiding to congest the channel, thereby improving network throughput, packet delivery ratio, and transmission delay.

Finally, the efficient handling of data itself gains utmost importance in the crowdsensing context, including both data acquisition and dissemination. Concerning the latter, novel paradigms have recently emerged, as is the case of the floating data paradigm which aims at enabling a floating data network in a distributed and collision-free way. The work by A. Bujari and C. E. Palazzi entitled "AirCache: A Crowd-Based Solution for Geoanchored Floating Data" is an example of such a solution, whose aim is to guarantee data availability in an Area of Interest while reducing the data access costs at the network edges. In particular, it relies on replication among passing-by or stationary users, in addition to a node election strategy, to improve the global energy/memory consumption needed to maintain the data floating.

With respect to the data gathering process, especially when using smartphones, it usually involves different data types due to the different granularity, multiple sensor sources, and time labelling. Such heterogeneity and time dependencies introduce new challenges in the data analysis process. The work of C. Ma et al. entitled "Representation Learning from Time Labelled Heterogeneous Data for Mobile Crowdsensing" addresses this challenge by proposing a new representation learning method for heterogeneous data with time labels that allows extracting typical features using deep learning. In their work they exemplify the applicability of their proposed method by differentiating between two mobile activities using smartphone sensors: walking versus cycling and driving versus taking the bus.

Overall, we hope that this special issue is able to shed some light on the major developments in the area of crowdsensing and vehicle-based sensing and attract the attention of the scientific community to undertake further investigations leading to the rapid deployment of such novel solutions.

Acknowledgments

We would like to express our appreciation to all the authors for their novel contributions and the reviewers for their support and constructive critiques in making this special issue possible.

Carlos T. Calafate
Celimuge Wu
Enrico Natalizio
Francisco J. Martínez

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 9681842, 26 pages
http://dx.doi.org/10.1155/2016/9681842

Review Article

A Survey on Smartphone-Based Crowdsensing Solutions

Willian Zamora, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni

Department of Computer Engineering, Universitat Politècnica de València, Camino de Vera S/N, 46022 Valencia, Spain

Correspondence should be addressed to Willian Zamora; wilzame@posgrado.upv.es

Received 19 May 2016; Accepted 16 October 2016

Academic Editor: Dik Lun Lee

In recent years, the widespread adoption of mobile phones, combined with the ever-increasing number of sensors that smartphones are equipped with, greatly simplified the generalized adoption of crowdsensing solutions by reducing hardware requirements and costs to a minimum. These factors have led to an outstanding growth of crowdsensing proposals from both academia and industry. In this paper, we provide a survey of smartphone-based crowdsensing solutions that have emerged in the past few years, focusing on 64 works published in top-ranked journals and conferences. To properly analyze these previous works, we first define a reference framework based on how we classify the different proposals under study. The results of our survey evidence that there is still much heterogeneity in terms of technologies adopted and deployment approaches, although modular designs at both client and server elements seem to be dominant. Also, the preferred client platform is Android, while server platforms are typically web-based, and client-server communications mostly rely on XML or JSON over HTTP. The main detected pitfall concerns the performance evaluation of the different proposals, which typically fail to make a scalability analysis despite being critical issue when targeting very large communities of users.

1. Introduction

The use of mobile phones has experienced a significant increase in the past decade. In fact, according to the 2015 ITU World Telecommunications report [1] for 2015, the ratio of Cellular phone subscriptions was 97%, which represents 7084 million subscribers in the world. In addition, this subscriber increase is reflected in the technological advantages offered by mobile devices. Furthermore, mobile devices available nowadays have a high computational power and include different communication technologies (e.g., WiFi, 4G, and Bluetooth) and have multiple embedded sensors (GPS, gyroscope, accelerometer, microphone, and camera, among others). This technological growth, together with the increasing number of subscribers, has caused the community of researchers and developers to create different applications based on smartphones as sensors.

Pioneering research anticipated this arising of new applications, describing them as "participatory sensing" [2] or "people-centered sensing" [3]. In both cases, the idea is that the user should be able to gather data anywhere, anytime, by making use of mobile sensor devices for information retrieval, processing, and sharing. Later on, researchers considered this new paradigm as a subtype of crowdsensing denoted as "mobile phone sensing" [4].

Mobile phone sensing benefits from the processing and communication capabilities of available smartphones which, combined with one or more sensors, become an enabling technology to support different types of applications. Moreover, mobile crowdsensing relies on a large number of participants to collect data from the environment through its integrated sensors and, after capturing the data, these are sent to a server to perform data mining tasks including data fusion, analysis, and information dissemination. Typically, sensors that register participant information (e.g., location, movements) and environmental data (e.g., images, sounds) are very common. On top of that, some solutions use external sensors, which are integrated into the mobile solution through its communication interfaces, including sensors for environmental pollution and health monitoring. In this sense, mobile crowdsensing provides new perspectives for improving living conditions in our digital society. A general example of mobile crowdsensing solution is shown in Figure 1.

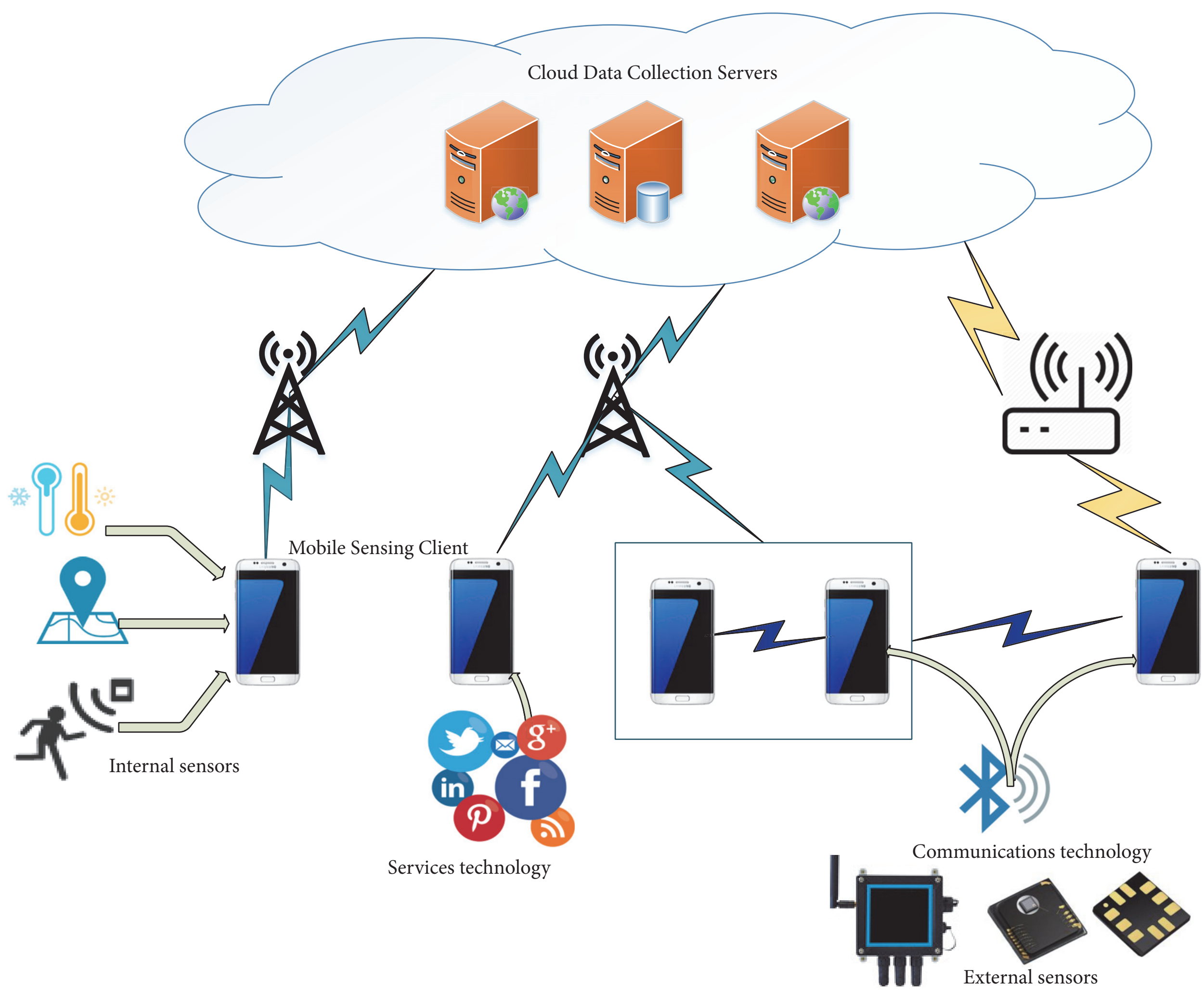


Figure 1: Generic structure of crowdsensing solutions.

Concerning mobile crowdsensing applications, Figure 2 shows that they have experienced a significant increase in the last 5 years. Specifically, researchers have focused their efforts on various areas including environment monitoring [5–8], transportation and urban sensing [9–13], healthcare [14–18], social issues [19–23], and others [24–28]. The different crowdsensing proposals available are characterized by having different designs and involve different architectural levels. For instance, some authors propose solutions they call framework, middleware, or system, among other terms. No matter which term is used, these solutions can have a global approach (full architecture) or only specify a subset of the architecture by describing one or more components.

The existence of a high number of proposals, and the absence (to date) of a survey that properly organizes such information, has led us to write this paper. In this work, we start by proposing a reference client-server architecture where the sensing device is the Mobile Sensing Client (MSC) and the server is the Cloud Data Collection Server (CDCS). Our idea is to propose an architecture that is generic and

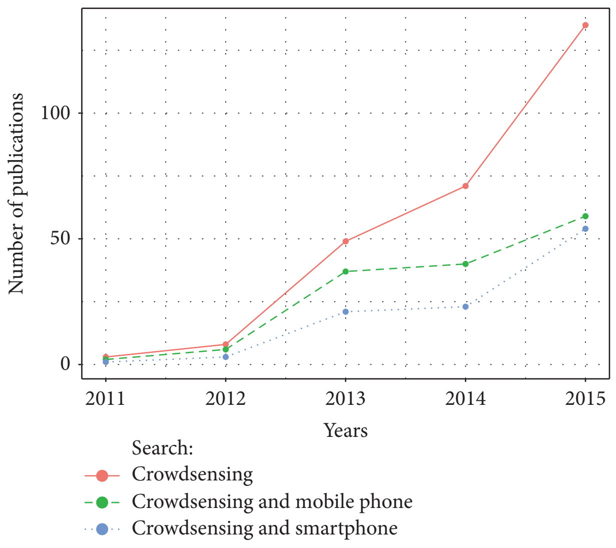


Figure 2: Number of crowdsensing-related proposals in the past 5 years.

flexible enough to accommodate any existing solution. With this in mind, we have identified whether the main contribution of each publication focuses on the client, on the server, on the transmission, or on some specific component. For client-side proposals, we determined whether their main contribution is in the processing or the data capture element, discriminating between sensor and data administration. For the server-side proposals, we determined the actual contributions in terms of mobile sensing tasks, dimensional analysis of data, and cloud services. Finally, we determined the contribution of those proposals focusing on data communications.

The paper is organized as follows: in the next section we present some related surveys on this topic. In Section 3 we provide an overview of the proposed architecture. Then, in Section 4, we make a detailed analysis of the proposed architecture. Section 4 provides the actual survey results, detailing the contributions made at the client and server sides, as well as to the end-to-end communications approach. Open research issues are then discussed in Section 5. Finally, in Section 6, we present our conclusions and future work.

2. Related Works

In recent years, the rise of solutions in the field of mobile crowdsensing is attracting huge interest because of the large amount of data that sensors can provide when relayed via mobile phones. Nonetheless, there are quite few surveys that actually study and summarize the many existing proposals. Below we proceed to describe briefly the different surveys found in the literature according to the chronological order of their publication.

Lane et al. [4] made a pioneer survey addressing the use of mobile phones as sensors, analyzing the significant progress mobile phones have experienced in order to incorporate multiple sensors. Also, they describe various existing proposals according to certain algorithms, applications, and systems developed to date. Similarly, they describe some proposals grouped by areas such as transportation, environmental monitoring, and health. They also propose an architecture composed of three different elements dedicated to sensing (mobile phone), learning (analysis), and informing (shared data).

Ganti et al. [29] proposed the term mobile crowdsensing (MCS) and described a reference architecture. This survey only showed a few proposals categorized as participatory (users are involved) and opportunistic (users are not involved). Additionally, it identifies some characteristics that influence these solutions such as limited resources, privacy and security, data integrity, data aggregation, and data analytics.

Khan et al. [30] present a taxonomy where they differentiate between personnel sensing, social sensing, and public sensing. This classification is performed from the point of view of participatory and opportunistic sensing.

Zhang et al. [31] propose an approach which characterizes the various crowdsensing proposals in four stages: task creation, task assignment, individual task execution, and crowd data integration. These features are described as what, when, where, who, and how (4W1H).

Table 1: Crowdsensing surveys.

Solutions	Year	# publications reviewed
Lane et al. [4]	2010	25
Ganti et al. [29]	2011	13
Khan et al. [30]	2013	43
Calabrese et al. [38]	2014	26
Zhang et al. [31]	2014	15
Zhang et al. [32]	2015	32
Jaimes et al. [33]	2015	22
Guo et al. [37]	2015	41

More recently, both Zhang et al. [32] and Jaimes et al. [33] proposed a classification based on incentive mechanisms for mobile crowdsensing. The former classified incentives into three categories: entertainment, services, and economic. The latter used incentive mechanisms as metrics to evaluate crowdsensing and introduced a tree-level taxonomy for crowdsensing incentive mechanisms. In the same context, different authors [34–36] propose incentive mechanisms that rely on auction techniques for evaluating quality awareness in the mobile crowdsensing context. In particular, the first uses combinatorial auction models, while the second extends that work by introducing more fine-grained techniques; concerning the third work, it proposes a framework that integrates incentives, data aggregation, and data perturbation mechanisms. However, they did not propose any reference architecture, addressing solely the taxonomy of their proposals and the algorithms supporting these proposals.

Guo et al. [37] propose a new sensing paradigm called Mobile Crowd Sensing and Computing (MCSC) that empowers ordinary citizens to contribute data sensed or generated from their mobile devices, aggregating and fusing the data in the cloud for crowd intelligence extraction and human-centric service delivery. This paper proposes a taxonomy and a reference architecture for MCSC. In the taxonomy the proposals are classified as mobile sensing (user involvement, data contribution, user awareness, and sampling), crowd data collection (networking, incentives, and scale), crowdsourced data processing and intelligence extraction (processing architecture, intelligence, purpose, data mining, and data quality), hybrid human-machine system, and security and privacy. With regard to the architecture, the proposals presented in this paper are divided into several levels: crowdsensing, data collection, data processing, and applications.

As it quickly becomes evident through this brief state-of-the-art analysis, to date only a few surveys specifically addressed existing crowdsensing solutions, being that some authors focused on specific issues such as incentives, and yet others focused on sensing styles. Our survey proposes a reference client-server architecture and then, based on that proposal, proceeds to classify up to 64 different proposals, thus providing a wider view than the surveys presented before on this topic (see Table 1 for details). Notice that the number of peer-reviewed publications only takes into account those references actually classified according to the proposed taxonomies.

3. Mobile Crowdsensing: Reference Architecture

In this section we propose a client-server design which can be adapted to the different mobile crowdsensing architectures available in the literature. By making the different proposals fit into our architecture, in sections that follow, it will then become straightforward to compare the different proposals in terms of scope, complexity, and completeness.

Our proposed architecture integrates two main modules: the Mobile Sensing Client (MSC) module and the Cloud Data Collection Server (CDCS) module. These two modules are connected to each other through a data transmission network, as shown in Figure 3. The MSC is the mobile phone or the set of mobile phones that provide sensing functionality by capturing data and then relaying that data to the CDCS. The latter is a single server or a server farm that allows receiving, processing, analyzing, and sharing sensed data. Typically, data sharing also includes the delivery of reports to participants (MSC).

For both the MSC and the CDCS we have considered four subcomponents, some of them sharing common characteristics on both MSC and CDCS. For instance, both Client and Server User Interfaces provide a graphical interface to a regular user or to the system administrator through the respective Interface Managers. On the bottom of the architecture, the Server and Client Communications Managers have also a similar purpose, typically being the component on the client that establishes connections with the server component since it should be always available. Nevertheless, configuration and task instructions, along with data reports, can also be transmitted from server to client through a push procedure.

The data management components at client and server also have some similarities, both being responsible for data processing, storage, and query. The main difference between these subcomponents is that, in the CDCS, the computation, storage, and analysis are made at a level and dimension that are clearly superior to the one made at the client, which has fewer resources.

Two distinctive components in our architecture are the Client Sensor Manager (CSM), responsible for the administration of the sensors, and the Server Task Manager (STM), which handles different tasks mostly related to data processing.

Below we proceed to describe the different architectural elements in more detail.

3.1. Mobile Sensing Client.

3.1. Mobile Sensing Client. In the scope of mobile crowdsensing, the main goal of the mobile client devices is performing data sensing and forwarding sensed data to the main server, although global data reports can also be returned to clients.

Concerning the target areas to be sensed, these can differ greatly depending on the type of application (inside buildings, outdoor, underground, in public places, etc.). In addition, each specific application will also have different requirements in terms of required sensors. For instance, sensors able to monitor the environment greatly differ from those able to monitor social interactions or the effectiveness of public transportation. In addition, sensing tasks can be triggered automatically (either periodically or based on events) or manually through an explicit user intervention. Typically, automatic mechanisms follow server instructions, while manual interactions are made possible through a User Interface specifically developed for that purpose. Independently of the actual mode of operation, the application can offer certain incentives in the form of a game [39] or another, to motivate users into adopting it. Such incentives become especially important when the user interest about the global generated data, which are based on the aggregation and processing of all measurements at the server, remain low (e.g., data being sensed is not a concern to the user); in those cases, complementary sources of motivation are required to make users run the crowdsensing application.

Focusing on the client architecture, Figure 4 shows that, to support all user activities, we have a set of managers responsible for all tasks: Client Interface Manager (CIM), Client Data Manager (CDM), Client Sensor Management (CSM), and Client Communications Manager (CCM). Each of these four components has a controller subcomponent, being the different controller elements, the ones actually responsible for supporting bidirectional interactions between the different system elements.

We now proceed to detail each of the client components in detail.

3.1.1. Client Interface Manager (CIM). This component allows applications to interact with the user (User Interface GUI). The User Interface allows displaying the values obtained from sensors in real time, to visualize previous traces through a query to its internal data storage or to query the server in order to retrieve global data reports about a certain target area. The values can be visualized through the use of graphics, maps, or other forms of representation. To achieve this goal two subcomponents are proposed: the Client User Interface and the Interface Controller.

(i) Client User Interface. It allows configuring the different parameters associated with sensing tasks, such as regulating the data acquisition frequency, defining when data should be sent to the server, and also when captures should start and stop, among others. It can also show the user feedback about ongoing or past captures, as well as global reports. It is worth highlighting that some crowdsensing solutions have no interface at the client side, meaning they only process captured data and relay them to the server.

(ii) Interface Controller. It provides the needed services to format data for presentation through the User Interface. For this endeavor, it must interact with the local storage or with the server, and it may rely on different external libraries as well (e.g., graphical representation of captured values in a map using Google Maps).

3.1.2. Client Data Manager (CDM). This element, responsible for data handling and storage, is one of the main architectural elements at the client. It is composed of five different subcomponents: data controller, Plugin Extensions, data processing,

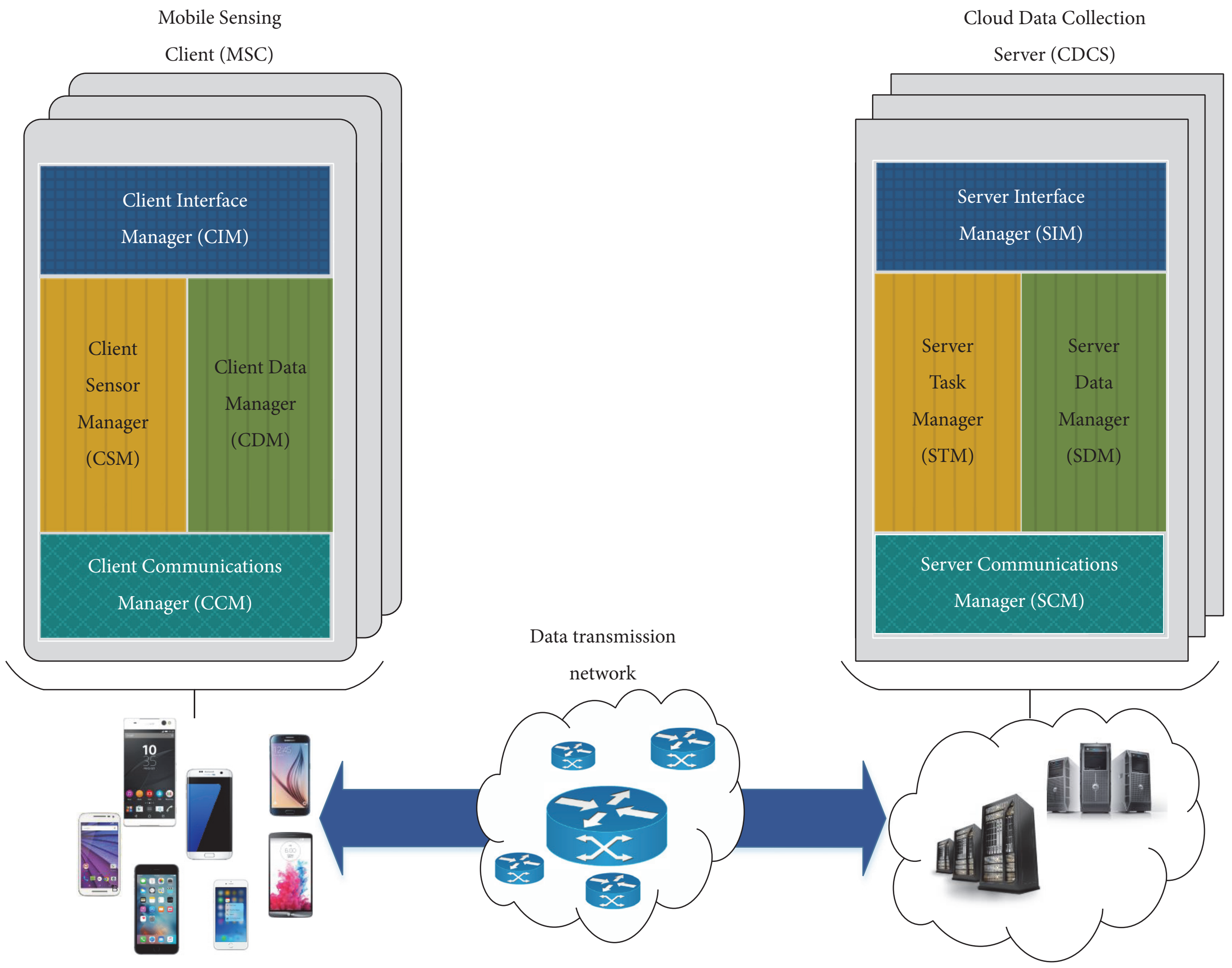

Mobile Sensing Client (MSC)
Client Interface Manager (CIM)
Client Sensor Manager (CSM)
Client Data Manager (CDM)
Client Communications Manager (CCM)
Data transmission network
Cloud Data Collection Server (CDCS)
Server Interface Manager (SIM)
Server Task Manager (STM)
Server Data Manager (SDM)
Server Communications Manager (SCM)

FIGURE 3: Proposed mobile crowdsensing architecture.

local storage, and query. We now proceed to detail each of them.

(i) Data Controller. It is the most critical subcomponent, providing the services and functions required to interact with the different subcomponents of the CDM. This interaction is made with the client via the Interface Controller, with the server via the Communications Controller and with the sensors via the Sensor Controller. In addition, it is able to handle data collection tasks as defined by the user or defined by the server through task pushing. It includes classes and methods to start, stop, and configure these tasks.

(ii) Plugin Extensions. This element allows integrating specialized plugins for a specific task such as data analytics or to add listeners to social networks like Facebook and Twitter, among others. The advantage of these plugins is that they can be easily incorporated into mobile devices via repositories such as Google Play or similar ones. Additionally, it allows plugging in a set of algorithms that perform functions including audio processing, online programming algorithms, and spatial coverage analysis.

(iii) Data Processing. This element processes raw data based on application requirements before displaying them to the end user or submitting them to the server. Although data processing can also be executed at the server side (CDCS), doing it at the client allows reducing the amount of unnecessary data produced by sensors, while also maximizing energy savings and communications bandwidth, and so it is often preferred. Typically, data processing elements include either filtering or aggregation or both functions. An example of filtering is the removal of unnecessary data fields. Examples of aggregation/fusion of data include the unification of data from different sensors or of different samples from a same sensor.

(iv) Local Storage. This element allows storing the captured data in a local data structure, which is usually a simple database like SQLite. Some solutions available in the literature skip this component, and they only process data and forward them to the CDCS. The local storage allows users to perform queries, inserts, updates, and deletes to the data according to application requirements. Typically, when storing data coming from sensors, it is often preprocessed before storage.

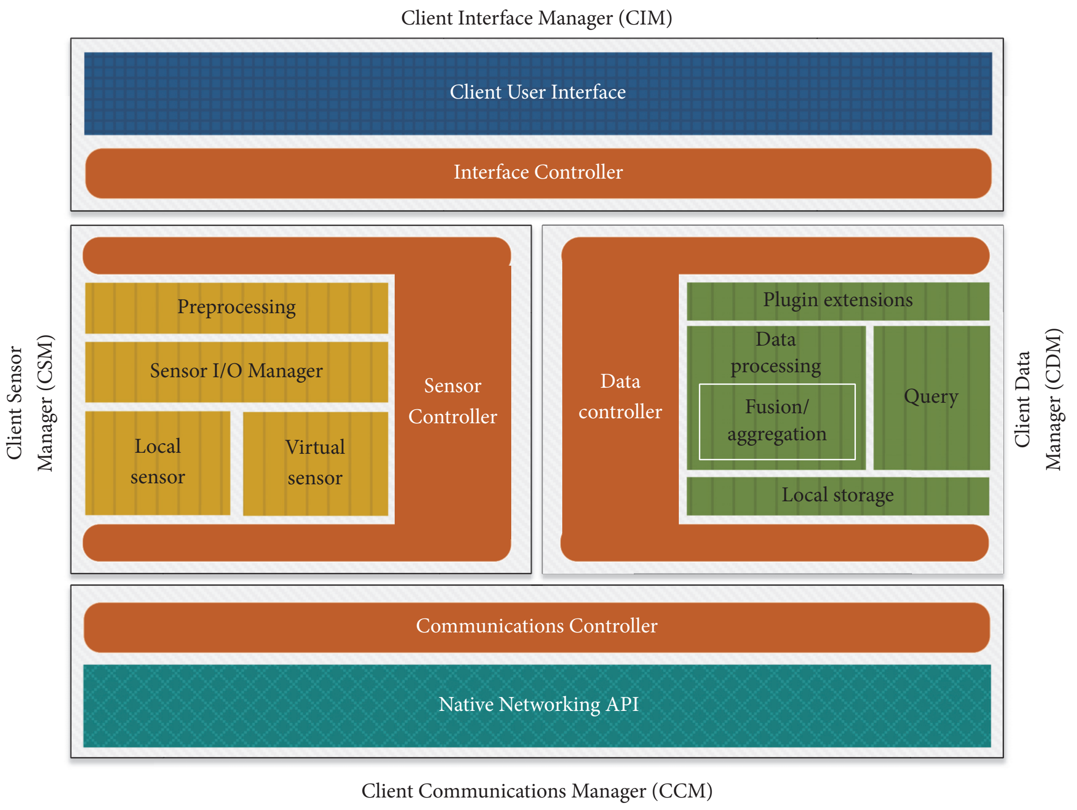


Figure 4: Mobile Sensing Client (MSC) components.

In the context of crowdsensing applications, the main types of data stored include location information, energy levels, and sensor-specific values.

(v) Query. This element allows, through structured language queries, accessing data from sensors. In particular, it will interact with all the components that make up the CDM. Among its typical features, one that stands out is the use of mobile analytics for optimizing data streaming from sensors. In some cases, this component facilitates the interaction with external databases at the CDCS in order to retrieve global data reports.

3.1.3. Client Sensor Manager (CSM). The Client Sensor Manager is the element responsible for the actual sensing tasks. Typically, it relies on high-level sensors abstractions to manage the underlying physical sensors (internal or external) as well as virtual sensors. Its functions usually include sensor discovery and sensing capabilities. Furthermore, it manages the sensor sampling frequency, as well as the preprocessing of captured data. The preprocessing executed at the CSM is only performed if necessary, and considering the actual characteristics of the sensor. Finally, the integration of external sensors and virtual sensors is performed by the Sensor Controller via the communications manager.

The CSM has five subcomponents: Sensor Controller, Preprocessing, Sensor I/O Manager, Physical Sensor, and virtual sensor. Below we describe each of its components.

(i) Sensor Controller. It enables access to the services offered by the Sensor Manager, thus providing access to virtual sensors, gyroscope, and GPS, among others.

(ii) Preprocessing. It allows the data delivered by the Sensor I/O Manager to be processed before being passed to other components. An application example is an audio capture which must be classified into voice and nonvoice regions, so that the individual speaker is segmented. Another example is the raw accelerometer data that is provided for the three axes, which can be combined to obtain the total value. In some cases these raw data can be processed at both CSM and CDM.

(iii) Sensor I/O Manager. It allows a level of abstraction for accessing both physical and virtual sensors, getting the raw data for subsequent treatment. This way, upper layers do not have to be aware of the type of sensor (physical/virtual) and its actual location.

(iv) Local Sensor. These are sensors available either on mobile devices themselves or external nearby sensors directly accessible by the mobile device. Concerning the type of sensor, most internal sensors used belong to the generic or media type. Generic sensors are those sensors embedded in mobile devices for general-purpose applications. Examples of these sensors include GPS, accelerometer, gyroscope, magnetometer, and barometer. With regard to media sensors, it refers to embedded sensors that provide support to multimedia

applications via microphone or camera. Finally, external sensors typically extend the sensing functionality by providing sensing capabilities not supported by the smartphone itself.

(v) Virtual Sensor. Virtual sensor is a logical type of sensor based on an abstract class that acts as a wrapper, encapsulating information that can be produced by a real sensor, a mobile phone, or a combination of other virtual sensors. Virtual sensors can have multiple input data streams that can be other virtual sensors or sensors accessible through a network, but there can be only one output data stream toward the sensing application. The GSN standard data model [40] is a good example of such a class of sensors.

3.1.4. Client Communications Manager (CCM). The Client Communications Manager is responsible for the transmission and reception of the data through the network. Since nowadays mobile phones include several communication interfaces including WiFi, Bluetooth, or Cellular, this empowers them to communicate in all sorts of environments, being able to adapt to different network topologies (centralized, distributed, or hybrid). The MSC may transfer the data to a primary server (centralized), toward several servers (distributed), or among themselves (peer-to-peer). The latter occurs when there are nodes that serve as intermediaries for the transmission of data between nodes and that have a limited ability to process and filter data from the sensor.

The CCM is composed of two subcomponents: the Communications Controller and the Native Networking API. In detail, these subcomponents are responsible for the following tasks.

(i) Communications Controller. It provides access to the services of the underlying communications network, allowing creating a data channel toward the CDCS (server). In particular, it is an abstract component that allows encapsulating SOAP and RESTful web services, where the first is an XML-based protocol that uses service interfaces to expose the business logic, and the second is an architectural paradigm that supports different data formats including JSON, XML, HTML, and TXT. Since communication between clients may also be required, this component will be endowed with peer-to-peer networking capabilities, possibly acting as a relay between other clients and the server(s).

(ii) Native Networking API. This component is inherent to each mobile operating system platform, and it is the one providing the actual establishment of end-to-end connections between client and server.

3.2. Cloud Data Collection Server. In the context of crowdsensing applications, the main goal of the server component, which may physically consist of a single server or a server farm, is to collect all data gathered by the different clients, storing the data, and then perform all sorts of data analytics to provide the administrator or clients themselves with a summary of the most relevant information. In addition, the server allows defining and automating some of the data collection tasks. For example, the administrator can create new tasks, and these can be deployed to clients either automatically or manually; an example of this can be the collection of the noise levels for a given target area during a given period of time. Figure 5 shows our proposed architecture for the CDCS, which includes four components: Server Interface Manager (SIM), Server Task Manager (STM), Server Data Manager (SDM), and Server Communications Manager (SCM). Notice that each of these components includes a controller. Such controllers have a critical function in the scope of our architecture, as it is the communication between adjacent controllers that allows the different components to work together, similarly to the situation at the client side.

Compared to clients, CDCS elements have much greater processing and storage capabilities. Thus, data are typically processed for a better understanding through different statistical techniques (data mining). Also, the management interface is usually web-based, allowing the administrator to easily manage, visualize, and share large amounts of data.

Depending on network scalability requirements, servers may work in either centralized, distributed, or cloud-based environments. The latter allows benefitting from deployment facilities, reduced cost, and optimized resource usage, thereby minimizing infrastructure requirements.

Concerning available technologies, server solutions may rely on a wide range of platforms, from distributed architectures in the cloud, such as Amazon Web Service (AWS) infrastructure services (EC2 and S3) [41] and Google Cloud Messaging (GCM) [42], to open source approaches such as Apache Tomcat [25, 43–45], BPEL4People [46, 47], WS-HumanTask, and JBoss JBPM [15].

Below we describe in more detail the different components at the server side.

3.2.1. Server Interface Manager (SIM). The Server Interface Manager is responsible for the interaction between user and system for task and data handling. It includes two components: the Server User Interface and the Interface Controllers.

(i) Server User Interface. It allows the user to interactively manage and schedule sensing tasks. It also supports the visualization of charts relative to sensed data. Both these actions are performed using a graphical interface that is in general web-based, meaning that the system manager can operate remotely.

(ii) Interface Controller. This is the component actually in charge of communicating with other components to meet the service requirements. An example is the programming of a sensing task, where the Interface Controller coordinates with task controllers for task planning and dissemination and with the data controller for handling data storage. In addition, it also provides application programming interfaces (APIs) to allow developers to participate in the development of different crowdsensing applications and services.

3.2.2. Server Task Manager (STM). Task Management is one of the main components at the server side according to our proposed architecture, being responsible for the planning,

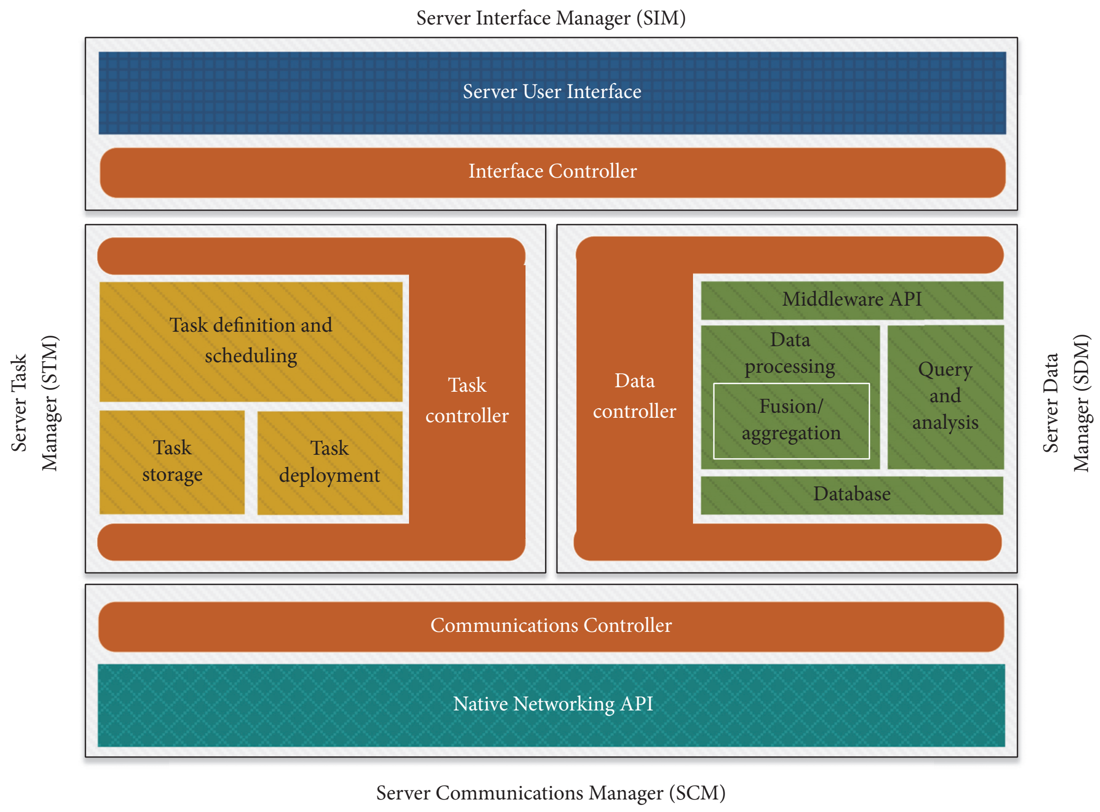


Figure 5: Cloud Data Collection Server components.

scheduling, and pushing of crowdsensing tasks. Tasks can be deployed to mobile devices either manually or automatically, and in general they rely on a system-specific language that typically differs from one solution to another due to lack of standardization. It is also worth highlighting that most implementations rely on open source tools.

The subcomponents that integrate the STM are the following.

(i) Task Controller. It works as a handler, providing the functionality required by the Server Task Manager. Typically it must attend administrator requests and may push new scheduled tasks onto clients. It can also make use of learning or approximation algorithms that optimize data collection in order to minimize energy/resource consumption at the client side. Additionally, this subcomponent includes classes and methods to start, stop, and configure the different tasks. Finally, it provides the services and functions required to interact with the other controller components. To contact clients, some implementations are based on Publish/Subscribe approaches where a server (or servers) provides a set of services to users. Additionally, in many of these Publish/Subscribe systems, the server can take intermediary functions where publishers send the messages to such intermediary server (broker), and the subscribers subscribe to information considered to be of interest, thus making this server responsible for handling the filtering, storage, and management toward the subscribers.

(ii) Task Definition and Scheduling. Among its features we can find the allocation of time and frequency of sensing, the number of mobile devices to be enabled for data collection, and the characteristics of the sensor to monitor, among others.

(iii) Task Deployment. It allows the deployment of tasks to MSCs, which can be a mere set of instructions interpreted by the existing applications. To support this option, a language defined by the application is often used, and it is typically based on SQL, XQUERY, or XML. Alternatively, a new application/component is pushed to the mobile terminal whenever new functionalities must be supported.

(iv) Task Storage. This component is responsible for the storage of current and past tasks. Since requirements are typically low, any database system suffices. In fact, it is not necessary to rely on a standard database, being also common to use a set of files, where each file describes a single task.

3.2.3. Server Data Manager (SDM).
This component is responsible for the processing, storage, and analysis of the data. It is composed of a data controller, middleware APIs, a data processing element, a query and analysis element, and a database. Below we describe in more detail each of these components.

(i) Data Controller. It offers access to the services offered by the SDM, supporting a set of algorithms or applications

that allow handling data in collaboration with the task controller, Interface Controller, and other system components. In addition, it acts as a handler for communications to/from middleware APIs.

(ii) Middleware API. A middleware API is typically an extension providing more sophisticated data processing/analysis. It can incorporate data analysis tools such as data mining, analytical libraries, or other, allowing easily handling large volumes of data. Deployments at this level can be drivers or web services that enable access to databases through JDBC or other methods, such as CUPUS [48] and CAROM [27], which additionally provide data fusion and data filtering techniques.

(iii) Data Processing. The functionality of this component is similar to the data processing made at the client (CSM). The main difference is the volume of data that has to be handled at the server side. Typically, it provides functions to filter and merge multiple streams of data, providing aggregation levels that clearly surpass those levels achievable at the client side. With this purpose, it uses techniques that require a higher level of processing, such as FSI or ECSTRA [27], among others.

(iv) Query and Analysis. This component integrates both query and data analysis functionalities. It allows, through a structured query language, accessing the resources available at the server's database. Additionally, it can rely on different analysis tools to meet the requirements of other system components.

(v) Database. This component provides a database management system that allows storing the gathered data coming from the different Mobile Sensing Clients (MSCs). In the scope of the SDM, it is mandatory since it is a basic system requirement. It should be noted, though, that the database itself is not necessarily contained in a single server, and so distributed storage environments are contemplated as well. Common database management systems include MySQL and PostgreSQL, among others.

3.2.4. Server Communications Manager (SCM). The Server Communications Manager is responsible for interacting with the different clients, having characteristics similar to the Client Communications Manager. The interaction with clients is bidirectional: we have transmission toward the client when pushing new tasks, and we have transmissions from clients when receiving sensed data.

The SCM has two main components, the Communications Controller and the Native Networking API, both of which we now detail.

(i) Communications Controller. It offers the services necessary to establish communication between the MSC and the CDCS, usually as listeners for data gathering, or starting connections when task pushing is required. Additionally, it can rely on high-level communication services like SOAP and can also have adapters for any specific protocol or method of communication used by different server components. An example can be a REST-SOAP Adapter, which receives a SOAP request and adapts it to a REST-service format.

(ii) Native Networking API. This component is the one responsible for actually communicating with client devices through the establishment of end-to-end connections. Typically, reliable TCP connections are established.

4. Analysis of Existing Proposals

In this section, we provide an analysis of the different solutions available in the literature, using the architecture in Section 3 as reference for our classification. For our study, we focused on research works published in the crowdsensing field during the past five years, with a special emphasis on smartphone-based crowdsensing solutions.

For the sake of clearness and completeness, our analysis was split into four well-defined parts: (1) general analysis, (2) client-side analysis, (3) server-side analysis, and (4) data delivery approaches. The first part presents a general analysis of the various proposals, and performs a synthesis of the different contributions in the scope of our architecture. In the second part we have addressed in more detail those proposals detailing a client-side architecture, that is, describing the CUI, CDM, and MSC components, while for the third one we detail server-side architectures, describing the SUI, STM, and SDM components. In addition, we have clearly assessed to what degree the different solutions are able to provide all the functionalities envisioned in our proposed architecture. It is worth highlighting that both client and server analysis include not only proposals specific to client/server sides, but also global solutions whenever they provide details about all the elements involved in the end-to-end interaction. Finally, we have classified those solutions by providing details about the communications system defined for interactions between clients and server and also about supported topology, selected technology, and other relevant features. Again, for this data delivery analysis, any proposal providing enough details was included, no matter how broad or how specific was the proposal itself.

4.1. General Analysis. In our general analysis of the different crowdsensing solutions, we have classified information based on three parts. In the first one we provide generic information about the different proposals, in the second one we describe aspects related to security/privacy and energy consumption, and finally we provide a summary of contributions for each proposed architecture. This classification and characterization is presented in Table 2.

4.1.1. General Features. With regard to the *general features*, we found that the vast majority of solutions propose an integral solution to the sensing tasks at both client and server sides. Other solutions propose a specific middleware to help in the tasks of data collection and processing.

Concerning the strategy adopted for data collection, most proposals opted for a participatory approach for data sensing where users are fully aware of the data collection process, and

Table 2: Classification of the different technologies according to the proposed architecture.

Publication	Year	Proposal	Type	Recollection strategy	Target applications	# components levels	Privacy	Energy	MSC				Tx	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
PRISM [24]	2010	Platform	Prototype	Both	Heterogeneous	3	X	X	★	☆	☆	☆		☆	☆	☆	☆
Ear-Phone [5]	2010	Client-server	Real/simulations	Opportunistic	Environment monitoring	2		X	☆	☆	☆	☆			★	☆	☆
Anonysense [6]	2011	Framework	Prototype	Opportunistic	Environment monitoring	3	X	X	☆	☆	☆	★		★	★	★	★
Medusa [25]	2012	Framework	Prototype	Participatory	Heterogeneous	2	X		★	☆	☆	★		★	☆	★	☆
Pogo [49]	2012	Middleware	Real imp.	Participatory	Heterogeneous	2	X	X	☆	☆	☆	★		★	☆	☆	☆
DAM4GSN [50]	2012	Architecture	Real imp.	Participatory	Heterogeneous	3		X	★	☆	★	★		★	★	☆	☆
MECA [26]	2012	Middleware	Prototype	Participatory	Heterogeneous	3				☆	☆	☆		★	★	★	☆
StressSense [14]	2012	Framework	Real imp.	Participatory	Healthcare	1			★	☆	☆	☆		☆	★	☆	☆
CrowdITS [9]	2012	Framework	Real imp.	Participatory	Transportation and urban sensing	3		X	★	☆	☆	★		★	★	★	★
SmartCity [10]	2013	Framework	Real imp.	Participatory	Transportation and urban sensing	4			☆	☆	☆	☆		★	★	☆	☆
ILR [51]	2013	Scheme	Simulations/real imp.	Participatory	Location Services	3	X		☆	☆	☆	☆		★	☆	☆	☆
CAROM [27]	2013	Framework	Real imp.	Participatory	Heterogeneous	3		X	★	☆	★	☆		☆	★	☆	★
SoundOfTheCity [7]	2013	Client-server	Real imp.	Both	Environment monitoring	2		X	★	☆	☆	☆		☆	★	☆	☆
MoPS [8]	2013	Middleware	Prototype/real imp.	Both	Environment monitoring	3		X	★	★	☆	★		☆	☆	★	☆
NoiseNYC [16]	2013	Framework	Real imp.	Participatory	Healthcare	3				☆			☆	★	★	☆	☆
Vita [15]	2013	System	Real imp.	Participatory	Healthcare	2		X	★	☆	☆	★	☆	★	☆	☆	☆
Matador [52]	2013	Framework	Simulations	Both	Location services	—		X	☆	☆	☆	★	☆	★	☆	★	★
Usense [53]	2013	Middleware	Real imp.	Both	Heterogeneous	3	X	X	★	★	☆	★		★	☆		☆
REPSense [11]	2013	Applications	Real imp.	Opportunistic	Environment monitoring	2			☆	☆	☆	☆				☆	
BeC3 [46]	2013	System	Real imp.	Opportunistic	Heterogeneous	3		X	★	☆	★	★		★	☆	★	★

Table 2: Continued.

Publication	Year	Proposal	Type	Recollection strategy	Target applications	# components levels	Privacy	Energy	MSC				Tx	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
McSenseFoster [12]	2013	System	Real imp.	Participatory	Transportation and urban sensing	3			☆	☆	☆	☆	☆	☆	☆	☆	★
NoizCrowd [18]	2013	Components	Real imp.	Participatory	Healthcare	4			☆	☆	☆	☆		★	★	☆	☆
PLUS [13]	2014	Framework	Real imp.	Participatory	Transportation and urban sensing	2		X	★	☆	☆	☆		☆	☆	☆	☆
MOSDEN [28]	2014	Middleware	Real imp.	Participatory	Heterogeneous	3		X	★	★	★	★		★	☆	★	☆
SenseDroid [54]	2014	Framework	Prototype/real imp.	Both	Location services	3		X	★	★	★	☆			★	☆	
SenSocial [19]	2014	Middleware	Real imp.	Both	Social recommendation	3	X	X	★	★	☆	★		★	★	★	☆
AlienvsMobile [51]	2014	Client-server	Prototype/real imp.	Participatory	Location services	3	X		★	☆	☆	☆		☆	☆	☆	☆
SaaS [44]	2014	Framework	Description	Participatory	Heterogeneous	3				☆		☆		★	★	☆	☆
CUPUS [48]	2014	Middleware	Prototype/real imp.	Both	Heterogeneous	3		X	★	☆	★	☆	☆	☆	☆	☆	☆
BLISS [48]	2014	Framework	Simulations	Participatory	Social-budget	—			☆	☆	☆	☆				☆	☆
GPS-Less [55]	2014	System	Real imp./simulations	Both	Transportation and urban sensing	2		X	☆	☆	☆	☆		★	☆	★	☆
MCS-Space [21]	2014	System	Real imp.	Participatory	Social-general	3		X	☆	☆	☆	☆		★	★	★	☆
SPREAD [56]	2014	Algorithm	Prototype	Participatory	Location services	—			☆	☆	☆	★				☆	☆
Map++ [57]	2014	System	Real imp.	Participatory	Transportation and urban sensing	—		X	☆	☆	☆	☆			☆	☆	☆
JoiPolices [58]	2014	System	Simulations	Participatory	Heterogeneous	3	X		☆	☆	☆	☆		★	☆	☆	☆
CrowdRecruiter [59]	2014	Framework	Real imp.	Both	Location services	—		X		☆		☆		☆	☆	★	☆
Neighbor [60]	2014	Middleware	Simulations	Opportunistic	Heterogeneous	—		X		☆	☆	☆	★	☆		☆	☆
LineKing [22]	2014	System	Real imp.	Both	Social recommendation	2		X	★	☆	☆	☆		★	☆	☆	☆
GROPING [61]	2014	Application/Fr	Prototype	Participatory	Location services	2		X	★	☆	☆	☆		☆	★	★	☆
EasyHarvest [45]	2014	Framework	Prototype	Opportunistic	Heterogeneous	2			★	☆	☆	★		★		☆	☆

Table 2: Continued.

Publication	Year	Proposal	Type	Recollection strategy	Target applications	# components levels	Privacy	Energy	MSC				Tx	CDCS			
									CIM	CDM	CSM	CCC		SCM	SDM	STM	SIM
WiFIScout [62]	2014	System	Real imp.	Participatory	Social recommendation	2			☆	☆	☆	☆		☆	☆		
QoS-Constrained [63]	2014	Framework	Simulations	Both	Heterogeneous	2					☆	☆		☆	☆	★	
Ecosystem [47]	2015	System	Real imp.	Participatory	Social-general	4		X	★	☆	☆	★		★	☆	★	☆
MCSaaS [43]	2015	Framework	Real imp.	Both	Heterogeneous	4		X	☆	☆	★	★		★	★	★	★
COUPON [64]	2015	Framework	Simulations	Opportunistic	Heterogeneous	—		X	☆	☆	☆	☆	★	☆			
ADTS [65]	2015	Application	Simulations	Participatory	Location services	1			☆	☆	☆	☆					
Anonymity [66]	2015	Framework	Prototype/real imp.	Participatory	Transportation and urban sensing	2	X	X	☆	☆	☆	☆	★		☆	☆	☆
TYT [17]	2015	Framework	Real imp.	Participatory	Healthcare	3	X		★	☆	☆	☆		☆	☆	☆	★
QOATA [67]	2015	Application	Simulations	Participatory	Heterogeneous	—				☆	☆	☆					
NeCoRPIA [68]	2015	Protocols	Simulations	Opportunistic	Heterogeneous	—				☆		☆	★	☆		☆	
QoSMCS [69]	2015	Framework	Simulations	Opportunistic	Heterogeneous	3		X		☆	☆	☆	★	☆	★	☆	☆
FlierMeet [70]	2015	Framework	Real imp.	Participatory	Transportation and urban sensing	3			★	☆	☆	☆		☆	☆	☆	☆
PRESM [23]	2015	Scheme	Real imp.	Participatory	Transportation and urban sensing	3	X		★	☆	☆	☆		☆	☆	★	☆
SmartRoad [71]	2015	Framework	Real imp.	Participatory	Transportation and urban sensing	2		X	★	☆	★	★		★	★	★	☆
MCSgame [72]	2015	System	Simulations	Participatory	Location services	2		X		☆	☆	☆		★	☆	☆	☆
RemoteCloud [73]	2015	—	Simulations	Participatory	Transportation and urban sensing	—					☆	☆	★		☆	☆	☆
MDPPs [74]	2015	Middleware	Real imp.	Opportunistic	Location services	3			★	☆	☆	☆		★	★	★	☆
RuPS [75]	2015	Framework	Real imp.	Participatory	Location services	4			☆	☆	☆	☆		☆	★	☆	★
EffSense [76]	2015	Framework	Simulations	Opportunistic	Heterogeneous	—		X		☆	☆	☆	★		☆	☆	
PLP [77]	2015	Scheme	Simulations	Opportunistic	Heterogeneous	—	X			☆	☆			★	★	☆	
Sahyog [78]	2015	Middleware	Real imp.	Participatory	Heterogeneous	3	X		★	☆	☆	★		☆		☆	
Context [79]	2015	Middleware	Simulations	Opportunistic	Social recommendation	—		X		☆	☆	☆	★		☆	☆	
MoreWithLess [80]	2015	Framework	Real imp.	Participatory	Heterogeneous	4		X		☆					★	★	
Sparse [81]	2016	Framework	Real imp.	Participatory	Heterogeneous	3	X			☆					★	★	

they actively participate in that process. Other approaches, however, prefer using opportunistic systems that operate in a more autonomous manner, gathering information in the background at appropriate times; finally, a few proposals combine these two approaches to achieve a more complete functionality.

Regarding the target applications addressed in the different works, we found that the majority of the proposals are flexible enough to embrace heterogeneous applications; that is, they can adapt to generic sensing tasks, although we can also find proposals that are specific to transportation and urban sensing environments, and to a lesser extent to health, social, and other environments.

Finally, with respect to the number of differentiated elements defined for each proposed architecture, we found that there are significant differences among authors. For instance, [18, 43, 47] split their proposed functionality into four different levels, similarly to our proposal. In particular, NoizCrowd [18] defines an architecture based on four components which are data gathering, data storage, noise modeling, and data analytics/visualization. SmartCity [47] also defines a four-element architecture composed of social networks, Ubiquitous Sensors, a Mobile Context-Aware Platform, and the Cloud Platform. MCSaaS [43] defines four core submodules, namely, Cloud Broker, Orchestrator, Customization Service, and Deployment Manager. In general, most proposed architectures only defined two or three levels, as is the case of [10], which defines a generic Publish-Subscribe communication with three roles (producers, services providers, and consumers), along with Analytics Components.

4.1.2. Privacy and Energy Issues. In general, the success of mobile crowdsensing applications is dependent on how each solution addresses user concerns about his/her own privacy. Energy consumption is another critical issue, as applications draining a significant amount of battery power will be rejected by most users. So, both energy and privacy issues are relevant in the scope of crowdsensing solutions, the reason why they have been addressed by different researchers.

Our analysis has shown that most studied proposals have addressed energy efficiency issues, while only some of these have introduced mechanisms to mitigate security and privacy concerns. In fact, we find that very few solutions [6, 19, 24, 53, 66] actually account for both privacy and energy efficiency issues. We now proceed to discuss these prominent solutions in more detail.

PRISM [24] supports privacy though a registration process on a PRISM server for each enabled terminal. The registration is maintained by software and it expires within a given period of time. When the registration period expires, terminals wait for a random time and proceed to register again. With regard to energy consumption, PRISM maintains a control of energy consumption on mobile phones through its prism sandbox, which is able to perform coarse-grain power monitoring.

Anonysense [6] uses a server that is responsible for registering and authorizing mobile phones. During registration, Anonysense installs its software along with the IP addresses and certificates for its task service and report service. Concerning energy consumption, the tasks can be divided into two suboperations: sensing and signing. In the first, the RogueFinder application is used to detect rogue APs in a given area, while the ObjectFinder application attempts to find a specific Bluetooth MAC address. The second group addresses whether a data report contains sensitive data. Additionally, it estimates the energy cost associated with these operations.

Usense [53] includes a component for securing communications. Additionally, it manages user preferences in terms of resource and privacy restrictions. These features are processed through the sensing agent, which is an application deployed on the device itself. In addition, Usense's middleware is able to save energy using a mechanism that avoids taking measurements in those areas where it already has enough data, or when the phenomenon is mostly invariant.

SenSocial [19] has a module for privacy management control which allows managing policies regarding the type and level of granularity of sensed data, deciding what will be stored and made available to the different middleware components. SenSocial uses filtering rules for maintaining energy efficiency, thereby restricting transmissions only to those cases passing the set of defined rules. Also, SenSocial discriminates the energy consumption associated with the accelerometer sensors, microphone, GPS, Bluetooth, and WiFi.

The last proposal in this group is Anonymity [66], which proposes an anonymous data reporting protocol for participatory applications. The idea is that the protocol avoids including identification information that can be vulnerable. The anonymous data protocol is divided into two stages: the first is a slot reservation stage (scheme based on public key encryption), while the second one is a data submission stage (scheme based on an XOR operation). Through comparison against a similar study, authors show how it is able to improve data submission performance. With regard to energy consumption, the smartphone's battery values are measured using a multimeter. It also presents an analysis of the energy overhead associated with data submission.

4.1.3. Analysis of Contributions for Each Proposal. The main goal of this survey is to assess the actual contributions made by the different authors taking as reference the architecture proposed in Section 3. So, the last part of Table 2 (columns MSC, Tx, and CDCS) provides a first insight into the actual contribution made by the different components at the client (MSC) and server (CDCS) sides, in addition to the end-to-end transmission process itself (Tx).

We provide a three-level classification of proposals, where a dark star means that the particular solution fulfills the expected functionality for that component, while a white star means that the solution only provides a partial fulfillment of the selected characteristics. The nonfulfillment of the characteristics of a component is represented by the absence of any star.

Overall, we can observe that the majority of the proposals are quite representative in the scope of our architecture, providing most of the expected functionalities. Nevertheless, we can also find solutions such as MOSDEM [28] and

SenseDroid [54] that focus mostly on MSC-related functionality. Similarly, we can find solutions such as MCSaaS [43] that focus on the CDCS instead.

4.2. Client-Side Analysis. In this section we focus on the specific contributions to the MSC, which is the client side of our proposed architecture. To achieve it, in Table 3, we describe the features of the different proposals regarding the Client Interface Manager (CIM), the Client Data Manager (CDM), and the Client Sensor Manager (CSM). Notice that we excluded the Client Communications Manager (CCM) from this section, as it will be addressed separately in Section 4.4. Also notice that the table is split into two sections, being that proposals in the upper section are client-specific, meaning that the publication only describes the client side of the crowdsensing architecture, while proposals in the bottom section describe both client and server sides.

Concerning the CIM, we found that most of the solutions provide a graphical User Interface designed for the Android operating system, thus typically adopting the Java language for development. In fact, only a few solutions such as LineKing [22], TYT [17], and DAM4GSN [50] focused on other operating systems. Also, most of the proposals allow the user to have access to an administrative interface in order to have control over sensing tasks.

With regard to the CDM we observe that, in general, most available solutions resort to plugins or external libraries in order to simplify their processing, query, and storage tasks on the device by reusing existing software. In particular, different techniques and algorithms are adopted mostly to support the data collection procedure including spatiotemporal area calculation and programming algorithms. The spatiotemporal coverage of an area refers to the amount of time and space needed to properly sense that area according to the target task, Usense [53] being the most widely used. Also, we found that although several solutions provide data analytics within the mobile device itself, such functionality is seldom combined with the use of plugins.

Among solutions integrating plugins, we would like to highlight solutions, such as DAM4GSN [50], MOSDEN [28], and CAROM [27], that use open source GSN technologies for IoT. In particular, CAROM [27] uses a plugin where, among other functionalities, it incorporates Open Mobile Miner (WMO), which is an open source solution that allows performing data analysis on the mobile terminal. Similarly, SenSocial [19] uses a plugin providing an agent able to retrieve data from both Facebook and Twitter, and its process is based on joining online social networks (OSNs) that provide a physical context data stream. In addition, we found that few solutions include a broker functionality. We also found that there is a balance between the approaches preferring pushing contents onto the servers and solutions that prefer the server to pull contents instead.

With regard to data processing, we find that few solutions perform aggregation-fusion on the mobile device, as opposed to data filtering, whose support is quite common. Finally, with regard to the Client Sensor Manager, in general, the different proposals available make use of generic sensors that are internal to the mobile devices, offering in a few cases support for external sensors. There is also evidence of applications using external sensors or multimedia stream processing before sending the streams to the server (see, e.g., StressSense [14] and REPSense [11]).

Finally, regarding the adoption of virtual sensors, only a minority of the proposals studied do so. In particular, options such as DAM4GSN [50], MOSDEN [28], and CAROM [27] relied on an adapted version of GSN [40], while other proposals like SenseDroid [54], CUPUS [48], and SmartRoad [71] provide their own virtualization solutions.

4.3. Server-Side Analysis. In this section we will focus instead on the server side, which in the scope of our proposed architecture takes the name "Cloud Data Collection Server" (CDCS).

Table 4 describes the features of server-related proposals. Similarly to the previous section, the table is split into two parts, being that proposals in the upper part are server-specific (the publication only describes the server side of the crowdsensing architecture), while proposals at the bottom section are complete ones, describing both client and server sides; obviously, since client-related details were already presented above, in this section we only focus on server-related issues.

In general, we observe that most of the proposals provide a web interface for management and result presentation purposes, and most of them also provide data management and data sharing functionalities. The technologies used in these proposals are generally open source solutions like Apache, Java, and PHP, among others, and many of them use a database manager such as MySQL and PostgreSQL. Also, there is evidence that many proposals rely on a cloud infrastructure provided by Amazon [41] or Google [42].

With respect to the Server Task Manager, we find that most proposals present mechanisms to manage and deploy sensing tasks. In particular, in terms of task deployment, we find that the number of proposals adopting a push-based approach is similar to those adopting a pull-based approach.

Regarding the language used for task definition, some solutions describe tasks using specific algorithms, while others prefer using a programming language, as is the case of Pogo [49], Anonysense [6], and Medusa [25].

With respect to data management at the server, most solutions perform data aggregation similarly to client-side solutions. Some of them use intelligent data analysis techniques such as Big Data [8, 10, 43], MCDM [75], and PFISR [21], and various other statistical tools. Recent research works [80–82] take advantage of the space and the time correlation between the discovered data of different subareas with the aim of reducing the number of tasks required for the target purposes. Wang et al. [81, 82] present a solution called sparse MCS framework that uses inference algorithms to ensure the quality of the data after being collected. Instead, Xu et al. [80] describe a framework that uses four states (data structure conversion, base training, sampling, and reconstruction). It relies on programming algorithms to create a baseline dataset using the K-SVD algorithm, while for the reconstruction the Orthogonal Matching Pursuit recovery algorithm is adopted. In both cases, the intention is to produce a global saving

Table 3: Classification of the different technologies according to the proposed client architecture.

	Client interface manager (CIM)				Client Data Manager (CDM)									Client Sensor Manager (CSM)		
						Plugins/external libraries				Data processing						
Publication	Interface	Admin	OS	Technology	Characteristic	Real time	Sensing task	Data analytics	Broker/plugin	Filter	Aggregate	Query	Local store	Type of sensor	Virtual sensor	Preprocessing
PRISM [24]	X		Windows mobile	C# and C++	PRISM's sandbox	Sent	Push-pull					X	X	Generic/multimedia		
DAM4GSN [50]	X	X	Android/IOS	Java	XML-based/GSN	Store and send	Push-pull		Both	X		X	X	Generic and external	X	X
StressSense [14]	X		Android	Java, C and C++	GMMs and simulate	Send	Pull		Plugin				X	External microphone		X
Usense [53]	X	X	Android	Java-BlueZen	XML-based/spatiotemporal	Store and send	Push		Plugin	X	X	X	X	Generic/multimedia		X
REPSense [11]	X		Android		Scheme divide/merge	Store and send				X		X	X	GPS, ambient light, air pressure, accelerometer		X
PLUS [13]		X	Android		Markov Predictor Model	Store and send		X		X			X	GPS		X
MOSDEN [28]	X	X	Android	Java	XML-based/GSN	Store and send	Push-pull	X	Both	X		X	X	Generic and external	X	X
SenseDroid [54]			Android		Spatiotemporal	Store and send			Broker	X	X		X	Generic and external	X	X
AlienvsMobile [39]	X		Android	Java	Area coverage	Send	Pull							GPS, barometric pressure		X
CUPUS [48]	X		Android		CUPUS MIOs	Store and send	Push-pull			X		X	X	Generic and external	X	X
BLISS [20]				Simulated	Simulated/online learning	Send		X		X		X		N/A simulated		
SPREAD [56]				Simulated	Simulated/area of interest	Send				X		X		GPS and simulated		
MAP+ [57]	X		Android		DBSCAN	Store and send	Pull			X			X	External GPS		X
ADTS [65]				Simulated	ADTS algorithm	Send		X		X				Generic		
FlierMeet [70]	X		Android		STA grouping	Send	Pull							GPS, light sensor, accelerometer, magnetometer		X
MDPPs [74]		X	Android		Spatiotemporal coverage (MDPPs)	Stored and sent	Push	X				X	X	Generic sensor		
EasyHarvest [45]	X	X	Android	Java for Android	Spatiotemporal coverage	Store and send	Binary push	X		X		X	X	Generic sensor		

TABLE 3: Continued.

	Client interface manager (CIM)				Client Data Manager (CDM)									Client Sensor Manager (CSM)		
						Plugins/external libraries				Data processing						
Publication	Interface	Admin	OS	Technology	Characteristic	Real time	Sensing task	Data analytics	Broker/plugin	Filter	Aggregate	Query	Local store	Type of sensor	Virtual sensor	Preprocessing
Sahyog [78]	X		Android	Java	Query format	Store and send	Push-pull					X	X	GPS, accelerometer		
WiFIScout [62]	X	X	Android		Read WiFi	Store and send	Pull			X			X	GPS/WiFi		
Ear-Phone [5]	X		Symbian	Java	Spatiotemporal coverage	Send	Pull			X		X	X	GPS microphone		X
Medusa [25]		X	Android	Java SMS and MMS	MedScript XML	Send	Push	X	Broker	X			X	GPS, multimedia		X
CrowITS [9]	X	X	Android/IOS		Plugin-based	Store and send	Push-pull		Plugin			X		GPS		
CAROM [27]	X	X	Android	Java/GSNLite	XML-based/GSN	Send	Push-pull	X	Plugin (OMM)		X	X	X	Generic and external sensor	X	X
SoundOfTheCity [7]	X	X	Android	Java		Store and send	Pull					X	X	GPS, microphone		X
MoPS [8]	X	X	Android	Java	Broker Mios		Push	X	Broker	X		X		External pollution		X
Vita [15]	X	X	Android	Java	XML-based	Send	Push		Broker	X		X	X	Generic and multimedia	X	
Matador [52]	X	X	Android		XML-based spatiotemporal	Send	Pull	X		X			X	GPS		X
SenSocial [19]	X		Android	Java	XML-based/plugin OSN	Send	Push-pull	X	Both	X	X	X	X	GPS, accelerometer, microphone, social		
MCSinSpace [21]	X	X	Android			Send	Push			X		X	X	GPS signal analysis		X
LineKing [22]	X		Android/IOS		Wait-time algorithm	Send	Pull	X		X		X	X	GPS, accelerometer		
Ecosystem [47]	X	X	Android	Android app	XML-based	Send	Push	X		X			X	Generic and external sensor		X
TYT [17]	X	X	Android/IOS	Android and IOS apps		Send	Pull					X	X	Heart rate, blood pressure, oxygen saturation		
PRESM [23]			Android	RSS map	Map generation	Store and send	Push	X		X				GPS locations		X
SmartRoad [71]	X	X	Android	Java	Detection and identification learning algorithm	Store and send	Push		Plugin			X		GPS, power sensor	X	X

Table 4: Classification of the different technologies according to the proposed server architecture.

Publication	Server Interface Manager (SIM)				Server Task Manager (STM)					Server Data Manager (SDM)				
	Interface	Manager	Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Data processing		Query and analysis	Database
											Filter	Aggregate		
Anonysense [6]	Web	X	X and maps	Ruby and small HTTP Severs	AnonyTL/Ruby	Authentication servers		X	Push/pull	Privacy distribution	X		Query	SQLite3
Pogo [49]	Web	X		JavaScript-Openfire XMPP Pub/Sub	Scripts-Rhino	Generic		X	Push	Multibroker	X	X	Query	SQL
MECA [26]	Web		X		Edge task	Generic	X		Pull	Analytics library and multibroker		X	Query and analysis	SN and device
SmartCity [10]	Web		X	XMPP Pub/Sub		Learning algorithm	X	X	Push	Big Data, data mining Casandra/s4	X	X	Query and analysis	NoSQL
ILR [51]	Web	X		Java/J2EE and Glassfish Application Server	ILR scheme	Location reliability algorithm			Push	Simulations NS2 Real Traces	X		Query and analysis	Derby
NoiseNYC [16]						Generic				3D tensor Kriging, heatmaps, and others		X	Query and analysis	S/N and device
BeC3 [46]	Web	X	X	C, Java/XMPP Pub/Sub, D-LITe Cloud	D-Lite/Python BPEL WS-CDL	Generic	X	X	Push/pull			X	Query	S/N and device
McSense [12]	Web	X	X	Java Servlet Ajax Web-Apache Spring		Generic	X	X	Push		X		Query and analysis	PostgreSQL
SaaS [44]	Web	X	X			Generic	X	X	Push		X	X	Query and analysis	S/N and device
GPS-less [55]	Web			JavaScript		Coverage model algorithm		X	Push			X	Query and analysis	S/N and PostgreSQL
JoinPolices [58]						Budget efficiency algorithm			Push	MatLab		X	Query and analysis	
MCSaaS [43]	Web	X	X	Java, Python/Apache Tomcat, Google Cloud Messaging (GCM)	XML task	Generic	X	X	Pull	Cloud broker Big Data	X	X	Query and analysis	S/N and device
GROPING [61]				Amazon Mechanical Turk (AMT)		Location estimation algorithm	X		Pull		X	X	Query and analysis	S/N

TABLE 4: Continued.

Publication	Server Interface Manager (SIM)				Server Task Manager (STM)						Server Data Manager (SDM)			
											Data processing			
	Interface	Manager	Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Filter	Aggregate	Query and analysis	Database
Qoata [67]					QOATA scheme simulated	Online learning and task allocation algorithm						X		
MCS game [72]						Q-learning algorithm	X		Push	Nash equilibrium (NE)			Query	
RuPS [75]	Web	X	X			Generic			Pull	Multi-Criteria Decision Making (MCDM)		X	Query and analysis	S/N and device
NoizCrowd [18]			X			Spatial and temporal algorithm	X		Pull	Big Data		X	Query and analysis	SciDB and SPARQL
QoS-Constrained [63]						Spatial temporal coverage			Pull	QoS: Min, Max, utility	X	X		
CrowdRecruiter [59]						Spatial temporal coverage		X	Pull		X	X		S/N
PLP [77]						Learning algorithm			Pull	RSS fingerprint	X	X	Query and analysis	S/N
MoreWithLess [80]						Spatial temporal correlations		X	Pull		X	X	Query and analysis	
Sparse [81]						Spatial temporal correlations			Pull		X	X	Query and analysis	
Ear-Phone [5]	Mobile			PHP script and Java		Generic			Pull	GPS MGRS converter		X	Query	MySQL
Medusa [25]	Web			Apache, PHP, Java, and AMT	MedScript/Python	Generic	X		Push	Stage Library	X		Query	MySQL and device
CrowITS [9]	Web	X	X	Google C2DM and PHP-Google Maps API		Generic	X	X	Push/pull		X	X	Query and analysis	MySQL and device
CAROM [27]	Web	X	X	AMT EC2 and GSN	Task-XML/GSN	Generic	X	X	Push/pull	Data mining/FSI fussy	X	X	Query and analysis	S/N and device
SoundOfTheCity [7]	Web		X	JavaScript		Generic			Pull	Media streaming and encoding		X	Query and analysis	MySQL
MoPS [8]	Web			Java and Google Cloud Messaging (GCM) Pub/Sub		Generic	X	X	Push/pull	Data mining		X	Query	S/N
Vita [15]	Web	X	X	Apache Tomcat and JBoss and (AWS) EC2 AND S3	Task BPEL/ BPEL4People: ODE and jBPM	Genetic algorithm and *K*-means	X	X	Push			X	Query	S/N
Matador [52]	Web	X	X		Matador task	Spatial temporal algorithm		X	Pull	Adaptive sampling algorithm		X	Query	S/N
SenSocial [19]	Web		X	PHP script/Pub-Sub (MQTT)	XML	Task OSN		X	Push/pull	OSN and filter aggregated	X		Query and analysis	MongoDB and device
MCSinSpace [21]	Web	X		Mapping and tomographic reconstruction algorithms		Generic	X	X	Push	PFISR, Kalman filter, and interpolates	X	X	Query and analysis	S/N

Table 4: Continued.

Publication	Server Interface Manager (SIM)					Server Task Manager (STM)					Server Data Manager (SDM)			
											Data processing			
	Interface	Manager	Shared	Technology	Language	Task	Cloud	Automatic task	Sensing task	Middleware	Filter	Aggregate	Query and analysis	Database
LineKing [22]	Web			Apache HTTP and AWS EC2		Generic	X	X	Pull	Wait-time estimator	X		Query	MySQL and device
Ecosystem [47]	Web	X	X	Amazon EC2 M1, Ubuntu, and Apache Tomcat	Task BPEL XML	Generic	X	X	Push/pull		X	X	Query and analysis	S/N
TYT [17]	Web	X	X	PHP Laravel		Generic	X		Pull	Algorithm TYT		X	Query	MySQL
PRESM [23]	Web		X			Algorithms CS-based and RSS		X	Push	RSS Map generation	X		Query	S/N
SmartRoad [71]	Web			Java, Django, and Python plugin/Google Maps		Learning algorithm		X	Push	Heatmap, 3D view		X	Query and analysis	MySQL and device

on detection costs (power consumption, network resources) while ensuring the overall data quality.

As output, data can be presented in different formats, the use of heatmaps being a representative example when sensing that information is geolocated.

4.4. Data Communications Issues. We conclude our analysis of the current crowdsensing literature by focusing on client-server communication solutions. Notice that, since communications simultaneously involve clients and servers, we address communication issues jointly in this section.

Table 5 summarizes the main communication characteristics associated with the different proposals. We have also surveyed the metrics used by each proposal for performance analysis and classified them according to their scope as generic, QoS, and scalability. As *generic* performance metrics we refer to those proposals addressing network performance in terms of packet delivery ratio, end-to-end delay, and transmission overhead, among others. *QoS* issues are associated with data acquisition, and they attempt to avoid the fact that the delivery of massive data (data without processing) directly from the source negatively impacts network traffic and the energy consumption of mobile devices. Concerning *scalability*, authors assess the capability of the infrastructure in terms of adaptability to an increasing number of sensing tasks and terminals to determine if it is able to adapt to both small and large deployments. Under the scalability concept we have also considered the elasticity of these services (middleware) to manage changes.

Notice that Table 5 is clustered into four different parts according to the scope of the proposal: T refers to those proposals only addressing transmission issues, C refers to proposals centered on the client side, S refers to proposals centered on the server side, and G refers to global solutions.

Concerning communication technologies used, a large number of proposals relied on WiFi and Cellular communications, although we can also find proposals that rely instead on Bluetooth due to its flexibility and low consumption features. Additionally, we find that most solutions opted for either a centralized topology or a distributed topology, with only a reduced number of proposals choosing a hybrid approach. Regarding the networking approach, most solutions adopt RESTful services based on HTTP or make use of the XML format.

Focusing now on the performance metrics addressed by each proposal, most solutions made a generic performance analysis (delivery delay, data rate, etc.). However, very few solutions addressed QoS and scalability issues. For instance, we can find solutions such as GCM [9] that address scalable services in the cloud, others that address scalability in the context of the Publish/Subscriber paradigm [8], and yet others that relate it to broker collaboration [54], but none of these actually assess performance in the scalability context.

Regarding proposals evaluating Quality of Service performance, they typically perform such evaluation in terms of task allocation and coverage optimization in the target area. For instance, proposals such as JoinPolices [58] evaluate the impact and the performance of task execution based on incentive policies, while QoSMCS [69] defines an ad hoc method for the evaluation of QoS in the context of mobile crowdsensing services based on Petri networks.

Finally, regarding scalability, solutions such as PRISM [24] assess the performance achieved through comparison against other solutions. Neighbor [60] measures message diffusion performance between the mobile nodes and the data collection server. Lastly, Medusa [25] proposes a prototype able to measure in runtime the time taken to perform several individual steps associated with task executions, both on the cloud and the smartphone.

5. Open Research Issues

Based on the analysis presented in the previous sections, it becomes clear that, despite the many advancements introduced in the mobile crowdsensing field in recent years, there are still several issues that should be properly addressed for solutions to become more effective and therefore gain more widespread acceptance.

At the user's side, it becomes clear that the sensing tasks should not become a burden. Thus, any external sensors, if required at all, should be small and lightweight, have a low power consumption, and have an elegant and stylish look. Ideally, additional sensors should be progressively integrated into new smartphones either directly from the manufacturer or as pluggable modules. Power and network resource consumption are also an issue, and so smart algorithms able to correctly determine the best sampling times while avoiding intensive CPU usage are required; in terms of network resources, peer-to-peer data delivery combined with smart network selection can help at avoiding to deplete radio resources and having a negative impact in terms of traffic quotas.

From a more global perspective, further studies are required in order to assess the scalability and the QoS support of the different proposals. In particular, their impact on the end-to-end communications infrastructures should be thoroughly studied. Additionally, new algorithms should be developed to improve the processes of data collection and analysis.

6. Conclusions

Crowdsensing solutions that benefit from smartphones are proliferating due to the multiple advantages offered. Thus, it becomes important to provide a unified view of the different author contributions to detect the major areas of improvement. In this paper we address this challenge through a survey that provides the reader with an extensive review of existing smartphone-based solutions in the field of mobile crowdsensing. We start by presenting a novel reference architecture where we identify the major components at client side, server side, and the communications level. Based on our proposed architecture, we then proceed to classify the different proposals, focusing separately on the client, the server, and the communications part of each solution.

Our extensive literature analysis has shown that most proposals provide some degree of adaptability to different work environments. In addition, we found that technologies

TABLE 5: Classification of the different data transmission solutions according to the proposed architecture.

Publication	Scope	Communication technologies	Topology	Networking approach	Protocols/format	Performance metrics		
						Generic	QoS	Scalability
Neighbor [60]		WiFi/Bluetooth/Cellular	Hybrid	Algorithm/one simulated		X		X
Coupon [64]		WiFi/Bluetooth	Hybrid	ERF and BSWF	TCP	X		
Anonymity [66]		WiFi	Centralized	Slot reservations stage and data submission stage	TCP	X		
NeCorpia [68]	T[1]	WiFi	Distributed-hybrid	Network encoding format/Gaussian eliminations	TCP	X		
QoSMCS [69]			Hybrid	Markovian stochastic Petri NeT	HTTP-TCP		X	
RemoteCloud [73]		WiFi/Cellular	Distributed	MANET or device-to-device (D2D)	TCP	X		
EffSense [76]		Bluetooth/WiFi/Cellular	Hybrid	Publish-Subscriber	HTTP-JSON	X		
Context [79]		Bluetooth	Hybrid	Simulated	TCP	X		
PRISM [24]		WiFi/Bluetooth/Cellular	Centralized	PRISM client and Sandbox				X
DAM4GSN [50]		WiFi/Cellular	Hybrid	GSN (SOAP and RESTful web services)	HTTP-XML			
StressSense [14]		WiFi/Cellular	Distributed	MatLab				
Usense [53]		WiFi/Cellular	Distributed	SOAP web services (RPC)	HTTP-XML	X		
REPSense [11]		WiFi/Cellular	Distributed					
PLUS [13]		WiFi	Centralized	Web services				
MOSDEN [28]		WiFi/Cellular	Distributed	GSN (SOAP and RESTful web services)	HTTP-XML	X		
SenseDroid [54]		WiFi/Bluetooth/Cellular	Hybrid	Provides libraries and APIs				
AlienvsMobile [39]		WiFi	Centralized	NS2 simulated	HTTP	X		
CUPUS [48]	C[2]	WiFi	Distributed	Publish-Subscriber (MQTT-MOSQUITO)	HTTP-XML	X		
BLISS [20]				Simulated				
SPREAD [56]		S/E		Simulated				
MAP+ [57]		WiFi/Cellular	Centralized					
ADTS [65]		WiFi/Cellular	Hybrid	Simulated				
FlierMeet [70]		WiFi	Distributed	—				
MDPPs [74]		WiFi/Bluetooth/Cellular	Distributed	SOAP web services	HTTP-XML			
EasyHarvest [45]		WiFi/Cellular	Centralized	RESTful web services	HTTP-XML			
Sahyog [78]		WiFi/Cellular	Centralized	Publish-Subscriber	HTTP-JSON	X		
WiFIScout [62]		WiFi	Centralized		HTTP			
Anonysense [6]	S[3]	WiFi/Bluetooth	Hybrid	SOAP web services (SMTP/SSL)	HTTP(S)-XML			
Pogo [49]		WiFi/Cellular	Centralized	Publish-Subscriber (XMPP Openfire)	HTTP-JSON	X		

Table 5: Continued.

Publication	Scope	Communication technologies	Topology	Networking approach	Protocols/format	Performance metrics		
						Generic	QoS	Scalability
MECA [26]		WiFi/Cellular	Distributed					
SmartCity [10]			Distributed	Publish-Subscriber (XMPP)	HTTP-XML			
ILR [51]		WiFi/Bluetooth	Centralized	ILR Algorithm/simulated NS2				
NoiseNYC [16]			Distributed					
BeC3 [46]		WiFi	Distributed	RESTful web services (COAP)	HTTP-XML			
McSense [12]		WiFi/Bluetooth	Distributed	RESTful web services	HTTP-XML	X		
SaaS [44]		WiFi/Bluetooth/Cellular	Distributed	Web services	HTTP			
GPS-Less [55]		WiFi	Distributed	Approximation algorithms	HTTP	X		
JoinPolices [58]		Budget	Distributed	Simulated MatLab		X	X	
MCSaaS [43]		WiFi/Cellular	Distributed	RESTful web services	HTTP-XML	X		
GROPING [61]		WiFi	Centralized					
Qoata [67]			Centralized	Simulated				
MCS game [72]		WiFi/Cellular	Centralized					
RuPS [75]		WiFi/Cellular	Centralized	RESTful web services	HTTP			
NoizCrowd [18]			Distributed					
CrowdRecruiter [59]		WiFi/Cellular	Centralized	Simulated				
QoS-Constrained [63]			Hybrid	Simulated				
PLP [77]			Hybrid					
MoreWithLess [80]			Distributed					
Sparse [81]			Distributed					
Ear-Phone [5]		WiFi/Cellular	Centralized					
Medusa [25]		WiFi/Cellular	Distributed	MedScript-SMS-MMS	HTTPS-XML	X		X
CrowITS [9]		WiFi/Cellular	Centralized	RESTful web services (C2DM)	HTTP-JSON			
CAROM [27]		WiFi/Bluetooth/Cellular	Hybrid	GSN (SOAP and RESTful web services)	HTTP	X		
SoundOfTheCity [7]		WiFi/Cellular	Centralized	SOAP web services (RTMP)	HTTP-XML			
MoPS [8]		WiFi/Cellular	Distributed	RESTful web services (GCM)	HTTP-XML			
Vita [15]		WiFi/Cellular	Hybrid	RESTful web services	HTTP-XML	X		X
Matador [52]	G[4]	WiFi/Cellular	Centralized	RESTful web services	HTTP-XML			
SenSocial [19]		WiFi/Bluetooth/Cellular	Centralized	Publish-Subscriber (MQTT-MOSQUITO)	HTTP-JSON	X		X
MCSinSpace [21]		WiFi/Cellular	Distributed	RESTful web services	HTTP			
LineKing [22]		WiFi	Centralized	SOAP web services	HTTP			
Ecosystem [47]		WiFi/Cellular	Distributed	RESTful web services	HTTP-XML	X		
TYT [17]		WiFi/Bluetooth/Cellular	Centralized	RESTful web services	HTTP			
PRESM [23]		WiFi/Cellular	Centralized					
SmartRoad [71]		WiFi/Cellular	Centralized	SOAP web services	HTTP-XML			

[1]T: proposals addressing transmission issues.
[2]C: proposals centered on the client side.
[3]S: proposals centered on the server side.
[4]G: global solutions.

and algorithms applicable at both client and server sides have evolved significantly and are often available in a modular format, allowing other researchers to include them in their proposed solutions. Concerning improvements in the data capture process itself, we found that the main issues are the software adaptability to different types of sensors and reducing power consumption. At the server side, the most critical improvements include task generation language and procedures, the analysis and storage of data, and providing an adequate interface for task management by administrators. The communication between client and server usually makes use of technologies like SOAP and RESTful, and most solutions support Publish/Subscriber models.

Overall, we believe that mobile crowdsensing is now achieving its maturity, being a widespread adoption of crowdsensing solutions expectable in the next few years.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was partially supported by the Ministerio de Economía y Competitividad, Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, Proyectos I+D+I 2014, Spain, under Grant TEC2014-52690-R, the "Universidad Laica Eloy Alfaro de Manabí-ULEAM," and the "Programa de Becas SENESCYT de la República del Ecuador."

References

[1] C. to connecting the world ITU, "Ict facts and figures, the world in 2015," http://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx.

[2] J. A. Burke, D. Estrin, M. Hansen et al., *Participatory Sensing*, Center for Embedded Network Sensing, 2006.

[3] A. T. Campbell, S. B. Eisenman, N. D. Lane et al., "The rise of people-centric sensing," *IEEE Internet Computing*, vol. 12, no. 4, pp. 12–21, 2008.

[4] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[5] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: an end-to-end participatory urban noise mapping system," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)*, pp. 105–116, ACM Press, New York, NY, USA, April 2010.

[6] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos, "AnonySense: a system for anonymous opportunistic sensing," *Pervasive and Mobile Computing*, vol. 7, no. 1, pp. 16–30, 2011.

[7] L. Ruge, B. Altakrouri, and A. Schrader, "Sound of the city—continuous noise monitoring for a healthy city," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 670–675, March 2013.

[8] I. Podnar Zarko, A. Antonic, and K. Pripužic, "Publish/subscribe middleware for energy-efficient mobile crowdsensing," in *Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp '13)*, pp. 1099–1110, Zurich, Switzerland, September 2013.

[9] K. Ali, D. Al-Yaseen, A. Ejaz, T. Javed, and H. S. Hassanein, "CrowdITS: crowdsourcing in intelligent transportation systems," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '12)*, pp. 3307–3311, IEEE, Shanghai, China, April 2012.

[10] R. Szabo, K. Farkas, M. Ispany et al., "Framework for smart city applications based on participatory sensing," in *Proceedings of the 4th IEEE International Conference on Cognitive Infocommunications (CogInfoCom '13)*, pp. 295–300, Budapest, Hungary, December 2013.

[11] G. Liu, M. Iwai, Y. Tobe, and K. Sezaki, "REPSense: on-line sensor data reduction while preserving data diversity for mobile sensing," in *Proceedings of the IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '13)*, pp. 584–591, October 2013.

[12] G. Cardone, L. Foschini, P. Bellavista et al., "Fostering participaction in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 112–119, 2013.

[13] A. Khan, S. K. A. Imon, and S. K. Das, "An energy efficient framework for localization and coverage in participatory urban sensing," in *Proceedings of the 39th Annual IEEE Conference on Local Computer Networks (LCN '14)*, pp. 193–201, Edmonton, Canada, September 2014.

[14] H. Lu, D. Frauendorfer, M. Rabbi et al., "StressSense: detecting stress in unconstrained acoustic environments using smartphones," in *Proceedings of the ACM Conference on Ubiquitous Computing (UbiComp '12)*, pp. 351–360, ACM, Pittsburgh, Pa, USA, September 2012.

[15] X. Hu, T. H. Chu, H. C. Chan, and V. C. Leung, "Vita: a crowdsensing-oriented mobile cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 148–165, 2013.

[16] Y. Zheng, T. Liu, Y. Wang, Y. Zhu, Y. Liu, and E. Chang, "Diagnosing New York city's noises with ubiquitous data," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, pp. 715–725, ACM Press, September 2014.

[17] R. Pryss, M. Reichert, B. Langguth, and W. Schlee, "Mobile crowd sensing services for tinnitus assessment, therapy, and research," in *Proceedings of the IEEE International Conference on Mobile Services (MS '15)*, vol. 32, no. 2, pp. 352–359, New York, NY, USA, June 2015.

[18] M. Wisniewski, G. Demartini, A. Malatras, and P. Cudré-Mauroux, "NoizCrowd: a crowd-based data gathering and management system for noise level data," in *Mobile Web Information Systems*, vol. 8093 of *Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 172–186, Springer, 2013.

[19] A. Mehrotra, V. Pejovic, and M. Musolesi, "SenSocial: a middleware for integrating online social networks and mobile sensing data streams," in *Proceedings of the 15th International Middleware Conference (Middleware '14)*, pp. 205–216, ACM Press, Bordeaux, France, December 2014.

[20] K. Han, C. Zhang, and J. Luo, "BLISS: budget Limited robust crowdsensing through online learning," in *Proceedings of the 11th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON '14)*, pp. 555–563, July 2014.

[21] V. Pankratius, F. Lind, A. Coster, P. Erickson, and J. Semeter, "Mobile crowd sensing in space weather monitoring: the mahali

project," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 22–28, 2014.

[22] M. F. Bulut, M. Demirbas, and H. Ferhatosmanoglu, "LineKing: coffee shop wait-time monitoring using smartphones," *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2045–2058, 2014.

[23] X. Wu, P. Yang, S. Tang, X. Zheng, and Y. Xiong, "Privacy preserving RSS map generation for a crowdsensing network," *IEEE Wireless Communications*, vol. 22, no. 4, pp. 42–48, 2015.

[24] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: platform for remote sensing using smartphones," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*, p. 63, ACM Press, San Francisco, Calif, USA, June 2010.

[25] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan, "Medusa: a programming framework for crowd-sensing applications," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, pp. 337–350, ACM, Ambleside, UK, June 2012.

[26] F. Ye, R. Ganti, R. Dimaghani, K. Grueneberg, and S. Calo, "MECA: mobile edge capture and analysis middleware for social sensing applications," in *Proceedings of the 21st Annual Conference on World Wide Web (WWW '12)*, pp. 699–702, ACM Press, Lyon, France, April 2012.

[27] A. Zaslavsky, P. P. Jayaraman, and S. Krishnaswamy, "ShareLikesCrowd: mobile analytics for participatory sensing and crowdsourcing applications," in *Proceedings of the IEEE 29th International Conference on Data Engineering Workshops (ICDEW '13)*, pp. 128–135, April 2013.

[28] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, and P. Christen, "MOSDEN: an internet of things middleware for resource constrained mobile devices," in *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS '14)*, pp. 1053–1062, IEEE, Waikoloa, Hawaii, USA, January 2014.

[29] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[30] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad, "Mobile phone sensing systems: a survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 402–427, 2013.

[31] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4W1H in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 42–48, 2014.

[32] X. Zhang, Z. Yang, W. Sun et al., "Incentives for mobile crowd sensing: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 54–67, 2016.

[33] L. G. Jaimes, I. J. Vergara-Laurens, and A. Raij, "A survey of incentive techniques for mobile crowd sensing," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 370–380, 2015.

[34] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu, "Quality of information aware incentive mechanisms for mobile crowd sensing systems," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '15)*, pp. 167–176, ACM, Hangzhou, China, June 2015.

[35] J. Wang, J. Tang, D. Yang, E. Wang, and G. Xue, "Quality-aware and fine-grained incentive mechanisms for mobile crowdsensing," in *Proceedings of the IEEE 36th International Conference on Distributed Computing Systems (ICDCS '16)*, pp. 354–363, IEEE, Nara, Japan, June 2016.

[36] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Inception: incentivizing privacy-preserving data aggregation for mobile crowd sensing systems," in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 341–350, ACM, Paderborn, Germany, July 2016.

[37] B. Guo, Z. Wang, Z. Yu et al., "Mobile crowd sensing and computing," *ACM Computing Surveys*, vol. 48, no. 1, article 7, 2015.

[38] F. Calabrese, L. Ferrari, and V. D. Blondel, "Urban sensing using mobile phone network data: a survey of research," *ACM Computing Surveys*, vol. 47, no. 2, Article ID 2655691, pp. 1–20, 2014.

[39] M. Talasila, R. Curtmola, and C. Borcea, "Alien vs. Mobile user game: fast and efficient area coverage in crowdsensing," in *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services (MobiCASE '14)*, pp. 65–74, November 2014.

[40] E. Kwan and J. R. Getta, *Design and implementation of data stream processing applications [Dissertation]*, vol. 4611, pp. 1–142, 2010.

[41] AWS — Elastic compute cloud (EC2), https://aws.amazon.com/ec2/.

[42] Google Cloud Platform, https://cloud.google.com/.

[43] G. Merlino, S. Arkoulis, S. Distefano, C. Papagianni, A. Puliafito, and S. Papavassiliou, "Mobile crowdsensing as a service: a platform for applications on top of sensing Clouds," *Future Generation Computer Systems*, vol. 56, pp. 623–639, 2016.

[44] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a service: challenges, solutions and future directions," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3733–3741, 2013.

[45] M. Katsomallos and S. Lalis, "EasyHarvest: supporting the deployment and management of sensing applications on smartphones," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS '14)*, pp. 80–85, March 2014.

[46] S. Cherrier, I. Salhi, Y. M. Ghamri-Doudane, S. Lohier, and P. Valembois, "BeC 3: behaviour crowd centric composition for IoT applications," *Mobile Networks and Applications*, vol. 19, no. 1, pp. 18–32, 2014.

[47] X. Hu, X. Li, E. C.-H. Ngai, V. C. M. Leung, and P. Kruchten, "Multidimensional context-aware social network architecture for mobile crowdsensing," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 78–87, 2014.

[48] A. Antonic, K. Roankovic, M. Marjanovic, K. Pripuic, and I. P. Arko, "A mobile crowdsensing ecosystem enabled by a cloud-based publish/subscribe middleware," in *Proceedings of the 2nd International Conference on Future Internet of Things and Cloud (FiCloud '14)*, pp. 107–114, IEEE, Barcelona, Spain, August 2014.

[49] N. Brouwers and K. Langendoen, "Pogo, a middleware for mobile phone sensing," in *Proceedings of the 13th International Middleware Conference*, pp. 21–40, Montreal, Canada, 2012.

[50] C. Pereral, A. Zaslavsky, P. Christen, A. Salehi, and D. Georgakopoulos, "Capturing sensor data from mobile phones using global sensor network middleware," in *Proceedings of the IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '12)*, pp. 24–29, IEEE, Sydney, Australia, September 2012.

[51] M. Talasila, R. Curtmola, and C. Borcea, "Improving location reliability in crowd sensed data with minimal efforts," in *Proceedings of the 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC '13)*, pp. 1–8, April 2013.

[52] I. Carreras, D. Miorandi, A. Tamilin, E. R. Ssebaggala, and N. Conci, "Matador: Mobile task detector for context-aware

crowd-sensing campaigns," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops*, pp. 212–217, IEEE, San Diego, Calif, USA, March 2013.

[53] V. Agarwal, N. Banerjee, D. Chakraborty, and S. Mittal, "USense—a smartphone middleware for community sensing," in *Proceedings of the 14th International Conference on Mobile Data Management (MDM '13)*, vol. 1, pp. 56–65, IEEE, Milan, Italy, June 2013.

[54] S. Sarma, N. Venkatasubramanian, and N. Dutt, "Sense-making from distributed and mobile sensing data," in *Proceedings of the 51st Annual Design Automation Conference on Design Automation Conference (DAC '14)*, pp. 1–6, ACM Press, San Francisco, Calif, USA, June 2014.

[55] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Leveraging GPS-less sensing scheduling for green mobile crowd sensing," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 328–336, 2014.

[56] L. G. Jaimes, I. Vergara-Laurens, and A. Chakeri, "SPREAD, a crowd sensing incentive mechanism to acquire better representative samples," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM '14)*, pp. 92–97, March 2014.

[57] H. Aly, A. Basalamah, and M. Youssef, "Map++: a crowd-sensing system for automatic map semantics identification," in *Proceedings of the 11th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON '14)*, pp. 546–554, July 2014.

[58] C. M. Angelopoulos, S. Nikoletseas, T. P. Raptis, and J. D. P. Rolim, "Characteristic utilities, join policies and efficient incentives in Mobile Crowdsensing Systems," in *Proceedings of the 7th IFIP/IEEE Wireless Days Conference (WD '14)*, pp. 1–6, Rio de Janeiro, Brazil, November 2014.

[59] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, pp. 703–714, ACM Press, September 2014.

[60] T. Higuchi, H. Yamaguchi, T. Higashino, and M. Takai, "A neighbor collaboration mechanism for mobile crowd sensing in opportunistic networks," in *Proceedings of the 1st IEEE International Conference on Communications (ICC '14)*, pp. 42–47, Sydney, Australia, June 2014.

[61] C. Zhang, K. P. Subbu, J. Luo, and J. Wu, "GROPING: geomagnetism and crowdsensing powered indoor navigation," *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 387–400, 2015.

[62] F.-J. Wu and T. Luo, "WiFiScout: a crowdsensing WiFi advisory system with gamification-based incentive," in *Proceedings of the 11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS '14)*, pp. 533–534, October 2014.

[63] Z. Wang, D. Huang, H. Wu, Y. Deng, A. Aikebaier, and Y. Teranishi, "QoS-constrained sensing task assignment for mobile crowd sensing," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM '14)*, pp. 311–316, Austin, Tex, USA, December 2014.

[64] D. Zhao, H. Ma, S. Tang, and X.-Y. Li, "COUPON: a cooperative framework for building sensing maps in mobile opportunistic networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 392–402, 2015.

[65] M. H. Cheung, R. Southwell, F. Hou, and J. Huang, "Distributed time-sensitive task selection in mobile crowdsensing," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '15)*, pp. 157–166, ACM Press, Hangzhou, China, June 2015.

[66] Y. Yao, L. T. Yang, and N. N. Xiong, "Anonymity-based privacy-preserving data reporting for participatory sensing," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 381–390, 2015.

[67] C. Zhou, C.-K. Tham, and M. Motani, "QOATA: QoI-aware task allocation scheme for mobile crowdsensing under limited budget," in *Proceedings of the 10th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '15)*, pp. 1–6, Singapore, April 2015.

[68] C. Greco, M. Kieffer, and C. Adjih, "NeCoRPIA: network coding with random packet-index assignment for mobile crowdsensing," in *Proceedings of the IEEE International Conference on Communications (ICC '15)*, pp. 6338–6344, IEEE, June 2015.

[69] S. Distefano, F. Longo, and M. Scarpa, "QoS assessment of mobile crowdsensing services," *Journal of Grid Computing*, vol. 13, no. 4, pp. 629–650, 2015.

[70] B. Guo, H. Chen, Z. Yu, X. Xie, S. Huangfu, and D. Zhang, "FlierMeet: a mobile crowdsensing system for cross-space public information reposting, tagging, and sharing," *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2020–2033, 2015.

[71] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher, "Smartroad: smartphone-based crowd sensing for traffic regulator detection and identification," *ACM Transactions on Sensor Networks*, vol. 11, no. 4, article 55, pp. 1–27, 2015.

[72] L. Xiao, J. Liu, Q. Li, and H. V. Poor, "Secure mobile crowdsensing game," in *Proceedings of the IEEE International Conference on Communications (ICC '15)*, pp. 7157–7162, June 2015.

[73] C. Song, M. Liu, and X. Dai, "Remote cloud or local crowd: communicating and sharing the crowdsensing data," in *Proceedings of the IEEE 5th International Conference on Big Data and Cloud Computing (BDCloud '15)*, pp. 293–297, Dalian, China, August 2015.

[74] S. Sathe, T. Sellis, and K. Aberer, "On crowdsensed data acquisition using multi-dimensional point processes," in *Proceedings of the 31st IEEE International Conference on Data Engineering Workshops (ICDEW '15)*, pp. 124–128, IEEE, Seoul, South Korea, April 2015.

[75] J. Mohite, Y. Karale, P. Gupta, S. Kulkarni, B. Jagyasi, and A. Zape, "RuPS: rural participatory sensing with rewarding mechanisms for crop monitoring," in *Proceedings of the 13th IEEE International Conference on Pervasive Computing and Communication (PerCom Workshops '15)*, pp. 378–383, St. Louis, Mo, USA, March 2015.

[76] L. Wang, D. Zhang, Z. Yan, H. Xiong, and B. Xie, "effSense: a novel mobile crowd-sensing framework for energy-efficient and cost-effective data uploading," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 12, pp. 1549–1563, 2015.

[77] S. Zhang, Q. Ma, T. Zhu et al., "PLP: protecting location privacy against correlation-analysis attack in crowdsensing," in *Proceedings of the 44th International Conference on Parallel Processing (ICPP '15)*, pp. 111–119, Beijing, China, September 2015.

[78] G. Bajaj and P. Singh, "Sahyog: a middleware for mobile collaborative applications," in *Proceedings of the 7th International Conference on New Technologies, Mobility and Security (NTMS '15)*, pp. 1–5, Paris, France, July 2015.

[79] P. Nguyen and K. Nahrstedt, "Context-aware crowd-sensing in opportunistic mobile social networks," in *Proceedings of*

the IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems (MASS '15), pp. 477–478, IEEE, Dallas, Tex, USA, October 2015.

[80] L. Xu, X. Hao, N. D. Lane, X. Liu, and T. Moscibroda, "More with less: lowering user burden in mobile crowdsourcing through compressive sensing," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*, pp. 659–670, ACM, Osaka, Japan, September 2015.

[81] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed, "Sparse mobile crowdsensing: challenges and opportunities," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 161–167, 2016.

[82] L. Wang, D. Zhang, A. Pathak et al., "CCS-TA: quality-guaranteed online task allocation in compressive crowdsensing," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*, pp. 683–694, ACM, Osaka, Japan, September 2015.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 7967249, 13 pages
http://dx.doi.org/10.1155/2016/7967249

Research Article

SenSafe: A Smartphone-Based Traffic Safety Framework by Sensing Vehicle and Pedestrian Behaviors

Zhenyu Liu, Mengfei Wu, Konglin Zhu, and Lin Zhang

Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, China

Correspondence should be addressed to Lin Zhang; zhanglin@bupt.edu.cn

Received 10 June 2016; Revised 7 September 2016; Accepted 28 September 2016

Academic Editor: Francisco Martínez

Traffic accident involving vehicles is one of the most serious problems in the transportation system nowadays. How to detect dangerous steering and then alarm drivers in real time is a problem. What is more, walking while using smartphones makes pedestrian more susceptible to various risks. Although dedicated short range communication (DSRC) provides the way for safety communications, most of vehicles have not been deployed with DSRC components. Even worse, DSRC is not supported by the smartphones for vehicle-to-pedestrian (V2P) communication. In this paper, a smartphone-based framework named SenSafe is developed to improve the traffic safety. SenSafe is a framework which only utilizes the smartphone to sense the surrounding events and provides alerts to drivers. Smartphone-based driving behaviors detection mechanism is developed inside the framework to discover various steering behaviors. Besides, the Wi-Fi association and authentication overhead is reduced to broadcast the compressed sensing data using the Wi-Fi beacon to inform the drivers of the surroundings. Furthermore, a collision estimation algorithm is designed to issue appropriate warnings. Finally, an Android-based implementation of SenSafe framework has been achieved to demonstrate the application reliability in real environments.

1. Introduction

Traffic safety becomes one of the serious problems in the transportation systems. As the number of the vehicles is growing, the levels of traffic accidents have increased significantly. Not only vehicles, but also pedestrians and cyclists are facing the safety threats from traffic accidents. According to [1], traffic accidents resulted in more than 500 thousand deaths and 14 million injures worldwide by the end of May in 2016. One of the main reasons for traffic accidents is that the drivers cannot notice behaviors of surrounding vehicles on time. Using smartphone while walking is increasingly apparent in our society, and when people walk with their attention on smartphones and distracted from the scenery, potential accidents are in front of them. To reduce the traffic accidents, it is necessary to assist the drivers to have a better understanding of the surrounding environments including the coming vehicles and adjacent pedestrians and cyclists.

Dedicated short range communication (DSRC) [2] has been accepted as the most promising approach to enhance the transportation safety. It consists of a set of vehicles equipped with the on-board unit (OBU), and a set of road side units (RSUs) along the roads, and aims to provide reliable and low latency vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications. However, due to additional effort for DSRC deployment, most of vehicles have not been equipped with OBU for DSRC. Besides, DSRC is not supported by the general smartphones for vehicle-to-pedestrian (V2P) communication.

To enable the safety of transportations, many strategies are proposed. There are some works using the vehicle-mounted device for safety assistance driving system. Cameras and radars are used to monitor the driving behaviors and alert the dangerous distances between vehicles to provide technical support for the auto manufacturers. Although the cost of cameras and radars based safety technology is decreasing, these safety technologies have not been deployed in economy vehicles. It still needs time before the majority of vehicles are deployed with these safety technologies. Furthermore, due to lack of communication, vehicles need to make decisions by themselves.

With the widespread use of the smartphone, most of drivers and pedestrians have smartphones, which provides the potential to use them for the enhancement of traffic safety. Nowadays, smartphones are progressively equipped with functional sensors (e.g., accelerometer, gyroscope, GPS, camera, etc.), and these sensors can be applied to recognize and monitor various vehicle behaviors. Additionally, they contain communication resources and enable the quick deployment of new applications [3].

Sensor technology available in smartphones enables the monitoring of vehicle mobility behaviors, including lane change, turns, acceleration, and brake. If such information can be sensed and transmitted to other vehicles or pedestrians, they can be potentially used in collision prevention, early crash detection, congestion avoidance, and so forth. Smartphones have abundant communication resources, such as Bluetooth, Wi-Fi, and 3G/LTE. However, using them to construct appropriate networks for efficient information sharing in transportation systems is difficult. In particular, the transmission range of the Bluetooth cannot satisfy the requirements of the communication. Wi-Fi has the procession of authentication and association before data transmission and is more suitable for 1-to-N communication. 3G/LTE needs the assistance of the base station.

In this paper, we propose SenSafe, a smartphone-based framework for traffic safety by sensing, communication, and alerting, which overcomes the limitations of requesting additional professional equipment inherent in the existing approaches.

SenSafe is a novel framework which only utilizes the smartphone to improve the transportation safety. It detects and reports vehicle's events based on the sensing data collected from steering behaviors in urban area. The changes in the angle of vehicle heading and the corresponding displacement during a steering maneuver are calculated for driving behavior classification. Driving behaviors are classified into different types, including turn, lane change, acceleration, and brake. Beacon-based communication is provided to exchange the driving information. Surrounding environment reminding and collision forewarning are provided based on the data from the surroundings.

We highlight our main contributions as follows:

(i) We propose a framework, SenSafe, which only uses the smartphone to sense the driving behavior, exchange the driving information, and afford the reminding and alerting.

(ii) We provide the detection and differentiation of various driving behaviors by only utilizing the smartphone's built-in sensors in urban area. We analyze the sensing data collected from urban area and find that vehicles cannot always make turns and lane changes smoothly owing to the influence of surrounding vehicles and pedestrians. The temporary interruptions are taken into consideration to improve the accuracy of the driving behaviors detection.

(iii) We furnish the beacon-based communication, which modifies the service set identifier (SSID) field of Wi-Fi beacon frame to carry the driving behavior, position, and mobility information. A data compression and decompression method is provided to make use of the SSID field. Event-driven communication is utilized to provide the surrounding drivers with immediate reminding. Ordinary promptings of the surrounding vehicles' behaviors and safety alerts of the coming collisions are provided to the drivers.

(iv) An Android-based implementation of SenSafe framework has been achieved to demonstrate the evaluation results and application reliability in real environments.

The rest of this paper is organized as follows. In Section 2, a brief overview of the related works is provided. In Section 3, the framework overview of SenSafe is introduced. The details for three modules of SenSafe are explained in Sections 4–6. The implementation, performance evaluation, and experiment results are shown in Section 7. Finally, this paper is concluded in Section 8.

2. Related Works

There are some works to detect the vehicle behavior. Some works use the fixed vehicle-mounted devices to monitor the vehicle behavior. Mobileye uses cameras and radars to provide the technical support for the auto manufacturers [4]. Although the cost of vehicle safety technology is decreasing, these safety technologies have not been deployed in economy vehicles. It still needs time before the majority of vehicles are deployed with these safety technologies. With the wide spread of smartphones, smartphone solutions can be used in vehicles, and there are some works focusing on using smartphones to assist drivers.

Drivea [5], iOnRoad [6], and Augmented Driving [7] are apps which have capability of detecting lane departures and warn drivers when the distance to the front vehicle is too close. CarSafe [8] alerts distracted drivers using dual cameras on smartphones, one for detecting driver state and the other for tracking road conditions. Although camera-based systems have the functionality in assisting the driver, they have limitations in terms of computational overhead and requirement of image quality and work worse at night and with bad lighting conditions.

Camera-free sensors in the smartphone have been utilized in traffic regulator detection [9], localization [10], transportation mode classification [11], vehicle speed estimation [12], and the driving behavior detection [10, 13–15]. In [13], smartphone sensing of vehicle dynamics is utilized to determine driver phone use. Inertial measurement unit (IMU) sensors including accelerometers and gyroscopes of the smartphone are used to capture differences in centripetal acceleration due to vehicle dynamics. In [14], three algorithms which detect driving events using motion sensors embedded on a smartphone are proposed. The first detection algorithm is based on data collected from GPS receiver. The second and third detection algorithms utilize accelerometer

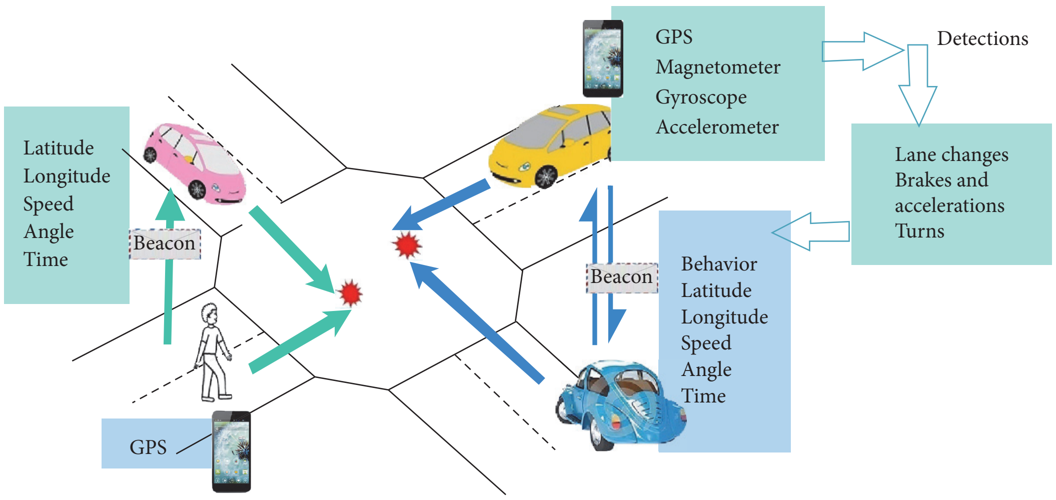


Figure 1: Application scenarios.

sensor and use pattern matching technique to detect driving events. In [15], a vehicle steering detection middleware called V-Sense is developed on commodity smartphones by only utilizing nonvision sensors on the smartphone. Algorithms are designed for detecting and distinguishing various vehicle maneuvers, including lane changes, turns, and driving on curvy roads. However, the paper only considers the smooth driving behaviors. Once there is a sudden brake because of the surrounding vehicles and pedestrians, it cannot work well. In [10], embedded sensors in smartphones are utilized to capture the patterns of lane change behaviors, and this paper on the driving environment of highway.

There are some works for the communication. Some companies and researchers focus on using the DSRC for V2V communication [16, 17]. However, all of these works need special devices for communications, which have not been deployed in most of vehicles. Some works use the communication resources of smartphone to avoid using extra devices. The master access point disseminates information to the rest of the units through Wi-Fi of smartphones [18], and it is suitable for the solid groups. But, for the dynamics groups, Wi-Fi has the processions of authentication and association and is usually for the 1-to-N communications. For the Wi-Fi Direct, the device discovery phase may take over ten seconds in some cases [19]. The authors modify the SSID of Wi-Fi beacons to store the location and speed of the smartphone for alert [20]. However, they do not consider the driving behaviors of the vehicles. Besides, as the access points (APs) can only broadcast beacons and the clients can only receive the beacons, how to achieve the bidirectional communications is not considered.

Different from the previous works, we propose an integrated smartphone-based framework SenSafe for traffic safety considering sensing vehicle behaviors. This framework is only based on smartphone and thus can be easily deployed due to the widespread use of the smartphone. Vehicle behavior sensing is one of the key points in this framework. To improve the robustness and the accuracy of the vehicle behavior sensing, nonvision sensors are utilized to reduce the influence of the image quality, and unsmooth driving behaviors are considered as there may be some brakes in the procession of turns. Considering that only AP broadcasts beacons and cannot receive beacons from the other APs, an event-driven communication method is proposed to achieve the bidirectional communications. Finally, the reminder events are divided into ordinary prompting of the surrounding vehicles' behaviors and safety alert of the coming collisions for the drivers.

3. Framework Overview

This section describes the high level overview of the framework SenSafe that we propose for traffic safety. SenSafe is focused on the vehicles and pedestrian safety. One of the main reasons for the traffic accident is that the drivers cannot notice the behaviors of surrounding vehicles and pedestrians on time. SenSafe considers using the smartphone which is widespread to improve the traffic safety without any cost of installing new device.

As is shown in Figure 1, SenSafe uses the sensors from the smartphone to collect the data about the vehicle mobility. After that, these data can be used to obtain the driving behavior of the vehicles. Using the information transferred from the surroundings, drivers can have a better understanding of the environments. Besides, the smartphone of the pedestrian obtains the moving information from the GPS and broadcasts it to the vehicles to enhance the vulnerable pedestrian safety.

System architecture is shown in Figure 2. The framework is made of three components: driving behaviors detection module, beacon-based communication module, and collision forewarning module. The driving behaviors detection module considers the output of motion sensors and GPS. Acceleration, braking, turns, and lane changes are detected using the proposed algorithm. Beacon-based communication module compresses the sensed data into the SSID of the beacon for communication. Collision forewarning module uses the data from the surrounding to remind the driver of the driving behaviors and finds out the vehicles which have

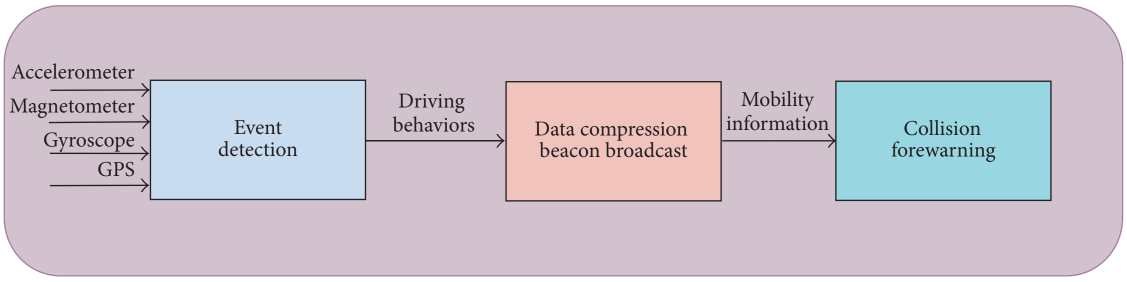


Figure 2: System Architecture.

threat to the current vehicle and pedestrians who might be threatened.

4. Driving Behaviors Detection

During a single trip, the smartphone collects input data from motion sensors and GPS, and the driving behavior detection system decides the type of the driving behaviors. This section details the design and functionalities of driving behavior detection. First, we describe how the sensors of smartphones are utilized to determine whether the vehicle is making acceleration, brake, turn, or changing lane. Then, we show how SenSafe classifies such different vehicle driving behaviors based on the detection results. The Android-based smartphone is used as our target platform to collect raw data from four on-board sensors, 3-axis accelerometer, magnetometer, gyroscope, and GPS receiver.

4.1. Coordinate Alignment. Given the direction of gravity on the smartphone body coordinate system, the smartphone attitude can be fixed within a conical surface in the earth coordinate system. As a result, we align the smartphone coordinate with the earth coordinate for removing 3 degrees of freedom to 1 degree by projection, as shown in Figure 3. Magnetometer is utilized to get the angle δ between Y_e and Y' axis in the earth coordinate system, where Y' is the projection of Y_s of the smartphone body coordinate system on the X_e-Y_e plane of the earth coordinate system. X_e (pointing to the earth east), Y_e (pointing to the earth north), and Z_e (parallel with the gravity) are the three reference axes in the earth coordinate system. Combining the result with the angle derived from the magnetometer reading and the rotation matrix, the component of sensing data corresponding to the vehicle moving direction can be calculated. The detailed formulation of the rotation matrix is explained in [21].

4.2. Detection Algorithm

4.2.1. Acceleration and Brake Detection. With the help of accelerometer, the acceleration and the brake can be distinguished (Figures 4 and 5). Moving average filter is used to remove noise from the raw acceleration. Due to the unpredictable road conditions and diverse driving preferences, the acceleration in real implementation includes noise. In order to filter out the noise, we use a state machine to detect the

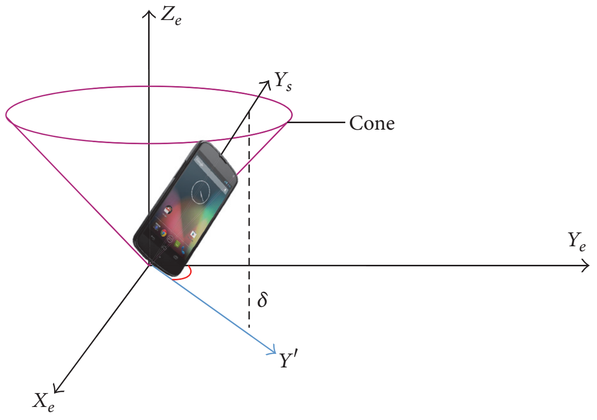


Figure 3: Align the smartphone coordinate system with the earth coordinate system.

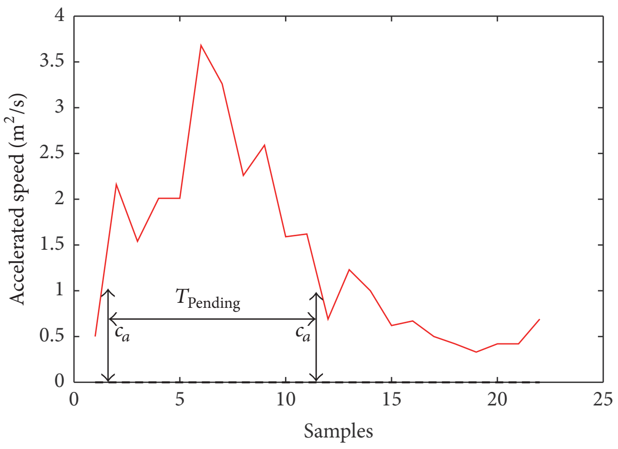


Figure 4: Accelerometer readings when the vehicle accelerates.

event. There are basically five states, Waiting-for-Acc, Acc-Pending, Brake-Pending, Acceleration, and Brake. The state transition procession is shown in Figure 6. The following list describes the specification of the parameters used in acceleration and brake detection algorithm:

Acc: acceleration from the moving average filter

Acc_Threshold: the threshold of the acceleration to enter the Acc-Pending state

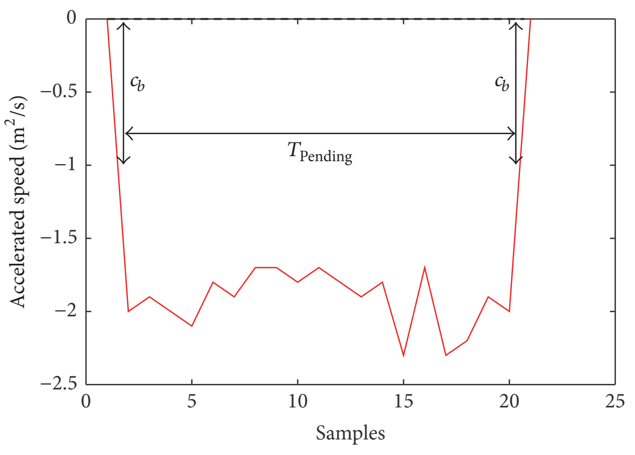


Figure 5: Accelerometer readings when the vehicle brakes.

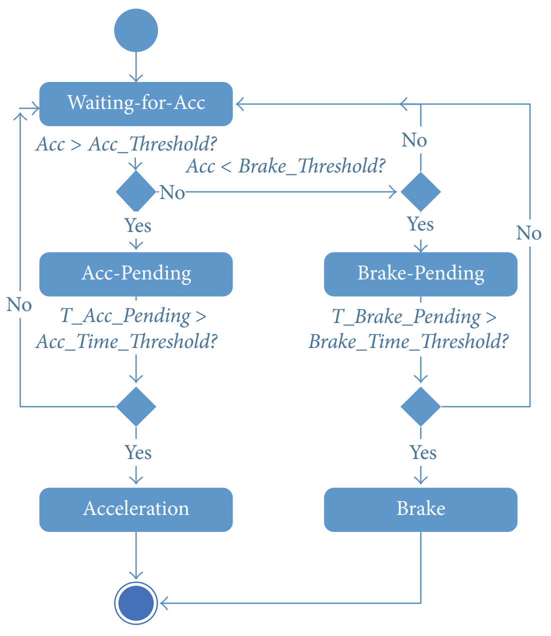


Figure 6: State diagram of the acceleration and brake detection.

Brake_Threshold: the threshold of the acceleration to enter the Brake-Pending state

Acc_Time_Threshold: the threshold of time in Acc-Pending state to enter the Acceleration state

Brake_Time_Threshold: the threshold of time in Brake-Pending state to enter the Brake state

T_Acc_Pending: the time in Acc-Pending state up to now

T_Brake_Pending: the time in Brake-Pending state up to now

Waiting-for-Acc is the initial state for the acceleration and brake detection. As the real acceleration appears as the undulatory curve due to noises, Acc-Pending and Brake-Pending are two transitional states to reduce the influence of the noise. For the acceleration, system enters into the Acc-Pending state after the *Acc* is greater than *Acc_Threshold* and exits the Acc-Pending state when the *Acc* is less than or equal to *Acc_Threshold*. If *T_Acc_Pending* T_{Pending} in Acc-Pending state is greater than *Acc_Time_Threshold*, the state becomes Acceleration, which means an acceleration is ongoing. For the brake, system enters into the Brake-Pending state after the *Acc* is less than *Brake_Threshold* and exits the Brake-Pending state when the *Acc* is greater than or equal to *Acc_Threshold*. If *T_Brake_Pending* in Brake-Pending state is greater than *Brake_Time_Threshold*, the state becomes Brake.

4.2.2. Turn and Lane Change Detection. The Z-axis readings of gyroscope on the smartphone are utilized to represent the vehicle angular speed. When the drivers do some behaviors to change the direction of vehicles (e.g., changing single or multiple lanes and making turns), the angular speeds have the obvious improving (Figure 7). For the left turn, a counterclockwise rotation around the Z-axis occurs and generates positive readings, whereas during a right turn, a clockwise rotation occurs and thus generates negative readings. For a left lane change, positive readings are followed by negative readings, whereas for a right lane change, negative readings are followed by positive readings.

By detecting bumps in the Z-axis gyroscope readings, we can determine whether the vehicle has made a left/right turn or has made single-lane change. The other steering maneuvers, that is, turn back and multiple-lane change, have a similar shape but with a different size in terms of width and height of the bumps. The parameter description of the turn and lane change detection is shown in the following list:

Ang_SE_Threshold: the threshold of angular speed to enter and leave a possible bump

Ang_Top_Threshold: the threshold of angular speed which the maximum value of the valid bump should be greater than

T_Bump: duration that angular speed is greater than *Ang_SE_Threshold* in a bump

T_Bump_Threshold: the minimum threshold of the *T_Bump* for a valid bump

G_threshold: the threshold of GPS speed to estimate if the vehicle is motionless

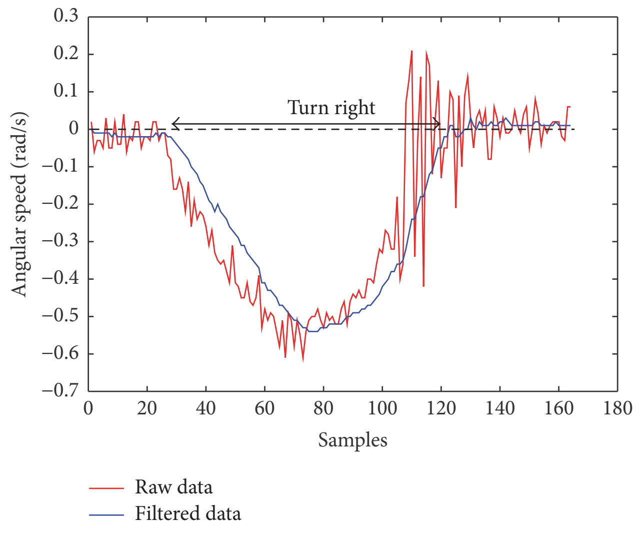
Delay_threshold: the threshold to judge the consecutive bump

T_stop: the duration whose GPS speed is less than the *G_threshold*

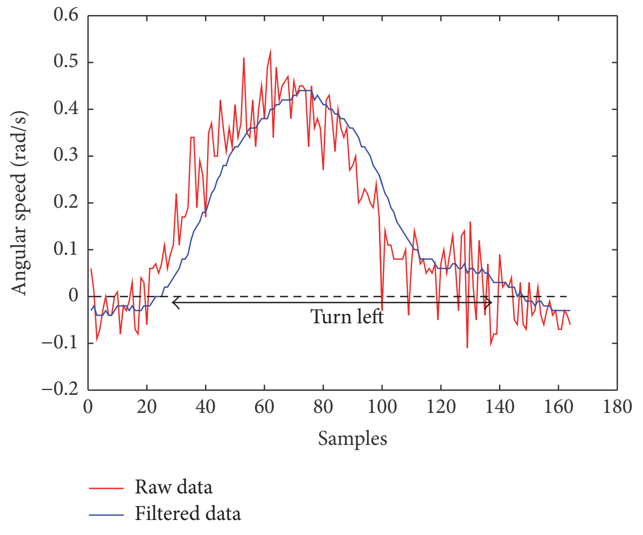
T_wait: duration of Waiting-for-Bump state

Moving average filter is adopted to remove noise from the raw gyroscope readings. The delay parameter of the filter is set to 15 samples which correspond to 0.75 seconds in the time domain. Experimental observation shows that it is short but good enough to extract the waveform of the bumps.

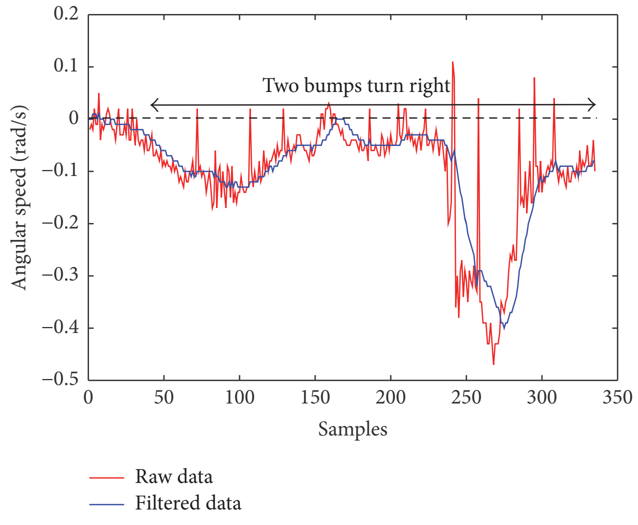
To reduce false positives and differentiate the bumps from jitter, a bump should satisfy the following three constraints for its validity: (1) all the readings during a bump should



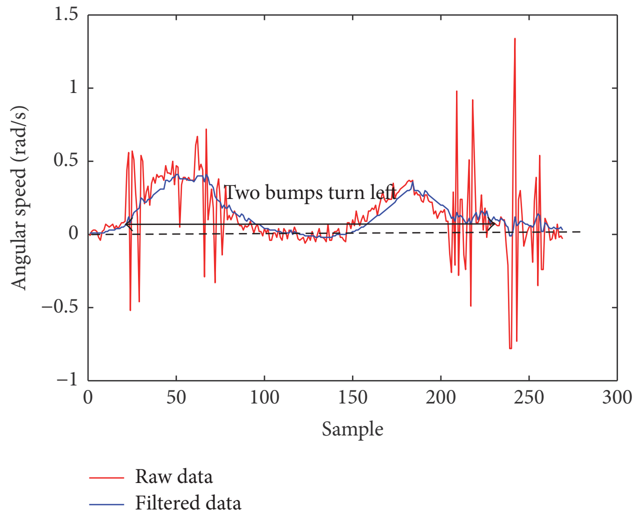
(a) Turn right (one bump)



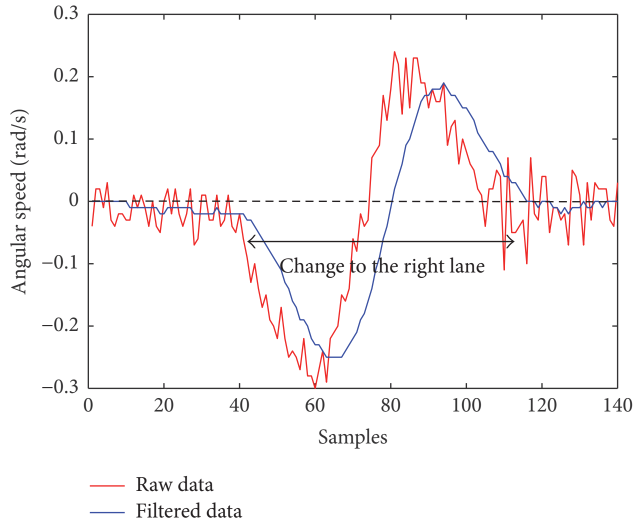
(b) Turn left (one bump)



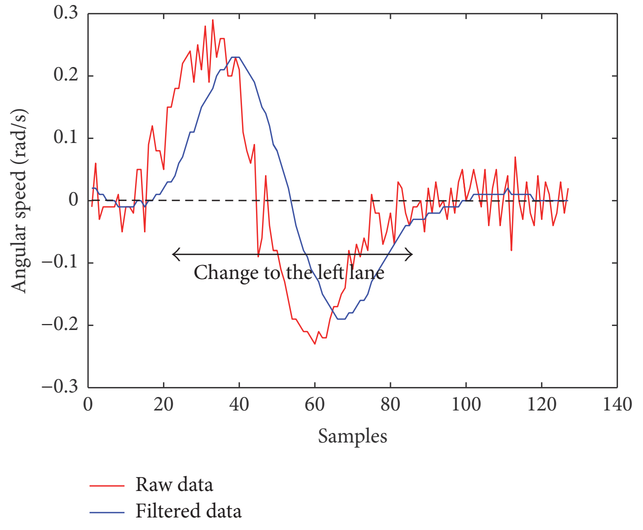
(c) Turn right (two bumps)



(d) Turn left (two bumps)

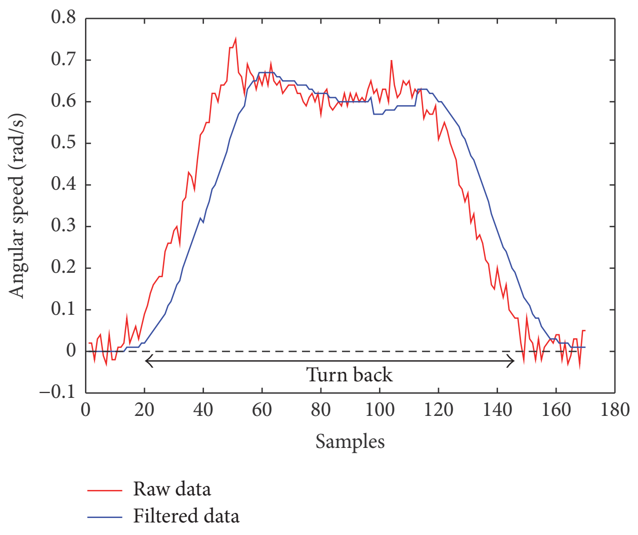


(e) Change to the right lane



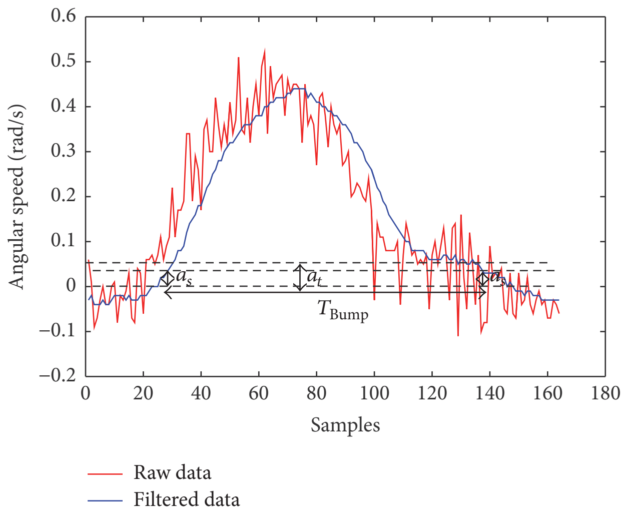
(f) Change to the left lane

Figure 7: Continued.

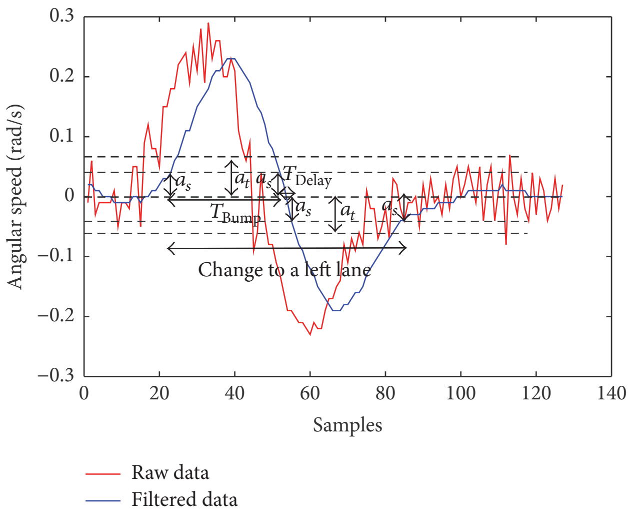


(g) Turn back

Figure 7: Gyroscope readings when the vehicle makes turns or lane changes.



(a) One bump: turn left



(b) Two bumps: lane change

Figure 8: Symbol description in gyroscope reading.

be larger than *Ang_SE_Threshold* a_s, (2) the largest value of a bump should be no less than *Ang_Top_Threshold* a_t, and (3) the duration of a bump *T_Bump* T_{Bump} should be no less than *T_Bump_Threshold* t_b (Figure 8(a)). For the two valid continuous bumps, the time between the two bumps should be less than *Delay_threshold* T_{Delay} except the time when the vehicle stops (Figure 8(b)).

There are seven states in the bump detection algorithm: No-Bump, One-Bump, Waiting-for-Bump, More-Bump, Bump-End, Turn, and Lane-Change. The state transition is influenced by angular speed and GPS speed. Angular speed is the Z-axis gyroscope reading. The state transition procession is shown in Figure 9.

In No-Bump state, angular speed is continuously monitored. When the absolute value of the measured angular speed reaches *Ang_SE_Threshold*, it is treated as the start of a possible bump, and the algorithm enters One-Bump state.

The One-Bump state terminates when the absolute value of the measured angular speed is below *Ang_SE_Threshold*. If the time of duration in One-Bump state is larger than *T_Bump* and the largest angular speed is larger than *Ang_Top_Threshold*, hence satisfying the three constraints, the first detected bump is assigned to be valid. In such a case, the algorithm enters Waiting-for-Bump state. Otherwise, it returns to No-Bump.

In Waiting-for-Bump state, it monitors the angular speed until its absolute value reaches *Ang_SE_Threshold*, and the duration is defined as *T_wait*. The GPS speed is also monitored to see if the turn is interrupted halfway. *T_stop* is used to define the duration whose GPS speed is less than

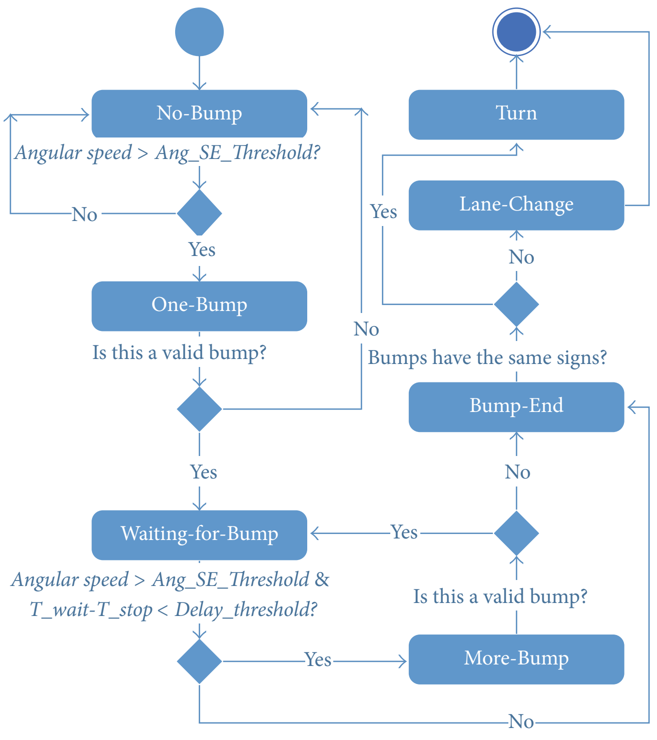


FIGURE 9: State diagram of the turn and lane change detection.

the *G_threshold*, which means the turn is interrupted. If the difference between the *T_wait* and *T_stop* is less than *Delay_threshold*, the system enters the More-Bump state, which means the oncoming bump is considered as consecutive bumps. If the new bump is valid, the system enters Waiting-for-Bump state for new bumps. If not, the system enters Bump-End state to determine the driving behavior.

In Bump-End state, the signs of these consecutive bumps can be used for driving behavior judgment. If these bumps have the same signs, the driving behavior is determined to be turn. If they have the opposite signs, the driving behavior is determined to be lane change. Besides, if there is only one valid bump (i.e., the second bump is invalid), the driving behavior is determined to be turn.

Based on bump detection, driving behaviors are classified into turns and lane changes. To get the detailed classification of the behaviors (e.g., left turn, right turn, and U-turn), the difference in the heading angle between the start and the end of a driving behavior is used.

We calculate the change in vehicle's heading angle from the readings of the gyroscope. Δt is the time interval of sampling. i is the bump index of the valid bumps. A_{n_i} is the angular speed of the $n_{i\text{th}}$ sampling and is used to approximately represent the average angular speed during the sampling period. Thus, we get the change in vehicle's heading angle from the integration of the heading angle change of each sampling period:

$$\theta = \sum_{i=1}^{I} \sum_{n_i=1}^{N_i} A_{n_i} \Delta t. \quad (1)$$

TABLE 1: The bounds of the angle change of heading for driving behavior.

	Lower bound	Upper bound
Turn left	70°	110°
Turn right	−70°	−110°
Turn back	−200°/160°	−160°/200°
Left lane change	−20°	20°
Right lane change	−20°	20°

For the turning left, θ is around 90°. For turning right, it is around +90°. For turning back, θ is around ±180°. For the lane change, it is around 0°.

As the drivers cannot make a perfect driving behavior to get the perfect coincident degree, the ranges of the driving behaviors are extended as shown in Table 1.

Based on the horizontal displacement, we furthermore extract the multiple-lane change from the lane change as the horizontal displacement of multiple-lane change is greater than the single one. If the horizontal displacement is less than *HD_Lower_Threshold* (e.g., the width of the lane), it means that these consecutive bumps are caused by some interference instead of lane change. If the horizontal displacement is greater than *HD_Upper_Threshold*, the behavior is treated as the multiple-lane change. Otherwise, the behavior is treated as single-lane change. The horizontal displacement H can be calculated as follows. Thereinto, Δt is the time interval of sampling, A_i is the angular speed of the i_{th} sampling, and G_n is the GPS speed of the n_{th} sampling:

$$\theta_n = \sum_{i=1}^{n} A_i \Delta t,$$
$$H = \sum_{n=1}^{N} G_n \Delta t \sin(\theta_n). \quad (2)$$

5. Beacon-Based Communication

We use the Wi-Fi of smartphones for communication. As the normal Wi-Fi has the association and authentication procession which can cause the latency, we choose the beacon frame of the Wi-Fi AP to carry the sensing information. The beacon frame is used to declare the existing AP and can be transferred by the AP without association and authentication procession. The Beacon Stuffing embeds the intended messages within the SSID field of the Wi-Fi beacon header and is available in the Wi-Fi AP mode [22].

The problem of using beacon to transfer sensing information is that the beacon can only be transferred from the AP to the client, which causes the one-way transmission. However, it is not enough that only a part of the devices transfers the sensing information, as every device needs to broadcast its mobility information to the others. To solve this problem, we propose the event-driven communication algorithm; once the vehicle detects an event (e.g., acceleration, brake, turn, and lane change), the smartphone inside will switch to the AP mode to broadcast the beacons for a while; after that the smartphone will switch to the mode to scan the beacons.

To provide the reference for collision forewarning module, these elements are selected: the latitude and longitude from the GPS reader, the speed from the GPS reader (m/s), the travel direction from the magnetometer sensor (degree 0~360), the time from the GPS reader, and the driving behaviors. These elements are combined together to replace SSID field of beacon. A special string "Sen" is used in front of these element for differentiating SenSafe's SSID from the others.

To satisfy the requirements of accuracy, the 0.1 meters approximately correspond to the 0.000001 degrees in the latitude and longitude, which means the latitude and the longitude need 19 characters in the SSID field (e.g., 116.364815 and 39.967001 in BUPT). Besides, the time from the GPS reader also needs too many characters. If we want to make the precision of the time to be millisecond (e.g., 20:20:20 234), it will take 9 characters even without considering separators. However, the length of beacon messages' SSID field is 32 characters, which is not enough for all elements. As a result, we compress these elements in the AP side and decompress these elements in the client side.

For the latitude, one degree is about 111 kilometers. As the maximum transmission range of the Wi-Fi beacon is less than 1 kilometer, most of the significant digits of latitude and longitude are same for the sender and receiver. There is no need to transfer all the significant digits of latitude and longitude. Thus, the significant digits of latitude and longitude are chosen to cover the range around the 1000 meters (i.e., last 5 significant digits). It is similar for the time, as hour and minute are always same for the senders and receivers. Therefore, we use 4 significant digits to represent the time from 0.01 seconds to 10 seconds. An example is shown in Figure 10. Further, due to the boundary situation of the time, when the local device decompresses the time from the remote devices, it will compare the difference between the decompressed time and the local time. If the absolute value of difference is greater than the threshold (30 seconds), the minute of the decompressed data will be modified to be the adjacent number (+1/−1) to get the minimum reasonable difference. An example is shown in Figure 11. The remote time is 20:20:59 121, and the local time is 20:21:00 136, which will make the normal decompressed time 20:21:59 121. As it is unreasonable, the adjacent minute (20:20) is chosen to be the prefix of the remote time to get the minimum the absolute value of time difference. For the latitude and longitude, the purpose of the decompression is obtaining the minimum difference between the remote location and the local location. Similar to the time, if the absolute value of difference is greater than the threshold (200 meters), the first decimal places of the latitude and longitude are modified to be the adjacent number (+1/−1) to get the minimum reasonable difference.

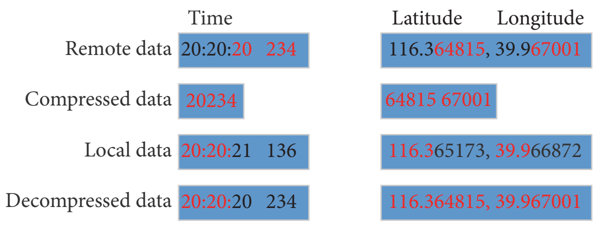


FIGURE 10: The example of compression and decompression.

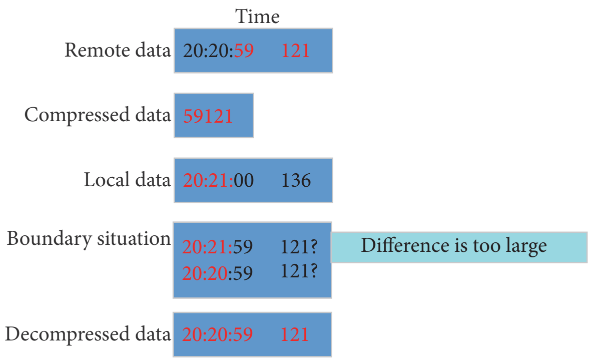


FIGURE 11: The example of time boundary situation.

The format of the SSID field is demonstrated in Figure 12. We use the number to represent the special driving behaviors in the driving behavior segment (0: pedestrian, 1: acceleration, 2: brake, 3: turn left, 4: turn right, 5: turn back, 6: left lane change, and 7: right lane change). For the pedestrians, their smartphones sense their moving information from the GPS readers and broadcast the same elements to remind the drivers of their existence. The driving behaviors element is set to "0", which means this message is from the pedestrian.

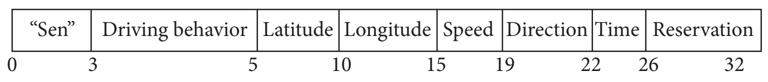


Figure 12: The format of the SSID field.

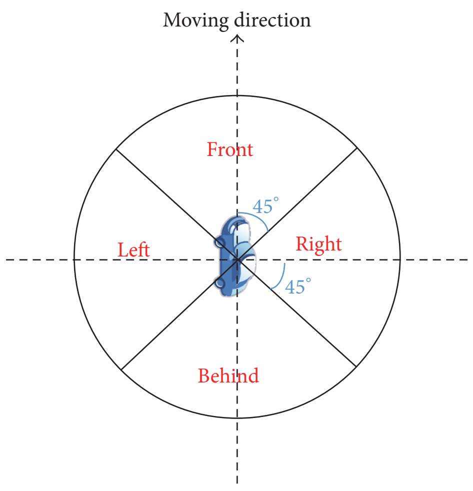


Figure 13: The area division for safety reminding.

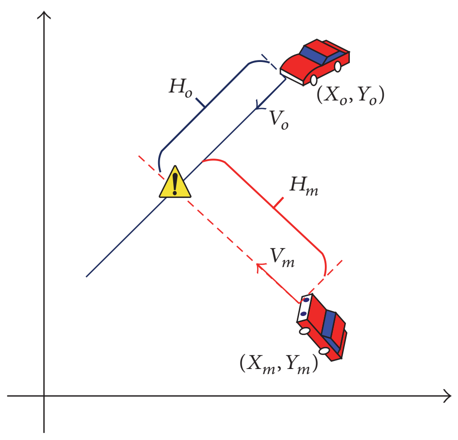


Figure 14: Intersection collision estimation.

6. Collision Forewarning

The vehicle will receive two kinds of messages: one kind from the pedestrians and the other from the vehicles.

After the vehicle gets the driving information of the surrounding vehicles, it will estimate if these vehicles will crash into it to ensure the safety of the vehicle and the driver. Besides, it will also remind the drivers of the driving behaviors from the surrounding vehicles to make the drivers have a better understanding of the driving environment.

As is shown in Figure 13, considering the vehicle moving direction, the surrounding area of the vehicle is divided into four quarters: front, behind, left, and right. After receiving a new message, the vehicle will calculate which area it belongs to. Furthermore, the driver can get the information where the threat comes from.

When the vehicle receives the messages from the surrounding vehicles, it translates latitudes and longitudes to Gauss-Krueger plane rectangular coordinates system. Then it calculates the driving vector of each vehicle and finds vehicles that have intersection with it and gets the location where they may have the intersection, as shown in Figure 14. If the vectors have the intersection in the near future, it will calculate the distance of the current vehicle and the threat vehicle to intersection and calculate the time to the intersection (safety reaction time) furthermore. If one of these two times is less than the *safety_reaction_time_threshold*, and the absolute difference of them (safety intersection time) is less than *safety_intersection_time_threshold*, these two vehicles will be estimated to have the threat of crashing into each other. When the vehicle receives the messages from the surrounding pedestrian, it will calculate the distance of the current vehicle and pedestrian to intersection and the time of vehicle to the intersection. If the distance of pedestrian is less than *safety_distance_threshold*, and the time of vehicle to the intersection is less than *safety_reaction_time_threshold*, the vehicle will be estimated to have the threat to the pedestrian.

We reference the safety reaction time threshold (3 seconds) recommended in [23] and safety intersection time threshold (2 seconds) recommended in [24] to avoid a collision when GPS is accurate. Assuming the inaccuracies of GPS location are up to 10 meters and the speeds of the vehicles are around 10 m/s, as a result, the error of the safety reaction time is up to 1 second and the error of the safety distance is up to 10 meters. Thus, we set safety reaction time threshold (4 seconds) to 1 second higher than the value which assumes the GPS is accurate and safety intersection time threshold (4 seconds) to 2 seconds higher than the value which assumes the GPS is accurate.

7. Performance Evaluation

To evaluate the performance of SenSafe, we implemented SenSafe on a Samsung Galaxy Note II and a Huawei Changwan 4X.

7.1. Parameter Setting. The parameters used in SenSafe are set as the values in the following list:

Acc_Threshold: 0.8 m/s^2

Brake_Threshold: −1 m/s^2

Acc_Time_Threshold: 0.6 seconds

Brake_Time_Threshold: 0.6 seconds

Ang_SE_Threshold: 0.03 rad/s

Ang_Top_Threshold: 0.05 rad/s

T_Bump_Threshold: 1.5 seconds

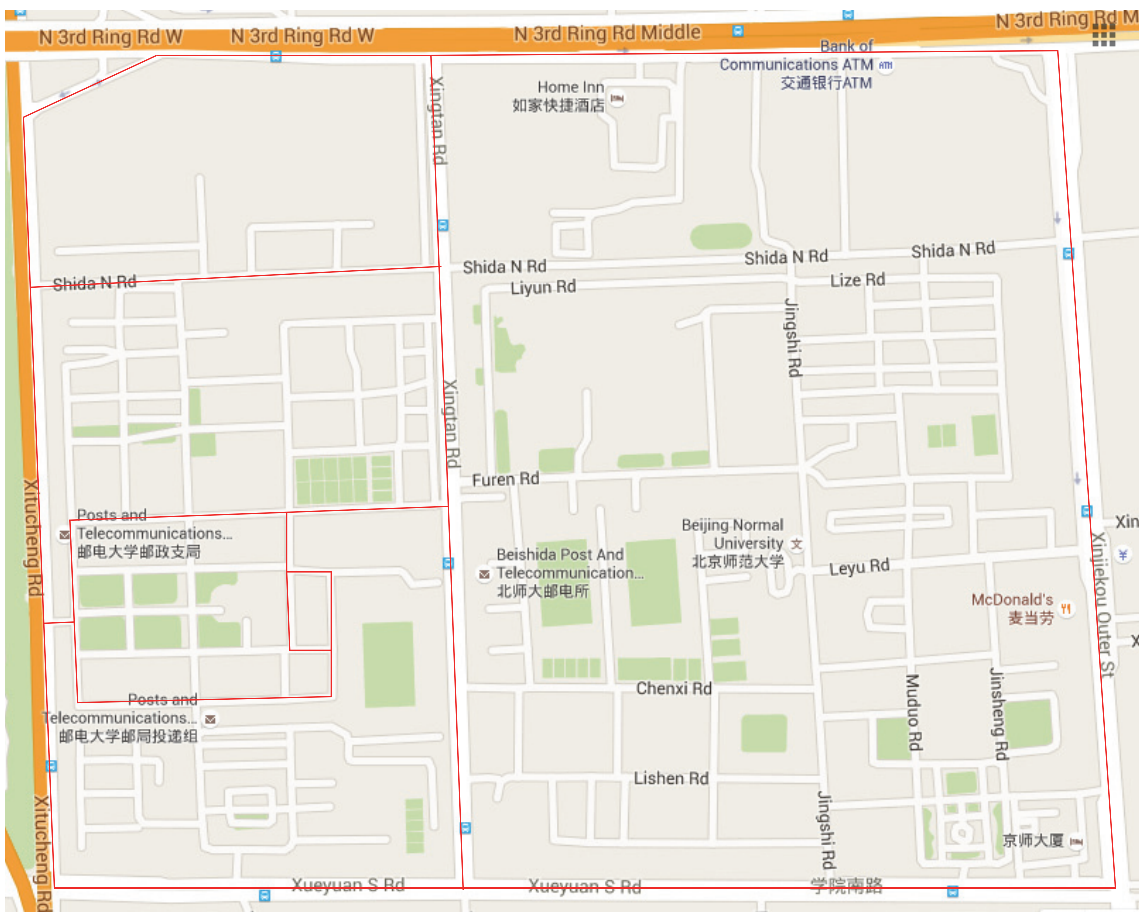


FIGURE 15: Real road testing routes.

Delay_threshold: 2 seconds

G_threshold: 0.3 m/s

safety_reaction_time_threshold: 4 seconds

safety_intersection_time_threshold: 4 seconds

safety_distance_threshold: 12 meters

HD_Lower_Threshold: 1.5 meters

HD_Upper_Threshold: 4 meters

7.2. Accuracy of the Driving Behavior Detection. First, we evaluated the accuracy of SenSafe in determining driving behaviors around the Beijing University of Posts and Telecommunications. The roads we used for testing are shown in Figure 15. The car we used for the test was Dongfeng Peugeot 307. During these experiments, the smartphones were mounted on the windshield.

We tested the driving behaviors including turn left, turn right, acceleration, brake, single-lane change, multiple-lane change, and turn back. The difference between the single-lane change and multiple-lane change is that multiple-lane change needs more horizontal displacement. The test results are shown in Figure 16. The real number of times for each driving behavior is manually recorded. From the figure we can see that almost all of the turn left, turn right, acceleration, and brake have been detected. 88% singe-lane change, 91% multiple-lane change, and 84% turn back can be successfully detected.

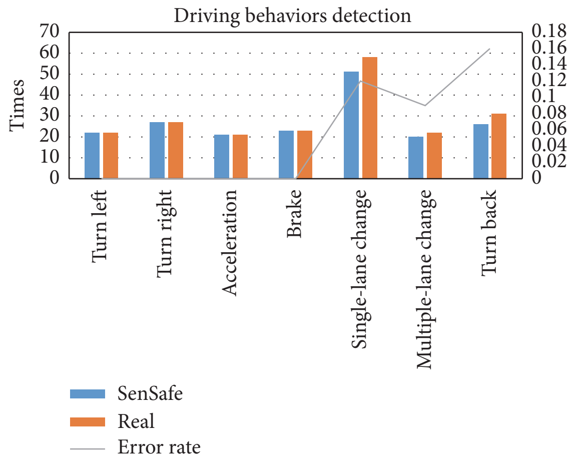


FIGURE 16: Driving behaviors detection results.

We furthermore summarized the horizontal displacement and angle change of heading for single-lane change, multiple-lane change, and turning back in Tables 2, 3, and 4.

Table 2: Horizontal displacement and angle change of heading for single-lane change.

Displacement	1	2	3	4	5	Average
SenSafe (m)	1.91	1.61	3.06	2.1	2.45	2.226
Real (m)	1.92	1.59	3	1.9	2.27	2.136
Heading angle change	0.6	0.86	1.63	0.66	4.24	1.598

Table 3: Horizontal displacement and angle change of heading for multiple-lane change.

Displacement	1	2	3	4	5	Average
SenSafe (m)	4.87	4.83	5.53	9.3	8.67	6.656
Real (m)	4.81	4.84	5.76	9.1	9.77	6.854
Heading angle change	1.98	0.66	1.55	7.76	0.52	2.494

Table 4: Horizontal displacement and angle change of heading for turning back.

Displacement	1	2	3	4	5	Average
SenSafe (m)	7.88	7.47	8.09	7.01	8.91	7.87
Real (m)	9.07	7.69	9.51	10.21	7.79	8.854
Heading angle change	179.45	172.4	179.48	183.98	183.86	179.834

The real horizontal displacement is calculated using the speed from the OBD of the vehicles. From Tables 2, 3, and 4 we can see that SenSafe can provide the accurate horizontal displacement and angle change of heading.

7.3. Performance of Communication. We test the performance of the communication considering the relationship between the distance and the probability of successfully receiving at least one packet per second.

We used one smartphone to broadcast the beacons and used the other to scan the beacons. The results are shown in Table 5. From the table we can see that, when the distance is less than 30 meters, the client can almost receive at least one packet per second. After the distance is beyond 50 m, the probability has an obvious drop.

7.4. Performance of Collision Forewarning. We tested the performance of collision forewarning considering the alert for the brake in front, acceleration in behind, and intersection collision forewarning.

For the brake in front, we used the front vehicle to make the brake to see if the behind vehicle can get the alert. The distance between the two vehicles was around 30 meters. We tested this scenario ten times, and the behind vehicles had gotten the alert ten times.

For acceleration in behind, we used the behind vehicle to make the acceleration to see if the front vehicle can get the alert. The distance between the two vehicles was around 30 meters. We tested this scenario ten times, and the front vehicles had gotten the alert ten times.

Table 5: Relationship between the distance and the probability of successfully receiving at least one packet per second.

Distance (m)	Total time (second)	Seconds no packet	Probability
10	1200	0	100%
20	1200	39	97%
30	1200	72	94%
40	1200	234	81%
50	1200	354	71%
60	1200	557	54%

For intersection collision forewarning, we tested it in the campus of Beijing University of Posts and Telecommunications. Considering that the probability of successfully receiving at least one packet per second is less than 80% when the distance is greater than the 40 meters, we add 1 more second in safety reaction time threshold and safety intersection time threshold when the distances between vehicles are greater than the 40 meters. We used two vehicles to meet at the intersection to see if the driver will be reminded of the coming vehicles. The speed of the vehicle was around 6 m/s. One of the vehicles accelerated for 1 second with acceleration around 1 m/s^2 to trigger the beacon broadcasting. We tested this scenario ten times, and the listening vehicle had gotten the alert nine times. We used one vehicle and a pedestrian to meet at the intersection to see if the driver will be alerted of the coming pedestrian. The speed of the pedestrian was around 2 m/s, and the speed of the vehicle was around 6 m/s. We tested this scenario ten times, and the vehicle had gotten the alert nine times.

8. Conclusion

In this paper, we develop a smartphone-based traffic safety framework named SenSafe to sense the surrounding events and provide alerts to drivers. Firstly, a driving behaviors detection mechanism is provided which can run on commodity smartphones. Secondly, the Wi-Fi association and authentication overhead is reduced to broadcast the compressed sensing data using the Wi-Fi beacon to inform the drivers of the surroundings. Thirdly, a collision estimation algorithm is proposed to provide the appropriate warnings. Finally, an Android-based implementation of SenSafe has been achieved to evaluate the performance of SenSafe in real environments, and experimental results show that SenSafe can work well in real environments.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the National Key R&D Program of China (2016YFB0100902).

References

[1] Real time traffic accidents statistics, http://www.icebike.org/real-time-traffic-accident-statistics/.

[2] Intelligent Transportation Systems-Dedicated Short Range Communications, http://www.its.dot.gov/DSRC/.

[3] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[4] Applications, http://www.mobileye.com/technology/applications/.

[5] Drivea-driving assistant app, https://play.google.com/store/apps/details?id=com.driveassist.experimental&hl=en.

[6] Turn Your Smartphone Into a Personal Driving Assistant, http://www.ionroad.com/.

[7] Augmented Driving, https://itunes.apple.com/us/app/augmented-driving/id366841514?mt=8.

[8] C.-W. You, N. D. Lane, F. Chen et al., "CarSafe app: alerting drowsy and distracted drivers using dual cameras on smartphones," in *Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*, pp. 13–26, ACM, Taipei, Taiwan, June 2013.

[9] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher, "Smartroad: smartphone-based crowd sensing for traffic regulator detection and identification," *ACM Transactions on Sensor Networks*, vol. 11, no. 4, article 55, 2015.

[10] Z. Wu, J. Li, J. Yu, Y. Zhu, G. Xue, and M. Li, "L3: sensing driving conditions for vehicle lane-level localization on highways," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM '16)*, pp. 1–9, IEEE, San Francisco, Calif, USA, April 2016.

[11] D. Shin, D. Aliaga, B. Tunçer et al., "Urban sensing: using smartphones for transportation mode classification," *Computers, Environment and Urban Systems*, vol. 53, pp. 76–86, 2015.

[12] J. Yu, H. Zhu, H. Han et al., "SenSpeed: sensing driving conditions to estimate vehicle speed in urban environments," *IEEE Transactions on Mobile Computing*, vol. 15, no. 1, pp. 202–216, 2016.

[13] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, "Sensing vehicle dynamics for determining driver phone use," in *Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*, pp. 41–54, ACM, Taipei, Taiwan, June 2013.

[14] C. Saiprasert, T. Pholprasit, and S. Thajchayapong, "Detection of driving events using sensory data on smartphone," *International Journal of Intelligent Transportation Systems Research*, 2015.

[15] D. Chen, K.-T. Cho, S. Han, Z. Jin, and K. G. Shin, "Invisible sensing of vehicle steering with smartphones," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '15)*, pp. 1–13, Florence, Italy, May 2015.

[16] CohdaWireless, http://www.cohdawireless.com/.

[17] L. Zhenyu, P. Lin, Z. Konglin, and Z. Lin, "Design and evaluation of V2X communication system for vehicle and pedestrian safety," *The Journal of China Universities of Posts and Telecommunications*, vol. 22, no. 6, pp. 18–26, 2015.

[18] U. Hernandez-Jayo, I. De-La-Iglesia, and J. Perez, "V-alert: description and validation of a vulnerable road user alert system in the framework of a smart city," *Sensors*, vol. 15, no. 8, pp. 18480–18505, 2015.

[19] W. Sun, C. Yang, S. Jin, and S. Choi, "Listen channel randomization for faster Wi-Fi direct device discovery," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM '16)*, IEEE, San Francisco, Calif, USA, April 2016.

[20] K. Dhondge, S. Song, B.-Y. Choi, and H. Park, "WiFiHonk: smartphone-based beacon stuffed WiFi Car2X-communication system for vulnerable road user safety," in *Proceedings of the 79th IEEE Vehicular Technology Conference (VTC '14)*, pp. 1–5, IEEE, Seoul, South Korea, May 2014.

[21] P. Zhou, M. Li, and G. Shen, "Use it free: instantly knowing your phone attitude," in *Proceedings of the 20th ACM Annual International Conference on Mobile Computing and Networking (MobiCom '14)*, pp. 605–616, ACM, Maui, Hawaii, USA, September 2014.

[22] R. Chandra, J. Padhye, L. Ravindranath, and A. Wolman, "Beacon-stuffing: Wi-Fi without associations," in *Proceedings of the 8th IEEE Workshop on Mobile Computing Systems and Applications (HotMobile '07)*, pp. 53–57, Tucson, AZ, USA, February 2007.

[23] M. M. Minderhoud and P. H. L. Bovy, "Extended time-to-collision measures for road traffic safety assessment," *Accident Analysis & Prevention*, vol. 33, no. 1, pp. 89–97, 2001.

[24] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident Analysis & Prevention*, vol. 35, no. 3, pp. 427–433, 2003.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 2821680, 14 pages
http://dx.doi.org/10.1155/2016/2821680

Research Article

A Service-Oriented Approach to Crowdsensing for Accessible Smart Mobility Scenarios

Silvia Mirri,[1] Catia Prandi,[1] Paola Salomoni,[1] Franco Callegati,[2] Andrea Melis,[2] and Marco Prandini[1]

[1] *Department of Computer Science and Engineering, University of Bologna, Bologna, Italy*
[2] *Department of Electrical and Information Engineering, University of Bologna, Bologna, Italy*

Correspondence should be addressed to Andrea Melis; a.melis@unibo.it

Received 20 May 2016; Accepted 1 September 2016

Academic Editor: Enrico Natalizio

This work presents an architecture to help designing and deploying smart mobility applications. The proposed solution builds on the experience already matured by the authors in different fields: crowdsourcing and sensing done by users to gather data related to urban barriers and facilities, computation of personalized paths for users with special needs, and integration of open data provided by bus companies to identify the actual accessibility features and estimate the real arrival time of vehicles at stops. In terms of functionality, the first "monolithic" prototype fulfilled the goal of composing the aforementioned pieces of information to support citizens with reduced mobility (users with disabilities and/or elderly people) in their urban movements. In this paper, we describe a service-oriented architecture that exploits the microservices orchestration paradigm to enable the creation of new services and to make the management of the various data sources easier and more effective. The proposed platform exposes standardized interfaces to access data, implements common services to manage metadata associated with them, such as trustworthiness and provenance, and provides an orchestration language to create complex services, naturally mapping their internal workflow to code. The manuscript demonstrates the effectiveness of the approach by means of some case studies.

1. Introduction

As world populations concentrate in cities, mobility in urban environments is becoming one of the most prominent and interesting research fields in the smart city context. A well-known definition of smart city is provided in [1] and says that a smart city is "a city well performing in a forward-looking way in economy, people, governance, mobility, environment, and living, built on the smart combination of endowments and activities of self-decisive, independent and aware citizens." The World Health Organization has recently released a report about Urban health [2], which claims that about 3.7 billion people live in cities today and that a further 1 billion people will be added by 2030, with 90% of the growth being in low- and middle-income countries. According to this study, the ways that cities are planned and built can profoundly affect the ability of their citizens to live long, healthy, and productive lives. Urban mobility plays a key-role in this context, because it is strategic in making cities age-friendly and accessible for communities, with particular regard to those persons with disabilities [2]. Hence, providing and adequately orchestrating services devoted to improving urban mobility is fundamental in achieving *smart mobility* [3].

In this context, the crowdsensing and the mobility as service paradigms are emerging. In particular, crowdsensing is rising thanks to the widespread diffusion of mobile devices: it involves people who are moving and collecting data from different places and routes, by carrying sensors integrated in their mobile devices, such as smartphones and tablets [4]. Here, we can identify the three interrelated components (space, people, and technology) of the urban computing systems, as presented in [5].

The concept of Mobility as a Service (MaaS) was born in Finland and it is rapidly spreading worldwide [6] as an effective approach to achieve business efficiency, traveler satisfaction, and government agenda fulfillment through smart mobility.

Given this background, we envision the creation of ICT infrastructures based on microservices. This modern and renowned development model [7] fosters the creation of an ecosystem of reusable components. In the context of MaaS, microservices shall efficiently and flexibly combine heterogeneous data sources, such as available transport options, real-time data regarding vehicles and infrastructures, and pricing, to provide customized travel planning, information, and ticketing to final users, as well as monitoring and strategic planning tools to policy-makers. In this context, crowdsensing plays a fundamental role, letting the users and their devices be a significant actor in the whole picture, becoming one of the data sources. As emerging by the results found in [8], crowdsensing from citizens' devices is an important advantage and opens a range of potential applications and tools. This would improve data and applications made available by operators, policy-makers, and transport providers, enriching the entire smart mobility context.

This paper presents the design of an infrastructure as a marketplace for mobility services, called *Smart Mobility for All* (SMAll). A prototype of such infrastructure has been developed and its architecture is described in the remainder of this paper, as well as some of the provided services. In our vision, SMAll is the enabling technology to solve the challenges of the MaaS market, from developing user-contributed, crowdsourced applications and crowdsensing services to launching a MaaS operator and to planning effective and sustainable transport policies for smart cities. Particular attention has been given to specific services offered with the aim of supporting mobility of citizens with disabilities and special needs within urban environments.

The remainder of this paper is organized as follows. Section 2 presents the background and some of the main related work which have inspired and driven our research. Section 3 describes the overall system architecture we have designed and developed, which is based on services orchestration. Two broad categories of services, namely, data quality management and data sources, are illustrated in detail in Sections 4 and 5, respectively, before their orchestration is described in Section 6. Two case studies are introduced in Section 7, and, finally, Section 8 concludes the paper and presents some future works.

2. Background and Related Work

An emerging trend introduced with cloud computing [9] defines a new category of models which can be identified under the umbrella term Everything as a Service (XaaS) [10]. The basic idea behind cloud computing is to concentrate resources, such as hardware and software, into few physical locations and offer those resources as services to a large number of users who are located in many different geographical locations around the world in an effective way. In this context, three major service models have been traditionally exploited: infrastructure-as-a-service, platform-as-a-service, and software-as-a-service. The main common element among them is that they all provide resources as a service. These models arose a wide popularity and starting from them, several similar yet context-specific models have been proposed [11].

One of these ones is the Sensing as a Service (SaaS) model, which can be considered a solution based on IoT infrastructure and it has the capability to address some of the most challenging issues in smart cities [12]. Many everyday objects are equipped with sensors and the European Commission has predicted that, by 2020, there will be 50 to 100 billion devices connected to the Internet [13]. This represents a strong motivation behind the diffusion and the opportunity of efficiently exploiting the SaaS model. In a typical SaaS cloud, multiple sensing servers can be deployed to handle sensing requests from different locations [14]. Usually, a SaaS cloud works as follows. When a cloud user initiates sensing requests through a Web front-end from either mobile phone or a computer, the request will be forwarded to a sensing server which will then push the request to a subset of mobile phones that happen to be in the area of interest [15]. Such mobile devices will fulfill the corresponding sensing task. The sensed data will then be collected by a sensing server, stored in a database and returned to the cloud user who requests the service. An interesting feature is that in such a system a mobile user can be at the same time a provider and a consumer of the sensing services [15, 16]. And this is the case of the sensing service prototype we are going to present in Section 4.2 of this paper.

Taking into account mobile phone sensing, we can identify two primary paradigms [17]:

(1) Participatory sensing: mobile users actively engage in sensing activities by manually determining how, when, what, and where to sense.

(2) Opportunistic sensing: sensing activities are fully automated without the involvement of mobile users.

It is worth mentioning that, despite the fact that in traditional sensor network the owner is typically a single organization, in mobile phone and sensors the control is spread between different individual users. This means that mobile sensing activities and resulting data are not controllable and not easy to predict [14]. In some contexts, in particular in the participatory sensing ones, SaaS is considered as a crowdsourcing system that depends on mobile users to provide data [15], and it can also be referred to as crowdsensing [18], which has been also defined as a subtype of crowdsourcing, where the outsourced job is a sensing task [4].

Crowdsensing is recognized as an important technological enabler for smart cities that has attracted several research efforts, with the aim of improving sensing quality on mobile devices, promoting user participation, and validating collected data [19, 20]. Compared to infrastructure-based sensing, crowdsensing has several advantages, even if it can bring some additional issues.

A system based on crowdsensing can potentially be cheaper than infrastructure-based sensing solutions, because it does not require the deployment of expensive fixed infrastructure. Moreover, it is easier to deploy and can be used in areas where deploying a fixed infrastructure can be difficult or maybe impossible, but it can introduce additional complexity and challenges. In general, mobile devices used for crowdsensing and infrastructure-based sensing are complementary technology that can cooperate to enable sensing in smart cities [18].

In the smart city context, crowdsensing can be exploited by involving sensors which are moving (since they are carried by users) and human intelligence into the sensing process [4]. Some of these use cases address tasks related to urban transportation systems, such as tracking of public vehicles (e.g., buses, trams, and subways) or others like mapping bumps on the road to inform authorities.

Crowdsensing can provide great support in optimizing urban transportation. Traffic can be unpredictable; moreover, most influenced public transportation lines can bear shorter or longer delays. Weather conditions can influence the traveling speed of vehicles in the city. In [21], an effective way of describing the entities of a crowdsourced public transit networks (including locations and vehicles) was presented and discussed. A system devoted to monitoring public transport vehicles with an application running on traveling users' mobile phones and detecting the stopping places of vehicles is described in [22]. An interesting example of participatory sensing is represented by the platform called Waze [23], which supports car drivers to get information on road conditions. Thanks to its versatility, it was the wider community-assisted navigation app in 2015. To improve routing, users can report changes in local maps to keep them up to date [4].

It is important to note that systems based on crowdsensing need to reach a critical mass of gathered data in effectively and efficiently providing services. For this reason, contributors have to feel motivated and involved in collecting data. Different research works have proved that resorting in intrinsic motivation (the activity is perceived as intrinsically rewarding) and/or extrinsic motivation (the action is driven by an external outcome, as rewards or an increase of reputation) makes it possible to engage contributors in participating, with an increment of the quantity and quality of collected data [24, 25]. An interesting concept that some of the authors are investigating is the one about the use of gamification so as to motivate the participation of the crowd in gathering data about the urban environment (see, e.g., [26–28]).

Some among the several crowdsourcing and crowdsensing systems and applications developed in the smart cities paradigm are devoted to let citizens collaborate in improving the quality of life in their urban environment [29, 30]. A part of them aims to collect data about urban accessibility [31], improving the quality of life and the level of independence of persons with disabilities [32, 33]. Many sensing apps have been developed to monitor human activities and a part of them could be effectively used to detect accessibility/pedestrian barriers (such as stairs) and facilities (such as zebra crossing). These researches present sensing architectures and algorithms studied to be used in different contexts, so they need to be adapted in order to be exploited in detecting barriers and facilities (see, e.g., [34, 35]). In [36], the authors (by using data obtained by a smartphone accelerometer) aim to recognize the position where a pedestrian stops and crosses a street ruled by a traffic light. Some barriers and facilities could be recognized more easily by using cooperative sensing, working on detecting movement of groups of people [37]. In [38, 39], the authors propose methodologies for developing large scale accessibility map with personal sensing by using smart phones. In particular, the idea is to exploit devices held by wheelchair citizens and then to apply machine learning technologies (i.e., supervised learning techniques) with the aim of estimating types of ground surfaces.

Using moving sensors in crowdsourcing is called *mobile crowdsensing* (MCS) [4]. MCS differs from the deployed sensor networks in involving people who are moving and collecting data from different places and paths. People can carry sensors integrated to their mobile devices and they can provide information about the surroundings manually [19]. The MCS as a Service (MCSaaS) paradigm has been proposed in [40]. The authors discussed about the MCSaaS vision and presented a platform prototype and its evaluation. In particular, regarding the MCSaaS vision, the authors proposed to implement such an approach by splitting the MCS application deployment into two domains: the infrastructure and the application ones.

Another important and interesting concept that is at the basis of our work is Mobility as a Service (MaaS) [6]. One of the main advantages of a MaaS provider is that it shall offer a unique and seamless interface to users, aggregating heterogeneous transport options offered by different mobility providers (e.g., different agencies providing transportation by taxi, bus, train, airplane, and car-sharing, including the public transportation providers) and handling the whole experience of traveling, from providing information to travel planning and payments [41].

All these concepts and studies have inspired our work and the resulting system we present in this paper. In particular, our prototype is exploiting sensing, mobile crowdsourcing, and mobility as a service with the specific purpose of supporting citizens in wandering the city (i.e., in the context of smart mobility). A specific attention has been paid to meet the needs of those people who would get more benefits than the others from the availability of information about urban accessibility in terms of barriers and facilities.

3. Smart Mobility for All (SMAll)

From a software engineering point of view, it is useful to frame the various functions needed to build any smart-city vertical application within a common reference model based on microservices [7].

By modeling and implementing every component of a mobility application as a service, several remarkable advantages emerge. Data can be transparently collected from different sources that, wrapped inside a microservice, become available through a standard interface. Preprocessing and labeling of data, for example, to assign trustworthiness values, can be implemented by means of different algorithms available as services; these, in turn, can take advantage of shared knowledge bases, for example, managing user ratings.

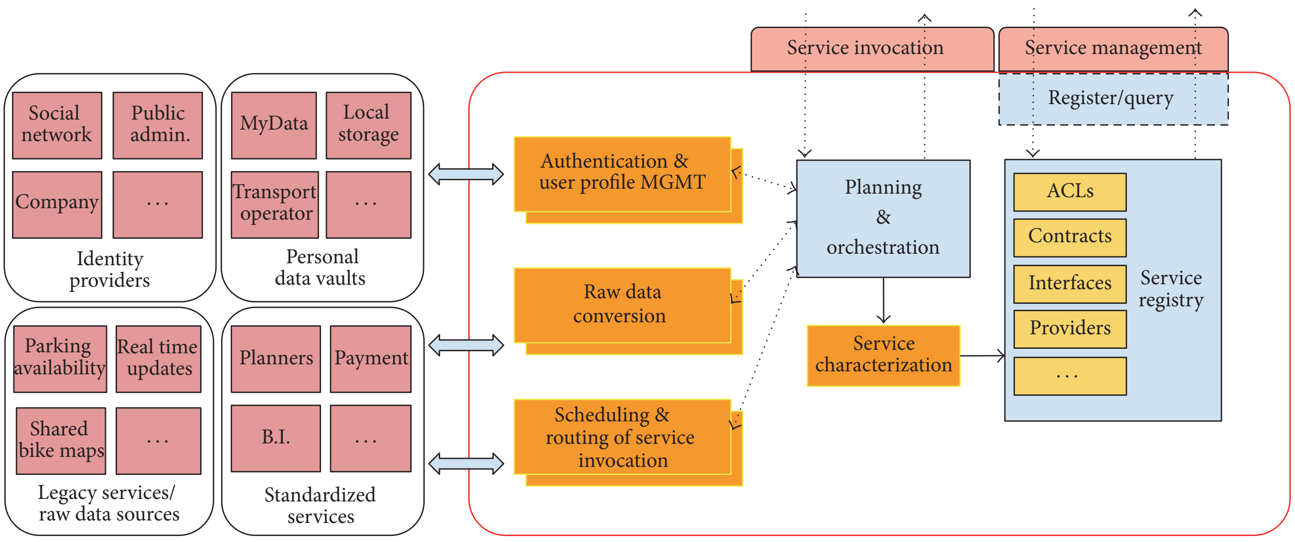


FIGURE 1: Architecture of a marketplace for mobility services.

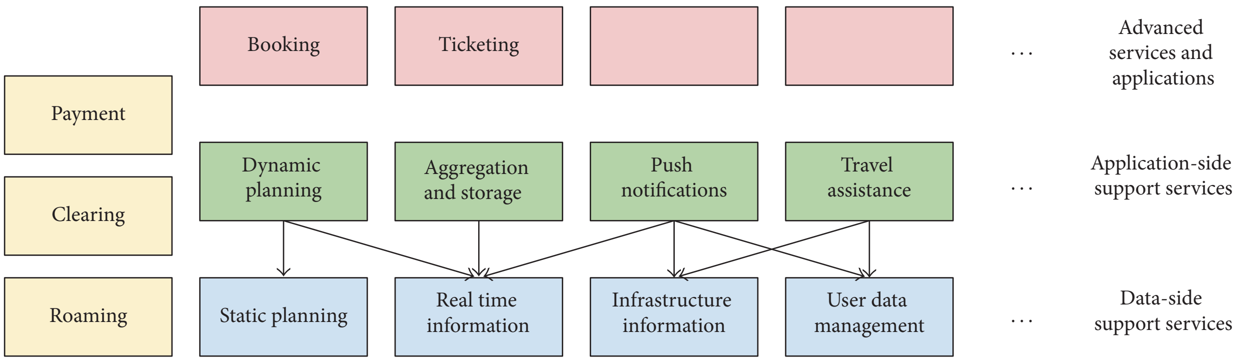


FIGURE 2: Service categories in SMAll.

Sharing databases through services instead of giving direct access means having a finer control on access policies, both in terms of simple access rights and in terms of precomputations that allow providing properly aggregated or otherwise sanitized data to applications. It is worth noting that security issues can emerge from this kind of open structure, yet the platform itself can play a crucial role in mitigating them [41, 42].

Generally speaking, a platform to "glue" mobility services together could enable the establishment of Mobility as a Service operators.

One way to develop a MaaS-enabling infrastructure is to structure it as a marketplace (Figure 1) for mobility services, where the definition of open standards for service invocation guarantees interoperability, the availability of infrastructural components (i.e., authentication, access control, QoS negotiation, and business intelligence) lowers the effort needed for the development of applications, and an orchestration framework streamlines the composition of available services into more complex applications. We are developing a prototype of this system called SMAll.

Indeed, we already classified some macrocategories of services that we can expect to find in such a marketplace. Figure 2 outlines some of the most important ones, arranged in layers of increasing complexity—in this context, "complex" means the creation of functionalities on top of other "simple" services. Starting from the bottom, we find services that are either wrappers for legacy software, for example, travel planners that do not include real-time functionalities, or services that process basic data. The aim of this class of services is to standardize the data and the interfaces of legacy software to make them available to other services. Other more complex services, found in the upper layers, orchestrate these basic ones to implement their behaviors, up to the very refined policies of MaaS operators and similar applications.

As mentioned at the beginning of the section, we envisage the adoption of microservices to provide seamless implementation of these categories of components:

(i) Wrappers converting legacy data source into standard services

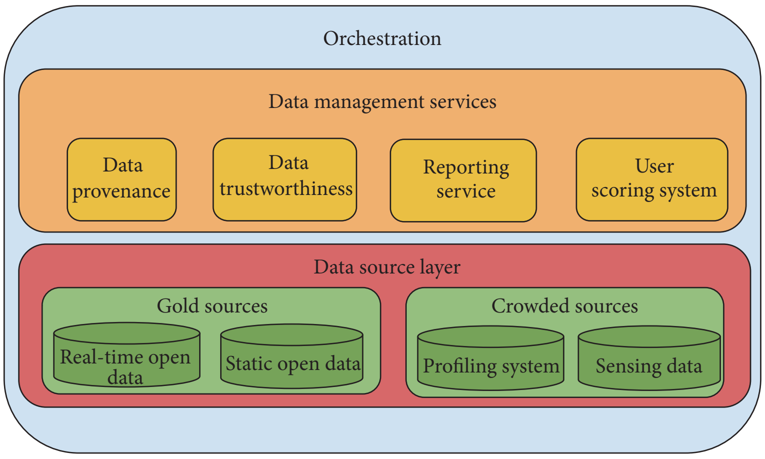


Figure 3: Organization of crowdsensing orchestration layers.

(ii) Helper services (e.g., authentication, authorization, scheduling, routing, and orchestration)

(iii) The service registry storing the definition of all the services deployed on the platform

(iv) The actual business logic deployed by operators or intermediaries, collecting, storing, and processing data to offer some data-related insight on the usage of services.

According to this model, there is no single actor responsible for data quality and service correctness, so we are introducing a layer of services to manage the quality issues of data resulting from crowdsourcing. This layer will offer metadata management services, such as the evaluation of data provenance, data reliability, and data trustworthiness, and the propagation of these indicators across services which compose data, ready to be exploited in coordination with any service that exposes data. The microservices architecture is suitable for this kind of scenario, because it allows creating independent services for specific tasks (and for different implementation of the same task) of the data quality management process. On the SMAll platform, it will be possible to exploit these services through orchestration.

The concept is illustrated in Figure 3.

On the bottom level, we have the services that expose data. These services are heterogeneous in terms of amount, sensitivity, expressiveness, representativeness, and so forth. Above the data level, there are the microservices in charge of the implementation of the mechanisms dealing with each of the data management problems described (provenance, trustworthiness, etc.). While the two levels are kept separated to highlight their different function, there is no hierarchical relation between them. From an architectural point of view, every box is a service, and their invocation is defined by a workflow, representing the desired data quality policies, and implemented through the orchestration mechanism.

4. Data Sources

Various kinds of data sources feed the system. We can classify them in broad categories, according to their provenance (e.g., official data about the transport infrastructure versus crowdsourced POIs) and their timescale (e.g., real-time information versus planned timetables and static features). Indeed, each stored data includes specific information and has peculiar characteristics depending on its own source. For instance, traffic data feeds are automatically posted and updated in the system; instead, the quantity and quality of crowdsourced/crowdsensed data are strongly influenced by the voluntary nature of the action and engagement of the participant [43].

In any case, all the data sources, independently of their category, will be accessed in a homogeneous fashion, through appropriate microservices.

4.1. Profiling System. To provide personalized services, we have to build a category of services that exploits a user (JSON-based) profile, structured in three interconnected parts: (A) the Generic Profile (GProfile) which includes some general data about the user, such as personal info, language, unit of measurement, device(s) in use, average walking speed, data about his/her credibility, and data about his/her favorite public means of transport routes; (B) the Urban Profile (UProfile), which describes users preferences related to the urban environment, expressed according to his/her needs, and preferences about the urban Point of Interests (POI); a specific section of such a part of the profile is devoted to describing the user's preferences about the urban barriers (such as stairs) and facilities (such as curb cuts); and (C) the eAccessibility Profile (eAProfile), which describes users preferences related to the e-accessibility and to the interface of the application.

4.1.1. Generic Profile. The Generic Profile describes the general information about the user. It includes personal data and data about the device in use, as well as the language and the unit of measurement. These latter data can be automatically set by the service, deriving them from users location, or manually set up by the user. In such part of the profile, the user can also declare his/her average speed when he/she moves in an urban environment. Alternatively, such data can be automatically derived from device sensor, which can track the users movement and then compute his/her average speed. This information is essential for our system, because the routing algorithms compute the best personalized paths taking them into account. For example, when combined to real time availability of buses (when the paths include the use of public means of transports), the user's ability to reach a stop in time to catch the bus prunes the set of feasible different paths. Finally, the user could store here information about his/her traveling habits, providing data about his/her favorite bus routes. The user can provide a location in the city by exploiting his/her current position or an address (i.e., street and number). Then, our system provides all the bus stops that the user can reach (in a configured time) with a list of the bus routes available at those stops; finally, the user can choose bus stops and routes of interest.

4.1.2. Urban Profile. The Urban Profile stores information about users preferences related to the urban environment. In particular, the urban elements are called Point of Interests (POIs) and users can set their preferences, classifying them as NEUTRAL, LIKE, UNLIKE, and AVOID on the basic of their degree of interest, preference, and/or need. Some examples of POIs mapped in our system are bus stops; subway stations; bicycle-sharing stations; parking; and so on. An interesting subset of POIs is related to identify urban barriers and facilities in the city. Such specific POIs are defined as aPOIs (accessibility Points of Interests). We have classified the aPOIs in categories that derive from the mobility context, in particular for those people with disabilities that we treat in the use cases of this work (see Section 7). These categories include items such as gaps, crosses, obstructions, and surface descriptions. Users have the possibility of defining their preferences about the above-listed aPOIs (stored in the Urban Accessibility Profile (UAProfile)) as follows:

(i) NEUTRAL: the user has neither difficulties nor preferences related to the aPOI type. The presence of this type of barrier/facility on a path is irrelevant to the user.

(ii) LIKE: the user prefers aPOIs of this type, when they are available. The presence of this type of barrier/facility on a path is positive to the user.

(iii) DISLIKE: the user can face this aPOI type, but with some efforts. In this case, an alternative path is preferred (when available), but it is not necessary. The presence of this type of barrier facility on a path is negative to the user.

(iv) AVOID: the user cannot face this aPOI type and an alternative path is necessary. The presence of this type of barrier/facility on a path prevents the user from following this path.

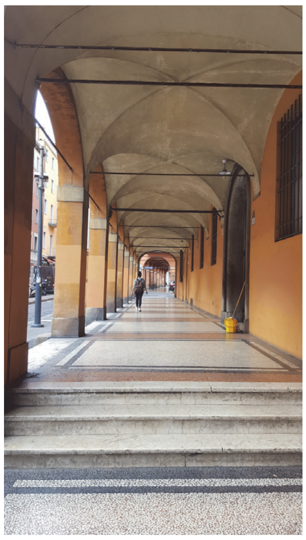


FIGURE 4: Stairs in Bologna porticos.

A more detailed description of such urban accessibility preferences can be found in [33, 44]. On the basis of them, our system computes an accessible route that comes across the LIKEd aPOIs when feasible, gets round the DISLIKE aPOIs if it is possible, and avoids the AVOIDed aPOIs every time. It is worth noting that positive preferences can be associated with barriers and negative preferences can be associated with facilities. As an example, a blind user can set as LIKE some specific barriers, such as stairs and steps, because they can represent a reference point. Analogously, wheelchair users can set tactile paving as DISLIKE, because such surfaces can be uncomfortable for them.

4.1.3. eAccessibility Profile. The e-Accessibility Profile is devoted to storing preferences and needs in terms of maps rendering. The main selection is the one related to textual/graphical representation of the map. On the basis of it, users can choose specific styles to represent POIs. For instance, the graphical representation can be personalized in terms of colors and size of the POIs icons in the map, addition of textual labels, and visualization (show or hide) of POI categories or of POI types. In particular, different style rules can be associated with the whole application, to a specific preference (LIKE, DISLIKE, etc.) or to a single type of POI.

4.2. Data Sensing. We designed and developed a specific sensing service prototype that would be exploited on users smartphones, with the aim of sensing stairs, automatically storing information about such a kind of urban barrier. Stairs are commonly placed in pedestrian areas of the urban environment, in particular in European cities, due to their old origins. As an example, we report in Figure 4 a picture taken in Bologna. Bologna is famous for its porticos, which are devoted to pedestrian paths all over the city (over 45 kilometers of arcades) and where stairs often affect the urban accessibility.

The design issues of such an ad hoc service were based on the need of low energy consumption and of high

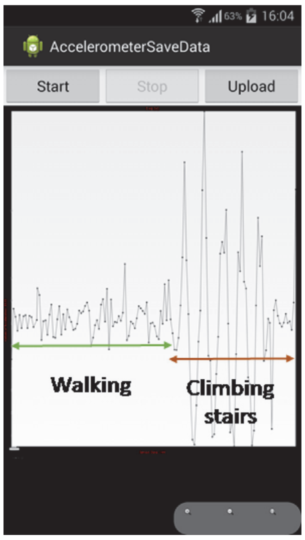


Figure 5: A screenshot of the signal sensed by means of the accelerometer.

precision, minimizing false positive and false negative results. Analyzing the sensors available on a smart phone, we focused on gyroscopes, accelerometers, and magnetometers. The idea was to create a service, which would be used in background, without affecting other uses activities, applying an opportunistic sensing.

Several methodologies have been evaluated, such as sensors fusion (combining data coming from the gyroscope, the accelerometers, and the magnetometer, with the aim of identifying the device inclination), Fourier analysis, Kalman filtering, convolution (cross-correlation and signal analysis of the forces applied on the device, with the aim of reconstructing and interpreting the device movements). We have exploited this latter method, by using only the accelerometer. In particular, our prototype compares the signal recorded by the smartphone accelerometer with a set of presampled ones, so as to assess the actual presence of a stair. Such presampled signals correspond to signals obtained climbing stairs up and down, by a group of different users equipped with their smartphones in different modalities (by walking, by running, and by keeping the smartphone in the pocket, in hand, in a bag, or in a backpack). The use of this method (and in particular of the sole accelerometer) lets us avoid the use of the gyroscope, then limiting the energy consumption and false positives and negatives.

The sensing prototype we have developed is based on Android operating systems; it exploits the spatial components thanks to the accelerometer, which senses the force applied to all spatial components. Once our sensing service prototype recognizes the presence of a stair, data about its position are sensed and stored. Hence, our prototype records the sensed data, analyses them, and stores the corresponding signal. A screenshot of a corresponding plot is shown in Figure 5.

4.3. Transit Infrastructure. Information regarding the operation of buses, trains, and other means of transport is possibly the most complete example of variety that benefits from the standardization offered by wrapper services.

(i) Operators are usually the authoritative source for static information about the transport infrastructure (stops, routes, etc.) and planned services (timetables, vehicles features, etc.). Operators can make this data available through different open data formats. GTFS [45] is rapidly growing to the status of de facto standard, yet many company-specific formats are still in use. A set of wrapper services is useful not only to convert these formats into a standard one but also to offer more sensible ways to access data, for example, allowing for discovering nearby stops given an address or set of coordinates, to know the set of bus lines serving a given stop, and so forth.

(ii) Real-time information about the transport services is, again, usually provided by operators. Depending on the end-user needs, it could be useful to know either the position of a vehicle or its delay with respect to planned operation or the estimated time of arrival at a given stop. Of course, these data are all mutually related, and it turns out that different operators may decide to provide different views of the same basic information (in our region, e.g., the biggest bus operator provides the arrival time of the next two buses at a given stop to the public, but at the same time, it feeds the "raw" GPS position of each vehicle to the regional transport authority, which is considering to make these data available for crowdsourced applications). By wrapping the composition between the available kind(s) of data and other information (e.g., position of vehicles crowdsensed by passengers, travel times measured on street segments), it is possible to obtain all the needed views and even to improve the precision of estimates.

(iii) Real-time information about the transport infrastructure comes from many different sources, such as public administrations announcing planned or extraordinary works, emergency teams intervening on accidents, operators giving notice of strikes of vehicle failures, weather reports, and of course people in the streets.

5. Data Quality Management Services

Various kinds of data sources feed the system. We can classify them in broad categories, according to their provenance (e.g., official data about the transport infrastructures versus crowdsourced POIs) and their timescale (e.g., real-time information versus planned timetables and static features).

Indeed, each stored data includes specific information and has peculiar characteristics depending on its own source. For instance, traffic data feeds are automatically posted and updated in the system; instead, the quantity and quality of crowdsourced/crowdsensed data are strongly influenced by the voluntary nature of the action and engagement of the participant [43].

In any case, all the data sources, independently of their category, will be accessed in a homogeneous fashion, through appropriate microservices.

5.1. Data Provenance. Data Provenance for single hosts sources is a known problem in literature. According to works like [46], this problem could be solved only with a creation of private and public key system for data stream certification. A good reference is the system developed in [47], describing a cryptographic provenance verification approach for ensuring data properties and integrity for single hosts. Specifically, the authors designed and implemented an efficient cryptographic protocol that enforces keystroke integrity. This kind of protocol can be integrated as a microservice in our architecture. However, public-key schemes are known for their significant computational load, thus existing techniques may not be suitable for high-rate, high-volume data sources. Moreover, there could be the need for an algorithm for the propagation of provenance data. In some cases, data originated from the composition of raw (or otherwise "lower ranked") sources should be accompanied by suitable metadata that allows verifying the provenance of the input values, in a cryptographically strong way. Merkle hash trees could be a good candidate to build proofs for composed data pieces [48].

5.2. Data Trustworthiness. Trustworthiness often referred to measuring and quantifying the quality of information coming from online resources and systems [49]. Several studies have been conducted with the aim of supporting users in quickly judging the trustworthiness of information they get, providing automatically computed values, which can be continuously updated [49, 50]. The authors of [51] based the trustworthiness model on users mobility and on the usefulness of their past contributions to the system. This work focuses on data integrity (for data coming from automatic readings from devices), data correctness, and quality. Users contributions are compared with those ones provided by local authoritative data sources, certified by the data provenance microservice. The trustworthiness microservice considers information provided by authoritative data sources (i.e., local administrations, municipalities) as a gold set. Thus, our idea is to compare information provided by users with trustworthy and correct data. Hence, it is possible to base our trustworthiness service on the computation and assignment of more effective credibility values to users, similar to what has been done in other works, for example, [52].

5.3. Data Reliability and Reporting Service. Once we are able to verify the provenance and trustworthiness of the data intended as verification of the correct elaboration process, we have to verify that the results or the data displayed are actually correct. The process of correctness verification of the results of a crowdsourced data can be done in two ways: through an automated system with artificial intelligence embedded or through a reporting system with a trusted source approach. Considering that this work is mainly aimed at helping disabled people, who are known to be more collaborative in using reporting systems, has obviously led us to implement the second solution. The description of the reporting system for our architecture is inspired from the mPASS model [32], which is based on the mapping of POI. Each POI and its related data can be added to our system by means of one or more reports. Reports are classified in three different source classes, according to how they are collected. The three source classes have a growing validity:

(i) U-report (report obtained by users): users can add POI to the DB system. This can be done in two ways: (i) spontaneously, a user encountering a specific barrier or an accessibility facility can send a report to the reporting service (RS); (ii) on demand, the RS can ask users to improve validity of an existing POI (usually a POI reported by sensors). Hence, the system will exploit the user report instead of sensor ones and the user gets an award badge on his/her public profile.

(ii) S-report (report obtained by sensors): the RS can automatically produce data by sensing from mobile devices sensors. These reports are supposed to have a low validity.

(iii) E-report (report produced by experts): experts are people working for organizations involved in monitoring urban accessibility (such as local administrations and municipalities or disability right organizations).

Being professionally able to correctly classify and measure every kind of POI and POIs, their reports are considered totally valid. Reports from administrators can be added in two ways: (i) spontaneously: administrators add reports according to their program of activities, sending to the RS reports on barriers or accessibility facilities; (ii) on demand: the RS can ask administrators to improve validity of an existing POI (usually a user-added one). Hence, the system will use the administrator report instead of user ones. Hence, the RS can have more reports of the same POI, classified with one or more different source classes. Both the map provided to users and the data set considered by the routing algorithm are based on the more valid reports available. For example, if a POI is added by both sensors and users, U-reports are used instead of S-reports, since they are considered more valid. Analogously, if a POI is added by both users and administrators, E-reports are used instead of U-reports, because they are considered more valid. To populate the RS database, we also added some POIs and reports obtained by converting, filtering, and mashing up existing data.

5.4. Feedback Scoring System. The Feedback Scoring System service is linked with the reporting service, an algorithm that calculates the reliability of a report based on the assigned scores on the basis of certain characteristics. It can happen, however, that in some cases these reports are uncertain, or missing, or simply they are too few to yield a reliable result. In this case, we can ask for the user interaction in order to give a feedback of a specific case. When uncertainty occurs on a POI, we activate a simple mechanism of user request, asking to confirm the presence/absence of this POI or to confirm

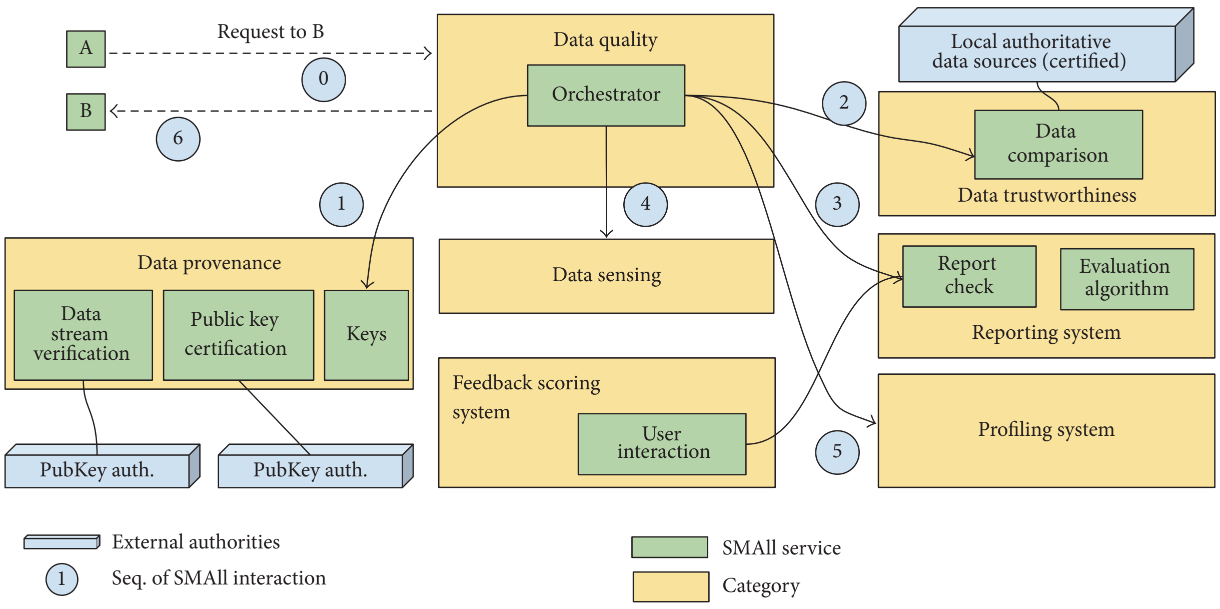


FIGURE 6: Example of orchestration workflow.

parameters about measures of this POI. This feedback cannot always be sharp; it can include a confidence score, showing how much the user trusts the POI features to be correct. This score will be used to recalculate the reliability of the crowdsourced data.

6. Orchestration

The core of this architecture is the orchestration process. As previously described, as service orchestration we mean the composition of microservices, tools, and processes invoked and the connection and automation of workflows to deliver a defined service [53]. To this end, our platform can natively run orchestration tasks written in Jolie [54], a programming language offering several structural advantages: it provides workflow constructs such as sequence, parallelism, and nondeterministic choice for composing communication interactions, it deals with statefulness by activating different workflow instances for each business task to manage, and it implements interfaces to almost every communication protocol commonly used.

Figure 6 represents one of the possible workflows managed by the orchestrator. The idea is that the evaluation of data quality is carried out by suitably combining one or more specific microservices.

In our case, the workflow represents the composition of data management services that, according to the legacy service policy, will produce results and an evaluation of their quality at the same time. To do that, the orchestrator begins invoking a service of the data management layer, in this case the data provenance service that certifies the provider. The results can be used by the service caller to refine authentication data as well feed back the data provenance service. This result will improve the data quality evaluation in subsequent data source service invocations.

7. Use Cases

In order to prove the effectiveness of our approach, we tested our system with many different user profiles (such as users with reduced mobility, elderly people, blind users, and users with low vision). In this section, we present two scenarios illustrating urban accessibility issues involving a wheelchair user and an elderly user. More generally, different scenarios can be pictured, involving all the different aspects of the smart mobility context. For instance, an interesting use case can be envisioned by considering a bicycle-sharing system together with subway/bus stops: real time subway/bus information are automatically provided by the transportation provider company, while data related to the bicycle-sharing stations with available bikes or open docks are crowdsourced by users and/or derived by crowdsensed data obtained in the activity of bike block/unblock, exploiting, for example, RFID/NFC technologies and GPS position. In this way, the SMAll system can compute personalized paths by taking into account the actual time of the interested subway or bus and the effective availability of bikes or of open docks suggesting the best bicycle-sharing station to reach, according to the defined destination of the route.

Another interesting use case that we are currently developing involves particular rural areas, whose main features are a low population density and a difficult road conditions due to rugged environmental conditions.

For these particular areas, in general, many public transport services such as buses and trains are not economically justified by the current demand. Conversely, however, more accessible urban public transport services become essential to reach other important social services such as healthcare which are often far apart.

SMAll provides the following solution. The public transport network is acting as a targeted service on request for each applicant. A network of taxis satisfies every single request.

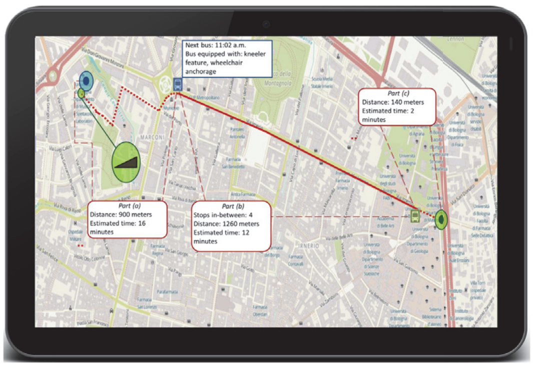

Next bus: 11:02 a.m.
Bus equipped with: kneeler feature, wheelchair anchorage
Part (a)
Distance: 900 meters
Estimated time: 16 minutes
Part (b)
Stops in-between: 4
Distance: 1260 meters
Estimated time: 12 minutes
Part (c)
Distance: 140 meters
Estimated time: 2 minutes

Figure 7: Path proposed by our system, tailored on a wheelchair user profile.

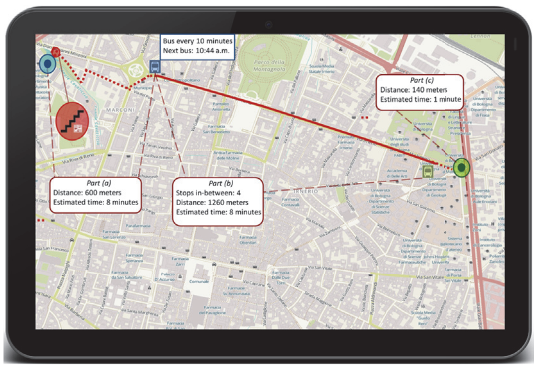


Figure 8: Path proposed by our system, tailored on an elderly user profile.

The SMAll task is to coordinate the various taxi operators who have the assigned races, from call handling to profit redistribution. SMAll would deal to merge close calls in an efficient way (e.g., Uber Pool).

The advantage is twofold, the administration spends less to provide a service and the quality of the service for citizen is improved. Moreover, this can be considered an example of fostering and supporting community awareness in rural area [55].

In the two user cases here detailed, the users request personalized paths, by using their own smartphones. In particular, let us consider a male user equipped with a manual wheelchair (first scenario) and an elderly woman (second scenario); both of them ask for a specific path (including bus routes) in the city of Bologna (Italy), with the same starting point A and the same destination B (shown in Figures 7 and 8). The path usually proposed by the most commonly used geospatial mapping platforms (e.g., Google Maps, Bing Maps) takes 17 minutes as a whole and is structured in three parts:

(i) A pedestrian part to reach the bus stop: this part is supposed to take 8 minutes to the user.

(ii) A part of a bus route (from the blue bus stop to the green bus stop): this part is supposed to take 8 minutes (with four in-between stops).

(iii) Another pedestrian part from the arrival bus stop to the final destination: this part is supposed to take 1 minute.

This path presents some issues our users have to face:

(1) There is a stair in the first pedestrian part of the path and there is no information about its presence; this means that our wheelchair user cannot afford the suggested pathway, but he has to find another alternative and accessible route.

(2) There is no information about accessibility of the public mean of transport and of the bus stops; in particular, not all the vehicles are provided with facilities to support our specific user, such as ramps, kneeler features, and lifts.

(3) Estimated time to reach the departure bus stop from the starting point (8 minutes, for 600 meters) is computed taking into account abilities and speed of an average user, instead of considering the actual abilities average speed of our specific users.

(4) Information about bus arrival time is derived from a time table, instead of referring to the real bus position and availability.

The following subsections detail the scenarios about the sensing and the data consuming activities of two different users with different needs and preferences about the urban environment. The design of the interface is under investigation and some preliminary results can be found in [56].

7.1. *First Scenario.*

As a first scenario, let us consider a wheelchair user who asks for an accessible path starting from A to the destination B. He has set up his UAProfile declaring that he stated as LIKE ramps and curb cuts (as gap facilities), parking slots reserved to people with disabilities (as parking facility), sidewalks with an adequate width (in the pathway category), and zebra crossing and traffic lights (as crossing facilities). He initialized uneven road surface and tactile paving (in the surface category) as DISLIKE and Gap category aPOIs and obstructions barriers as AVOID. Handrails and audible traffic lights are NEUTRAL for him, as well as street lighting. Algorithm 1 shows a fragment of his profile in JSON format.

When this user asks for a path from the starting point A to the destination B, then our system computes a personalized route taking into account the users profile (i.e., avoiding such barriers which affect him and including as much as possible the LIKEd facilities).

Our system computes a personalized path, by taking into account real data about bus availability and the users profile, in terms of barriers to avoid, LIKEd facilities to include as much as possible, and users personal average speed (set up as 0.98 m/s, according to [57]). This path is structured in three parts (shown in Figure 7), where only the first part is different from the path previously described. In particular,

(1) our path suggests a different first pedestrian part of the path, taking into account the presence of that stair,

```
{
  "UAProfile": {
    "style": {
      "neutral": {
        "_style": "hidden"
      },
      "like": {
        "_style": "ok"
      },
      "dislike": {
        "_style": "warning"
      },
      "avoid": {
        "_style": "alert"
      }
    },
    "gap": {
      "steps": {
        "_type": "barrier",
        "_pref": "avoid"
      },
      "gaps": {
        "_type": "barrier",
        "_pref": "avoid"
      },
      "stairs": {
        "_type": "barrier",
        "_pref": "avoid"
      },
      "ramps": {
        "_type": "facility",
        "_pref": "like"
      },
      . . .
    },
    "crossing": {
      "zebra_crossings": {
        "_type": "facility",
        "_pref": "like"
      },
      . . .
    },
    "parking": {
      "slots_for_disabled": {
        "_type": "facility",
        "_pref": "like"
      }
    },
    "pathway": {
      "sidewalk": {
        "_width": "90",
        "_units": "cm",
        "_pref": "like",
        "_style": "emphasis"
      }
    }
  . . .
}
```

Algorithm 1

and finds an alternative accessible path, including a ramp (highlighted in Figure 7 with a green icon);

(2) information about the accessibility of the public means of transport is provided; in particular, the path is computed taking into account a bus equipped with a kneeler and wheelchair anchorage features;

(3) estimated time to reach the departure bus stop from the starting point is computed taking into account our specific users abilities and average speed, as declared in his profile (16 minutes, for 900 meters);

(4) information about bus arrival time is provided taking into account open data about the real bus position and eventual delays, provided by the local public means of transport operator.

The time to complete the path is estimated to be 30 minutes and it is computed according to the users average speed and real bus availability (by considering real time data about eventual delays, traffic, and so on, coming from open data made available by the public transportation provider), as follows: 16 minutes for the first part, 12 minutes for the second part, and 2 minutes for the last one. Meanwhile, crowdsensing and crowdsourcing services are exploited on the user's mobile device, with the aim of collecting data and reports about urban barriers and facilities.

7.2. Second Scenario. As a second scenario, let us consider an elderly woman who asks for a path from A to B, tailored according to her preferences. She has set up her UAProfile declaring that she stated as LIKE streets lighting, crossing facilities, sidewalks, ramps, curb cuts, and handrails. She also stated as LIKE stairs, because her doctor suggested her to do some exercise, climbing stairs. She stated as DISLIKE garbage bins, while steps, gaps, uneven road surface, and tactile paving are NEUTRAL. Algorithm 2 shows a fragment of her profile in JSON format.

Once such a user asks for a pedestrian path, our system computes a personalized route from the starting point (A) to the destination point (B) taking into account her profile (i.e., stairs) and real data about bus availability. Also in this case the personalized path is structured in three parts and it is similar to the one previously described, including the stairs in its first part. Since this user is equipped with a smart phone, she would actively provide data coming from her mobile device accelerometer, so as to enrich the available information that are exploited by SMAll, with the aim of equipping citizens with smart mobility applications and data.

8. Conclusion

Smart mobility is a key point in supporting citizens in their daily activities and in offering them a feasible smart city. Information about urban transportation (including taxis, buses, trains, and car-sharing), urban barriers and facilities, and pedestrian and multimodal paths would be of great benefit in this context, as well as all the information about the whole experience of traveling and wandering the city, including travel planning and payments. Crowdsensing and

```
{
  "UAProfile": {
    "style": {
      "neutral": {
        "_style": "hidden"
      },
      "like": {
        "_style": "ok"
      },
      "dislike": {
        "_style": "warning"
      },
      "avoid": {
        "_style": "alert"
      }
    },
    "gap": {
      "steps": {
        "_type": "barrier",
        "_pref": "neutral"
      },
      "gaps": {
        "_type": "barrier",
        "_pref": "neutral"
      },
      "stairs": {
        "_type": "barrier",
        "_pref": "like"
      },
      "ramps": {
        "_type": "facility",
        "_pref": "like"
      },
      "curbcuts": {
        "_type": "facility",
        "_pref": "like"
      },
      ...
    },
    "crossing": {
      "zebra_crossings": {
        "_type": "facility",
        "_pref": "like"
      },
      "traffic_lights": {
        "_type": "facility",
        "_pref": "like"
      },
      ...
    },
    ...
    }
  }
}
```

Algorithm 2

Mobility as a Service can play a key role in this background. As discussed in Sections 3–6, in providing a complete and smart urban mobility service, different requirements need to be considered and orchestrated. In particular, an efficient service-oriented approach for smart mobility needs (i) real time data about public means of transport; (ii) updated urban data collected via crowdsensing and crowdsourcing; (iii) a model able to calculate the trustworthiness of collected data; (iv) a definition of a precise profile according to user's preferences and needs. Keeping into account these design issues, we designed and prototyped an infrastructure as a marketplace for mobility services, called Smart Mobility for All (SMAll). A prototype of such infrastructure has been developed and its architecture has been described in the paper, as well as some of the provided services. In particular, two use cases have been presented, focusing on a wheelchair user and an elderly person. We are now doing further studies with the aim of profiling users by tracking their daily journeys, by exploiting machine learning techniques, integrating them in crowdsensing activities. Adaptation mechanisms will be applied to the profile, so as to dynamically and automatically modify it according to users actual abilities and habits. The adopted SOA approach will make all future additions easy to integrate, since each new algorithm or service will be developed as an independent microservice and plugged into the orchestration logic as needed.

As future work, we are planning to conduct the evaluation of the system in terms of (i) efficiency, scalability, and robustness and (ii) effectiveness, user experience, and usability.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] R. Giffinger and H. Gudrun, "Smart cities ranking: an effective instrument for the positioning of the cities?" *ACE: Architecture, City and Environment*, vol. 4, no. 12, pp. 7–26, 2010.

[2] WHO, Urban health: major opportunities for improving global health outcomes, despite persistent health inequities, http://www.who.int/mediacentre/news/releases/2016/urban-health-report/en/.

[3] H. Chourabi, T. Nam, S. Walker et al., "Understanding smart cities: an integrative framework," in *Proceedings of the 45th IEEE Hawaii International Conference on System Sciences (HICSS '12)*, pp. 2289–2297, Maui, Hawaii, USA, January 2012.

[4] Á. Petkovics, V. Simon, I. Gódor, and B. Böröcz, "Crowdsensing solutions in smart cities: introducing a horizontal architecture," in *Proceedings of the 13th International Conference on Advances in Mobile Computing and Multimedia (MoMM '15)*, pp. 33–37, ACM, 2015.

[5] H. Kukka, J. Ylipulli, A. Luusua, and A. K. Dey, "Urban computing in theory and practice: towards a transdisciplinary approach," in *Proceedings of the 8th Nordic Conference on Human-Computer Interaction (NordiCHI '14)*, pp. 658–667, ACM, Helsinki, Finland, October 2014.

[6] K. P. Sami Pippuri, Sampo Hietanen. Maas finland, http://maas.fi/.

[7] S. Newman, *Building Microservices*, O'Reilly Media, Farnham, UK, 2015.

[8] V. Kostakos, T. Ojala, and T. Juntunen, "Traffic in the smart city: exploring city-wide sensing for traffic control center augmentation," *IEEE Internet Computing*, vol. 17, no. 6, pp. 22–29, 2013.

[9] S. Patidar, D. Rane, and P. Jain, "A survey paper on cloud computing," in *Proceedings of the 2nd International Conference on Advanced Computing and Communication Technologies (ACCT '12)*, pp. 394–398, January 2012.

[10] P. Banerjee, R. Friedrich, C. Bash et al., "Everything as a service: powering the new information economy," *Computer*, vol. 44, no. 3, Article ID 5719575, pp. 36–43, 2011.

[11] M. Zhou, R. Zhang, D. Zeng, and W. Qian, "Services in the cloud computing era: a survey," in *Proceedings of the 4th International Universal Communication Symposium (IUCS '10)*, pp. 40–46, IEEE, Beijing, China, October 2010.

[12] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *European Transactions on Telecommunications*, vol. 25, no. 1, pp. 81–93, 2014.

[13] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, "Vision and challenges for realising the Internet of Things," *EUR-OP*, vol. 20, no. 10, 2010.

[14] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a service: challenges, solutions and future directions," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3733–3741, 2013.

[15] X. Sheng, X. Xiao, J. Tang, and G. Xue, "Sensing as a service: a cloud computing system for mobile phone sensing," in *Proceedings of the 11th IEEE SENSORS 2012 Conference*, pp. 1–4, IEEE, Taipei, Taiwan, 2012.

[16] A. Melis, S. Mirri, C. Prandi, M. Prandini, P. Salomoni, and F. Callegati, "Crowdsensing for smart mobility through a serviceoriented architecture," in *Proceedings of the 2nd IEEE International Smart Cities Conference*, 2016.

[17] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[18] G. Cardone, L. Foschini, P. Bellavista et al., "Fostering participaction in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 112–119, 2013.

[19] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[20] M. Talasila, R. Curtmola, and C. Borcea, "Improving location reliability in crowd sensed data with minimal efforts," in *Proceedings of the 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC '13)*, pp. 1–8, IEEE, Dubai, UAE, April 2013.

[21] A. Vemula, N. Patil, V. Paharia et al., "Improving public transportation through crowd-sourcing," in *Proceedings of the 7th International Conference on Communication Systems and Networks (COMSNETS '15)*, pp. 1–6, Bangalore, India, January 2015.

[22] Á. Petkovics and K. Farkas, "Efficient event detection in public transport tracking," in *Proceedings of the International Conference on Telecommunications and Multimedia (TEMU '14)*, pp. 74–79, July 2014.

[23] Waze, crowdsensing applicaiton, http://www.waze.com/.

[24] A. Sassi and F. Zambonelli, "Coordination infrastructures for future smart social mobility services," *IEEE Intelligent Systems*, vol. 29, no. 5, pp. 78–82, 2014.

[25] J. Goncalves, S. Hosio, J. Rogstadius, E. Karapanos, and V. Kostakos, "Motivating participation and improving quality of contribution in ubiquitous crowdsourcing," *Computer Networks*, vol. 90, pp. 34–48, 2015.

[26] P. Salomoni, C. Prandi, M. Roccetti, V. Nisi, and N. J. Nunes, "Crowdsourcing urban accessibility: some preliminary experiences with results," in *Proceedings of the the 11th Biannual Conference on Italian SIGCHI Chapter*, pp. 130–133, ACM, Rome, Italy, September 2015.

[27] C. Prandi, V. Nisi, P. Salomoni, and N. J. Nunes, "From gamification to pervasive game in mapping urban accessibility," in *Proceedings of the the 11th Biannual Conference*, pp. 126–129, Rome, Italy, September 2015.

[28] C. Prandi, M. Roccetti, P. Salomoni, V. Nisi, and N. J. Nunes, "Fighting exclusion: a multimedia mobile app with zombies and maps as a medium for civic engagement and design," *Multimedia Tools and Applications*, 2016.

[29] F. Zambonelli, "Pervasive urban crowdsourcing: visions and challenges," in *Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops '11)*, pp. 578–583, IEEE, Seattle, Wash, USA, March 2011.

[30] N. Bicocchi, A. Cecaj, D. Fontana, M. Mamei, A. Sassi, and F. Zambonelli, "Collective awareness for human-ICT collaboration in smart cities," in *Proceedings of the IEEE 22nd International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '13)*, pp. 3–8, Hammamet, Tunisia, June 2013.

[31] S. Mirri, L. A. Muratori, and P. Salomoni, "Monitoring accessibility: large scale evaluations at a Geo-political level," in *Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '11)*, pp. 163–170, ACM, October 2011.

[32] C. Prandi, P. Salomoni, and S. Mirri, "mPASS: integrating people sensing and crowdsourcing to map urban accessibility," in *Proceedings of the IEEE 11th Consumer Communications and Networking Conference (CCNC '14)*, pp. 591–595, Las Vegas, Nev, USA, January 2014.

[33] S. Mirri, C. Prandi, P. Salomoni, F. Callegati, and A. Campi, "On combining crowdsourcing, sensing and open data for an accessible smart city," in *Proceedings of the 8th International Conference on Next Generation Mobile Apps, Services and Technologies (NGMAST '14)*, pp. 294–299, IEEE, Oxford, UK, September 2014.

[34] S. Choi, R. Lemay, and J.-H. Youn, "On-board processing of acceleration data for real-time activity classification," in *Proceedings of the IEEE 10th Consumer Communications and Networking Conference (CCNC '13)*, pp. 68–73, IEEE, January 2013.

[35] A. Anjum and M. U. Ilyas, "Activity recognition using smartphone sensors," in *Proceedings of the IEEE 10th Consumer Communications and Networking Conference (CCNC '13)*, pp. 914–919, IEEE, Las Vegas, Nev, USA, January 2013.

[36] A. Bujari, B. Licar, and C. E. Palazzi, "Movement pattern recognition through smartphone's accelerometer," in *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC '12)*, pp. 502–506, January 2012.

[37] M. B. Kjærgaard, M. Wirz, D. Roggen, and G. Tröster, "Detecting pedestrian flocks by fusion of multi-modal sensors in mobile phones," in *Proceedings of the ACM Conference on Ubiquitous Computing (UbiComp '12)*, pp. 240–249, 2012.

[38] Y. Iwasawa, K. Nagamine, I. E. Yairi, and Y. Matsuo, "Toward an automatic road accessibility information collecting and sharing based on human behavior sensing technologies of wheelchair users," *Procedia Computer Science*, vol. 63, pp. 74–81, 2015.

[39] Y. Iwasawa, K. Nagamine, Y. Matsuo, and I. Eguchi Yairi, "Road sensing: personal sensing and machine learning for development of large scale accessibility map," in *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, pp. 335–336, ACM, Lisbon, Portugal, October 2015.

[40] G. Merlino, S. Arkoulis, S. Distefano, C. Papagianni, A. Puliafito, and S. Papavassiliou, "Mobile crowdsensing as a service: a platform for applications on top of sensing clouds," *Future Generation Computer Systems*, vol. 56, pp. 623–639, 2016.

[41] F. Callegati, A. Campi, A. Melis, M. Prandini, and B. Zevenbergen, "Privacy-preserving design of data processing systems in the public transport context," *Pacific Asia Journal of the Association for Information Systems*, vol. 7, no. 4, 2015.

[42] A. Melis, M. Prandini, L. Sartori, and F. Callegati, "Public transportation, IoT, trust and urban habits," in *Internet Science*, F. Bagnoli, A. Satsiou, I. Stavrakakis et al., Eds., vol. 9934 of *Lecture Notes in Computer Science*, pp. 318–325, Springer, Berlin, Germany, 2016.

[43] M. Roccetti, S. Ferretti, C. E. Palazzi, P. Salomoni, and M. Furini, "Riding the web evolution: from egoism to altruism," in *Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC '08)*, pp. 1123–1127, IEEE, Las Vegas, Nev, USA, January 2008.

[44] S. Mirri, C. Prandi, and P. Salomoni, "A context-aware system for personalized and accessible pedestrian paths," in *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS '14)*, pp. 833–840, July 2014.

[45] Google, "General Transit Feed Specification," https://developers .google.com/transit/gtfs/reference.

[46] Y. L. Simmhan, B. Plale, and D. Gannon, *A Survey of Data Provenance Techniques*, vol. 47405, Computer Science Department, Indiana University, Bloomington, Ind, USA, 2005.

[47] K. Xu, H. Xiong, C. Wu, D. Stefan, and D. Yao, "Data-provenance verification for secure hosts," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 173–183, 2012.

[48] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology—CRYPTO '87*, vol. 293 of *Lecture Notes in Computer Science*, pp. 369–378, Springer, Berlin, Germany, 2000.

[49] C. Shahabi, "Towards a generic framework for trustworthy spatial crowdsourcing," in *Proceedings of the 12th International ACM Workshop on Data Engineering for Wireless and Mobile Acess (MobiDE '13)*, pp. 1–4, ACM, New York, NY, USA, June 2013.

[50] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on amazon mechanical turk," in *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP '10)*, pp. 64–67, ACM, New York, NY, USA, 2010.

[51] P. Gilbert, L. P. Cox, J. Jung, and D. Wetherall, "Toward trustworthy mobile sensing," in *Proceedings of the 11th Workshop on Mobile Computing Systems and Applications (HotMobile '10)*, pp. 31–36, Annapolis, Md, USA,, February 2010.

[52] C. Prandi, S. Ferretti, S. Mirri, and P. Salomoni, "Trustworthiness in crowd- sensed and sourced georeferenced data," in *Proceedings of the 13th IEEE International Conference on Pervasive Computing and Communication (PerCom '15)*, pp. 402–407, March 2015.

[53] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall PTR, River Edge, NJ, USA, 2005.

[54] F. Montesi, C. Guidi, and G. Zavattaro, "Composing services with JOLIE," in *Proceedings of the 5th IEEE European Conference on Web Services (ECOWS '07)*, pp. 13–22, Halle, Germany, November 2007.

[55] N. Taylor and K. Cheverst, "Supporting community awareness with interactive displays," *Computer*, vol. 45, no. 5, Article ID 6171144, pp. 26–32, 2012.

[56] S. Mirri, C. Prandi, and P. Salomoni, "Personalizing pedestrian accessible way-finding with mPASS," in *Proceedings of the 13th IEEE Annual Consumer Communications & Networking Conference (CCNC '16)*, pp. 1119–1124, Las Vegas, Nev, USA, January 2016.

[57] M. L. Tolerico, D. Ding, R. A. Cooper et al., "Assessing mobility characteristics and activity levels of manual wheelchair users," *Journal of Rehabilitation Research and Development*, vol. 44, no. 4, pp. 561–571, 2007.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 9178539, 15 pages
http://dx.doi.org/10.1155/2016/9178539

Research Article

Human Mobility Modelling Based on Dense Transit Areas Detection with Opportunistic Sensing

Fernando Terroso-Sáenz, Mercedes Valdes-Vela, Aurora González-Vidal, and Antonio F. Skarmeta

Department of Information and Communications Engineering, Computer Science Faculty, University of Murcia, Murcia, Spain

Correspondence should be addressed to Fernando Terroso-Sáenz; fterroso@um.es

Received 20 May 2016; Accepted 28 June 2016

Academic Editor: Celimuge Wu

With the advent of smartphones, opportunistic mobile crowdsensing has become an instrumental approach to perceive large-scale urban dynamics. In this context, the present work presents a novel approach based on such a sensing paradigm to automatically identify and monitor the areas of a city comprising most of the human transit. Unlike previous approaches, the system performs such detection in real time at the same time the opportunistic sensing is carried out. Furthermore, a novel multilayered grill partitioning to represent such areas is stated. Finally, the proposal is evaluated by means of a real-world dataset.

1. Introduction

For the last years, smartphones have been the center of most of the technological advances due to their growing popularity. As a result of these improvements, they are now equipped with several sensors like GPS, accelerometer, microphone, and so forth.

This palette of sensors allows capturing a large amount of contextual information related to the phone's holders and their surrounding environment [1]. This has eased the development of the *mobile crowdsensing* (MCS) or *human/people sensing* paradigm so as to perceive large-scale phenomena that can not be detected at an individual level [2].

One of the most useful phenomena to be perceived is human dynamics. Due to the steady improvement of the positioning sensors installed in mobile devices or vehicles and the fact that location is the most critical element in reflecting users' movement, the *mobility mining* discipline is one of the domains where MCS has been most widely applied so as to uncover different human mobility aspects [3]. In turn, this eases the deployment of innovative location-based services like predictive queries for moving object databases [4] or pervasive navigation systems [5].

Although several studies already exist, mobility mining solutions based on MCS still face the following challenges.

(i) The intensive usage of the positioning and communication capabilities of mobile devices required by this type of solution is rather battery draining. This is an important barrier for users to contribute to MCS-based solutions for mobility detection. Nonetheless, when it comes to composing detailed mobility models, most works assume that a large set of users are always available to report their location traces.

(ii) Existing mechanisms usually follow an offline mining process of a previously gathered mobility dataset. As a result, the extracted knowledge is fixed and can not smoothly adapt to changes of human dynamics.

(iii) Last but not least, location data is quite sensitive in terms of privacy for many people. Therefore, mobility models should be designed so that they can not be used to uncover meaningful places or routes of particular users.

In this context, the present work proposes an innovative framework for human mobility modelling with opportunistic

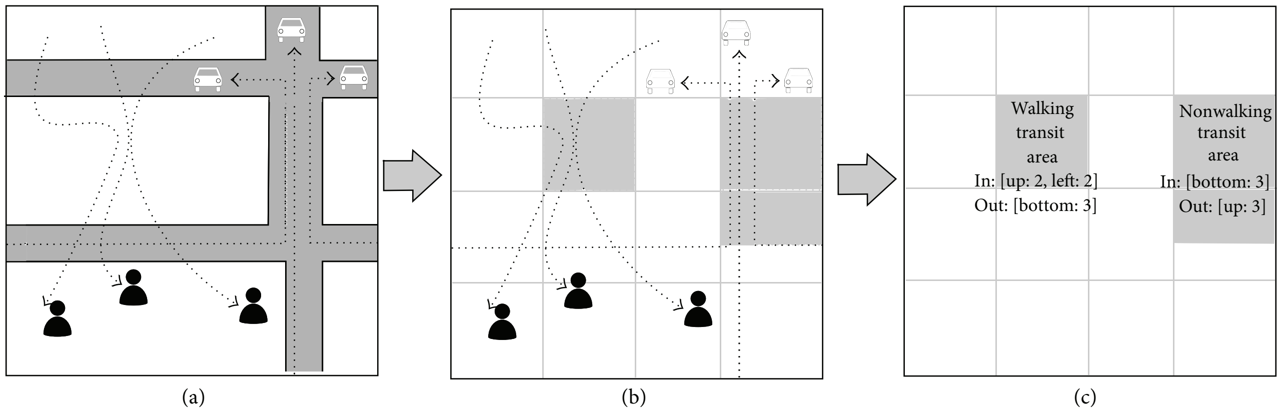


Figure 1: Overview of transit areas detection in the 2D space. (a) Users' raw trajectories, (b) detection high transit areas, and (c) transit areas parameters extraction.

MCS that considers the aforementioned open challenges. The key goal of the proposal is to detect regions with a high density of human transit that capture most of the mobility of a city.

In large city deployments, the number of these dense transit areas is usually very high. Hence, in order to ease their storage and management, a novel region abstraction for mobility mining is introduced. As Figure 1 shows, the idea is to represent such regions of interest with basic geometrical forms and define the incoming and outgoing flow of people inside each region with respect to their (left, right, up, and bottom) sides.

To do so, an online aggregation of the spatiotemporal traces from a set of contributing users is proposed. Unlike previous offline proposals, the introduced mechanism allows discovering the target regions at the same time the trajectories are being received. Once a stable set of transit regions have been discovered, they are continuously monitored by a subset of suitable contributors who are dynamically selected.

The goal of such monitoring is to detect sudden or long-standing meaningful changes of human movement within the detected regions like people driving slower than usual or walking in unusual directions. These mobility shifts can be signs of events of interest, like unplanned demonstrations or serious traffic problems, whose early perception is of great help for many public and private stakeholders.

All in all, bearing in mind the open challenges of mobility mining based on MCS listed before, the salient contributions of the present work are the following.

(i) A novel mechanism uses only a subset of contributors to monitor the state of the composed transit areas: since the rest of users are deactivated, it reduces the extra cost of taking part of this type of solution.

(ii) A new solution explicitly detects changes in the movement of people within the target areas.

(iii) As far as privacy is concerned, the model of the detected areas only exposes general mobility information without disclosing any personal details of the contributors. This allows sharing the detected areas with third-party services without suffering serious privacy leaks.

Finally, the remainder of the paper is structured as follows. To start with, Section 2 is devoted to describing in detail the concept of dense transit area and the logic structure of the proposed system. Section 3 puts forward the procedure to discover such transit areas. Next, how these areas are monitored is stated in Section 4. Then, Section 5 discusses the main results of the experiments. An overview about mobility mining is put forward in Section 6. Finally, the main conclusions and the future work are summed up in Section 7.

2. Dense Transit Areas Detection System

This section is devoted to explaining the goal of the proposal along with the architecture. For the sake of clarity, Abbreviations summarizes the key acronyms and symbols used in the following sections.

2.1. Dense Transit Area Definition. The main goal of the system is to detect the spatial areas within a city where a high density of human transit exists. In our setting, such human transit is defined as the routes that people follow to move from one place to another (e.g., home, work, and school).

Definition 1. A city's region of influence is the spatial region comprising all the frequent origins $\mathcal{O}$ and destinations $\mathcal{D}$ of its citizens.

Definition 2. A route of a person p, $r(p)$, is the continuous movement in a city's region of influence from an origin $o_r \in \mathcal{O}$ to a destination $d_r \in \mathcal{D}$.

Definition 3. The lifetime of a route $r(p)$, $t(r(p))$, is the time interval $[t_o, t_d]$ between the instant at which p departed from $\mathcal{O}_r(t_o)$ and the arrival time at $\mathcal{D}_r(t_d)$.

Definition 4. A subroute of a route $r(p)$, $r(p)^s$, is the part of the continuous movement of r during a time interval $[t_i, t_f] \subseteq t(r(p))$.

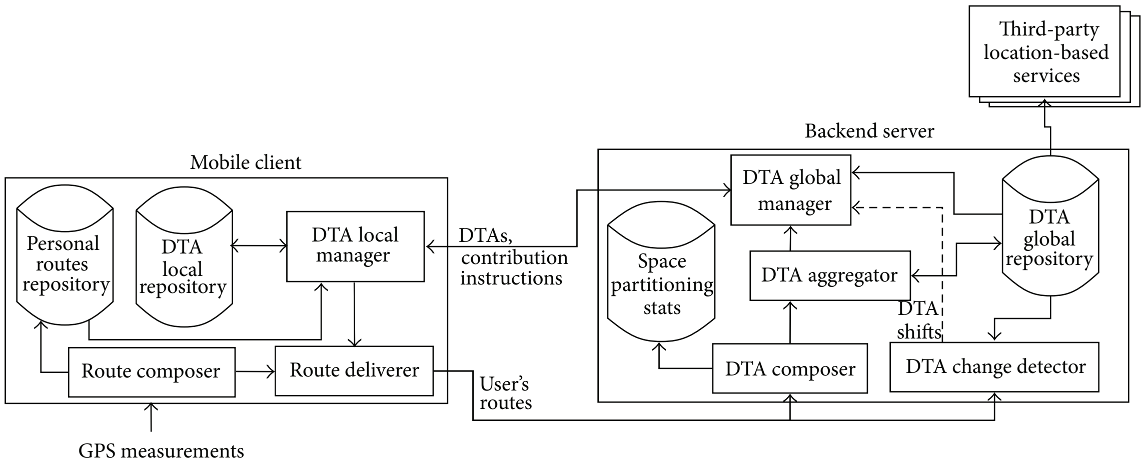


Figure 2: System architecture.

Bearing in mind the aforementioned concepts of human movement, we can then come up with a transit area definition.

Definition 5. A dense transit area (DTA) is a spatial region that has been visited by a set of subroutes $\mathscr{R}^s$, $|\mathscr{R}^s| \gg 1$, from a set of people $\mathscr{P}$, $|\mathscr{P}| \gg 1$.

Consequently, a DTA represents a spatial region of a city that is visited by a large number of citizens' routes (e.g., ring roads, central avenues, or parks). For example, Figure 1 shows two DTAs, each one comprising 3 different subroutes. Note that these routes may have any purpose like commuting, going to the school, or shopping. As a result, the set of DTAs of a city comprise the areas that capture most of the movement of its population. The following sections describe how such DTAs can be perceived.

2.2. System Architecture. As it has been previously stated, the system follows an opportunistic MCS approach so as to detect the DTAs. Therefore, it relies on a set of participants or contributors that voluntarily accept to undertake sensing tasks.

From an architectural point of view, Figure 2 depicts that the system comprises two different elements, a thin client running in the personal or vehicle-mounted devices of the users and a back-end server.

The mobile client is in charge of detecting, at each moment, the routes of the device's holder and sending them to the central server. Due to the adopted opportunistic sensing, this task is carried out in unconscious mode; namely, the client runs in the background and opportunistically collects and delivers the routes without active involvement of user.

Next, the server, on the basis of the collected routes, composes and manages the DTAs. It is important to note that such DTA generation is undertaken in an incremental manner at the same time users cover their routes, so the system does not rely on any type of previously gathered data.

2.3. System Operation Modes. The present system supports two different modes of execution, *DTA discovery* and *DTA monitoring*. Depending on the active mode, the system focuses on a particular task and it changes from one mode to the other when certain conditions arise.

(i) Firstly, the *DTA discovery* phase is the initial mode of the system. During this state, the system focuses on generating a set of DTAs, $\mathscr{A}$, as complete and detailed as possible. To do so, the system collects all the routes from all the participants and timely composes $\mathscr{A}$ on the basis of these routes.

(ii) Once the system has been able to generate a suitable set $\mathscr{A}$ then it transitions to the *DTA monitoring* mode. In this second mode, the system focuses on controlling the evolution of certain mobility features of the DTAs in $\mathscr{A}$ to early detect potential mobility shifts. To do so, the system processes the routes of a subset of participants that allow reliably perceiving the aforementioned features. Finally, in case the system actually detects a potential shift, it moves to the initial *DTA discovery* mode in order to fully capture the mobility change in $\mathscr{A}$.

By means of these two modes, the system is able not only to detect the DTAs but also to perceive the movement inside them. Besides, for the monitoring task the system only uses part of the contributors so it does not require all the participants to report their locations all the time. For the sake of clarity, Figure 3 shows the state machine of the system. We will see the inner functionality of the system with respect to both modes in the upcoming sections.

3. DTA Discovery

During the *DTA discovery mode*, the system's components of both the thin client and the back-end server are intensively executed so as to generate an initial or refined set of DTAs $\mathscr{A}$. In more detail, the steps followed by the system are put forward next.

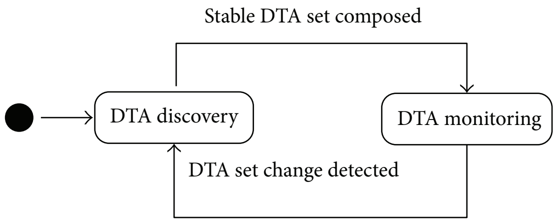


Figure 3: System's state machine where the *DTA discovery* is the initial state of the system.

3.1. Users' Routes Generation. In order to generate the routes covered by the user, both the *route composer* and the *route deliverer* components work in a collaborative manner.

3.1.1. Route Composer. As Figure 2 shows, this element is part of the mobile client running in each user's mobile device. Its key goal is to detect the current route $r(p)$ that the device's holder p is covering at each moment.

For this goal, this module only relies on the device's GPS sensor to extract the routes' raw locations. In particular, the sensor periodically provides the module with a new timestamped location l comprising the tuple $\langle x, y, t\rangle$ where $\langle x, y\rangle$ is the location in terms of latitude-longitude coordinates at instant t.

On the basis of the collected locations, the system incrementally composes the sequence of timestamped locations, $r(p)_l = \{l_1, l_2, \ldots, l_n\}$, of the ongoing route $r(p)$. This is done by a two-step procedure.

Firstly, the algorithm removes erroneous or irrelevant locations that the GPS sensor may return [7]. This is done by means of the distance-based filtering applied to each new location l_{new} described in [8] that allows performing this cleaning in real time.

Secondly, if l_{new} is not discarded by the aforementioned filter then it is appended to $r(p)_l$ by following a spatiotemporal gap identification approach. To do so, a maximum distance, $\mathcal{S}_{\text{gap}}$, and time interval, $\mathcal{T}_{\text{gap}}$, between two consecutive locations are defined. If the spatial or temporal distance between l_{new} and the last location in $r(p)_l$, l_{last}, exceeds $\mathcal{S}_{\text{gap}}$ or $\mathcal{T}_{\text{gap}}$, it identifies l_{last} as ending point d_r of the ongoing route $r(p)$ and l_{new} as the starting point o_r of a new $r(p)$. Otherwise, l_{new} is appended to $r(p)_l$ as the new last location.

Finally, the current sequence of the ongoing route, $r(p)_l$ (and the one of the just-completed route, $r(p)_l^{\text{ended}}$, if any), is sent to the *route deliverer* component.

3.1.2. Route Deliverer. This element is in charge of controlling the routes sequences that are delivered to the central server from the mobile client.

When the system runs in the *DTA discovery* mode, this component only processes each completed route sequence $r(p)_l^{\text{ended}}$. In particular, it carries out two different tasks: (1) it delivers the route to the central server and (2) it stores such route in the local *personal routes repository* within the mobile client (see Figure 2). Such repository keeps the last routes covered by the user. As we will see later, this repository is instrumental when the system runs in the *DTA monitoring* mode.

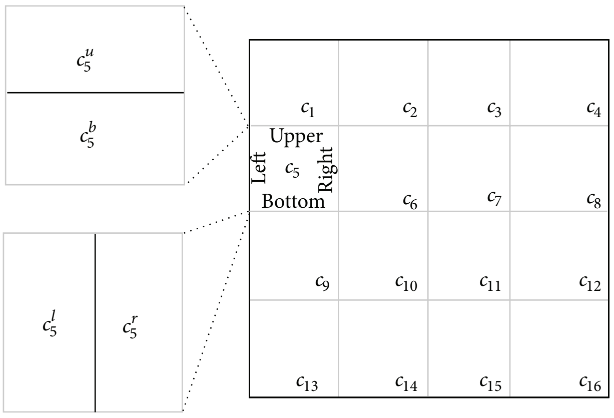


Figure 4: Graphical representation of the set $\mathscr{C}(c_5)^{\text{sub}}$.

3.2. DTA Generation. Once the server receives the users' routes, it makes up the DTAs by means of two of its modules, the *DTA composer* and the *DTA aggregator*.

3.2.1. DTA Composer. This module of the back-end server is responsible for actually detecting the new DTAs of the city. Hence, it receives all the completed routes from all the users when the system runs in *DTA discovery*.

Bearing in mind Definition 5, we can regard a DTA as a spatial region exhibiting a high density of routes from many different users. Consequently, this module adopts an approach based on computing certain features of the incoming routes with respect to a predefined spatial partition so as to calculate the density of routes in each part of the city and, thus, uncover its DTAs.

To do so, *DTA composer* firstly divides the whole spatial region of the city under study into squared cells of the same size, $\mathscr{C}$. In turn, each cell $c_i \in \mathscr{C}$ is further divided into four *subcells* each one covering a different spatial region inside c_i; namely, $\mathscr{C}(c_i)^{\text{sub}} = \{c_i^u, c_i^b, c_i^l, c_i^r\}$. As Figure 4 shows, these subcells split the cell regarding the different manners a route can traverse it.

On the basis of this multilevel spatial partition, the module calculates the route density in each cell and subcell by the procedure shown in Algorithm 1. This algorithm is launched whenever a new route $r(p)_l^{\text{ended}}$ from any user is received.

First of all, the algorithm gets the timestamp at which the incoming route started (line 2 of Algorithm 1). Next, it maps the sequence of timestamped locations of the incoming route to the multilayered spatial partition described above. As a result, the route is translated into a sequence of cells, $r(p)_c^{\text{ended}}$ (line 3). As Table 1 shows, each cell $c \in r(p)_c^{\text{ended}}$ comprises five movement attributes related to $r(p)_l^{\text{ended}}$.

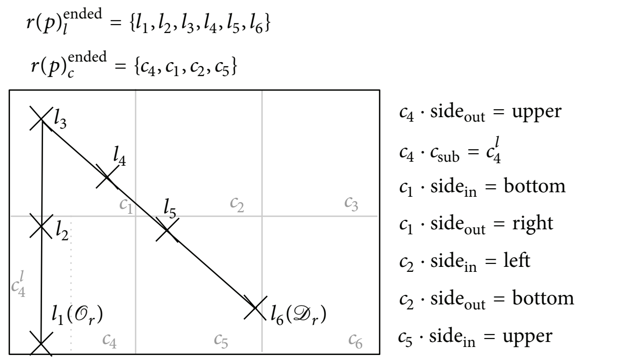
These five attributes can be computed using simple mathematics and computational geometry [9]. In that sense, Figure 5 shows an example of how a route comprising six

```
Input: A completed route's abstraction $r(p)_l^{\text{ended}}$
Output: Set of new generated DTAs $\mathscr{A}_{\text{new}}$
(1) $\mathscr{A}_{\text{new}} \leftarrow \emptyset$
(2) $t \leftarrow r(p)_l^{\text{ended}}.t_{\text{init}}$
(3) $r(p)_c^{\text{ended}} \leftarrow map(\mathscr{C}, r(p)_l^{\text{ended}})$
(4) for each $c \in r(p)_c^{\text{ended}}$ do
(5)      if $c.speed \le speed_{walking}$ then
(6)           $c_{\text{tr}} \leftarrow \mathscr{C}_{\text{tr}}^{w}.get(c)$
(7)      else
(8)           $c_{\text{tr}} \leftarrow \mathscr{C}_{\text{tr}}^{\text{nw}}.get(c)$
(9)      if $update\_cell(c_{\text{tr}}, c, t) = true$ then
(10)          $dta_{\text{new}} \leftarrow$ new DTA($c_{\text{tr}}$)
(11)          $\mathscr{A}_{\text{new}} \leftarrow \mathscr{A}_{\text{new}} \cup dta_{\text{new}}$
(12)     else if $c.c_{\text{sub}} \ne \emptyset$ then
(13)          $c_{\text{tr}}^{\text{sub}} \leftarrow c_{\text{tr}}.get\_subcell(c.c_{\text{sub}})$
(14)          if $update\_cell(c_{\text{tr}}^{\text{sub}}, c.c_{\text{sub}}, t) = true$ then
(15)               $dta_{\text{new}} \leftarrow$ new DTA($c_{\text{tr}}$)
(16)               $\mathscr{A}_{\text{new}} \leftarrow \mathscr{A}_{\text{new}} \cup dta_{\text{new}}$
(17) return $\mathscr{A}_{\text{new}}$
(18) function $update\_cell(c_{\text{tr}}, c, t)$
(19) $c_{\text{tr}}.users \leftarrow c_{\text{stats}}.users \cup p$
(20) $c_{\text{tr}}.n\_routes$ ++
(21) $c_{\text{tr}}.get\_time(t).get\_side(c.side_{\text{in}}).routes_{\text{in}}$ ++
(22) $c_{\text{tr}}.get\_time(t).get\_side(c.side_{\text{out}}).routes_{\text{out}}$ ++
(23) $c_{\text{tr}}.get\_time(t).speed \leftarrow inc\_avg(c_{\text{stats}}.n\_routes, c_{\text{stats}}.get\_hour(h).speed, c.speed)$
     $c_{\text{tr}}.route\_len \leftarrow c_{\text{tr}}.route\_len + c.route\_len$
(24) $a \leftarrow c.side\_length \times c.side\_length$
(25) if $c_{\text{tr}}.route\_len/a \ge \delta_{\min} \wedge |c_{\text{tr}}.users| \ge \mu_{\min}$ then
(26)      return true
(27) return false
```

Algorithm 1: DTA detection algorithm.

Table 1: Attributes for each cell c in a cell-based route sequence $r(p)_c^{\text{ended}}$.

Attribute	Meaning
side_length	Length of each side of the cell. This value is the same for all $c \in \mathscr{C}$
speed	Average speed of $r(p)_l^{\text{end}}$ in c
route_len	Spatial length of $r(p)_l^{\text{end}}$ in c
$side_{\text{in}}$	Side of the cell at which the route got into it (*upper, bottom, right,* or *left*)
$side_{\text{out}}$	Outgoing side of the route for c
c_{sub}	Subcell of $\mathscr{C}(c)^{\text{sub}}$ fully covered by $r(p)_{\text{abs}}^{\text{ended}}$ in c



$r(p)_l^{\text{ended}} = \{l_1, l_2, l_3, l_4, l_5, l_6\}$
$r(p)_c^{\text{ended}} = \{c_4, c_1, c_2, c_5\}$
l_3
l_4
c_1
l_5
c_2
c_3
l_2
c_4^l
$l_1(\mathcal{O}_r)$
$l_6(\mathcal{D}_r)$
c_4
c_5
c_6
$c_4 \cdot \text{side}_{\text{out}} = \text{upper}$
$c_4 \cdot c_{\text{sub}} = c_4^l$
$c_1 \cdot \text{side}_{\text{in}} = \text{bottom}$
$c_1 \cdot \text{side}_{\text{out}} = \text{right}$
$c_2 \cdot \text{side}_{\text{in}} = \text{left}$
$c_2 \cdot \text{side}_{\text{out}} = \text{bottom}$
$c_5 \cdot \text{side}_{\text{in}} = \text{upper}$

Figure 5: Example mapping of a route and the attributes of each of its cells.

timestamped locations is mapped to a sequence of four different cells $r(p)_c^{\text{ended}} = \{c_4, c_1, c_2, c_5\}$ along with some attributes of each cell. In that sense, we can see that not all the cells include the c_{sub} attribute (see Table 1). This is because, in many cases, the subroute in a cell covers more than one of the subcells. For example, in Figure 5 the subroute of $r(p)_l$ in cell c_4 is fully covered by the subcell c_4^l whereas in cell c_1 the subroute is partially covered by the 4 subcells.

Once the mapping is completed, the algorithm uses the resulting sequence $r(p)_c^{\text{ended}}$ to update the *space partitioning stats* repository (see Figure 2).

This repository stores aggregated transit data of the space partition $\mathscr{C}$. In particular, such data is organized in two entities, $\mathscr{C}_{\text{tr}}^{w}$ and $\mathscr{C}_{\text{tr}}^{\text{nw}}$. The former stores information about routes having a low speed so that they are likely to have been covered walking whereas the latter stores information about nonwalking routes as they exhibit higher speed representing vehicle-based routes. Both $\mathscr{C}_{\mathrm{tr}}^{w}$ and $\mathscr{C}_{\mathrm{tr}}^{\mathrm{nw}}$ store for each $c \in \mathscr{C}$ the transit properties shown in Table 2.

The rationale of having two separate instances is that urban dynamics might be different depending on whether we consider movement on foot or in vehicles. For example, vehicle-based routes are constrained by the road network of the city whereas walking-based routes usually do not show so constrained displacements.

Consequently, for each cell c in $r(p)_c^{\mathrm{ended}}$ the system updates $\mathscr{C}_{\mathrm{stats}}^{w}$ or $\mathscr{C}_{\mathrm{stats}}^{\mathrm{nw}}$ depending on its speed attribute and a domain-dependant threshold $speed_{walking}$ (lines 5–8 of Algorithm 1). This update process is carried out by the *update_cell* function (line 9). This function takes a cell from $r(p)_c^{\mathrm{ended}}$, c, and its associated element in the repository c_{tr}. As a result, it returns a Boolean value indicating whether the historical transit data in c_{tr} allows classifying it as a DTA. If that is the case, such entity gives rise to a new DTA (line 10) that is eventually added to the set of new DTAs (line 11). In this generation, the system removes the *users* attribute to keep DTAs anonymized. Otherwise, the system repeats the same process but this time with the subcell of c, c_{sub} (see Table 1) (lines 12–16 in Algorithm 1).

This way, the algorithm follows a top-down approach so as to generate the DTAs, as it firstly tries to generate cell-based DTAs. If that is not possible, it focuses on detecting smaller DTAs with subcell granularity. Finally, the algorithm returns the set of new DTAs generated on the basis of the incoming route (line 18).

Concerning the inner functionality of the *update_cell* function (lines 18–27), it firstly updates c_{tr} with the attributes of its associated cell c by simple or incremental addition (lines 19–23). In that sense, attributes speed, routes$_{\mathrm{in}}$, and routes$_{\mathrm{out}}$ are disaggregated by a temporal criterion $\mathscr{T}_{\mathrm{crit}}$. This way, it is possible to know the speed and information about incoming and outgoing sides in a particular time interval $t \in \mathscr{T}_{\mathrm{crit}}$ with predefined granularity. As a matter of fact, if an hour granularity is chosen then $\mathscr{T}_{\mathrm{crit}}$ is defined as an array $\mathscr{T}_{\mathrm{crit}} = \{0, 1, \ldots, 23\}$.

Once c_{tr} has been updated, *update_cell* also detects whether it actually can give rise to a DTA (lines 25-26). For that goal, the function checks two features of c_{tr}, (1) its density of routes and (2) the number of users that have visited at least once the cell.

The first one can be calculated with respect to the total length of historical subroutes within c_{tr} and its geometric area. Since we are using square cells, we can easily compute such area as the cell's side length squared (line 24). At the end, if such density and number of users are over two domain-dependant thresholds, $\delta_{\min}$ and $\mu_{\min}$, the function concludes that c_{tr} actually represents a DTA, thus returning the Boolean value *true*. Otherwise, *false* is returned.

3.2.2. DTA Aggregator. The resulting set, $\mathscr{A}_{\mathrm{new}}$, from the previous algorithm is directly delivered to this module (see Figure 2). In that sense, $\mathscr{A}_{\mathrm{new}}$ comprises DTAs at a cell or subcell granularity. However, real DTAs can cover spatial areas larger than the predefined size of a cell. For that reason, the *DTA aggregator* element applies a fusion procedure to the resulting DTAs to merge such areas.

The key idea of this procedure is that two closed DTAs can be merged together, creating a larger DTA, if the transit information they represent is strongly related. In our setting, we infer that two DTAs represent the same transit flow if people move at a similar speed and direction in both areas. More specifically, we distinguish between two types of similarities among DTAs, namely, the following:

(i) Parallel-transit similarity, sim$_{\mathrm{par}}$: this similarity arises when the subroutes of the DTAs have a very similar direction and speed.

(ii) Common-transit similarity, sim$_{\mathrm{com}}$: this similarity occurs when the DTAs have a certain number of common subroutes covering them.

For example, the two DTAs in Figure 6(a) comprise quite different transit flows in terms of both speed (one of them is mostly covered by vehicle-based routes whereas the other one has more walking routes) and direction (the routes in each DTA go in reverse with respect to the other). However, Figure 6(b) shows a parallel-transit similarity between the two DTAs whereas Figure 6(c) depicts a common-transit similarity. Therefore, the DTAs in these two last situations could be merged to make up a new and larger DTA.

In order to compute each similarity we made use of the number of incoming and outgoing subroutes that each DTA comprises (see Table 2). Thus, given two DTAs a and b their parallel similarity $\mathrm{sim}_{\mathrm{par}}(a, b) \in [0, 1]$ is calculated as follows:

$$\mathrm{dissim}_{\mathrm{par}}^{\mathrm{in}}(a, b) = \frac{\sum_s \left| a.\mathrm{routes}_{\mathrm{in}}^{s} / a.n\_\mathrm{routes} - b.\mathrm{routes}_{\mathrm{in}}^{s} / b.n\_\mathrm{routes} \right|}{4}, \quad (1)$$

$$\mathrm{dissim}_{\mathrm{par}}^{\mathrm{out}}(a, b) = \frac{\sum_s \left| a.\mathrm{routes}_{\mathrm{out}}^{s} / a.n\_\mathrm{routes} - b.\mathrm{routes}_{\mathrm{out}}^{s} / b.n\_\mathrm{routes} \right|}{4}, \quad (2)$$

$$\mathrm{sim}_{\mathrm{par}}(a, b) = 1 - \left(\frac{\mathrm{dissim}_{\mathrm{par}}^{\mathrm{in}}(a, b) + \mathrm{dissim}_{\mathrm{par}}^{\mathrm{out}}(a, b)}{2} \right). \quad (3)$$

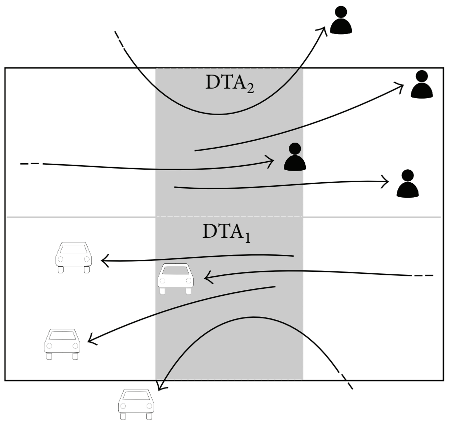
Equations (1) and (2) calculate the dissimilarity between the two rates of incoming and outgoing subroutes for each of the four sides of a DTA. Finally, (3) aggregates both rate differences to generate the final parallel similarity.

Furthermore, the common-transit similarity $\mathrm{sim}_{\mathrm{com}}(a, b) \in [0, 1]$ is calculated as follows:

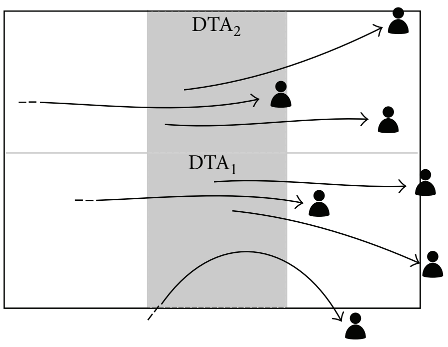
$$\mathrm{sim}_{\mathrm{com}}(a, b) = 1 - \left(\frac{\left| a.\mathrm{routes}_{\mathrm{in}}^{s} - b.\mathrm{routes}_{\mathrm{out}}^{s'} \right|}{2} + \frac{\left| a.\mathrm{routes}_{\mathrm{out}}^{s} - b.\mathrm{routes}_{\mathrm{in}}^{s'} \right|}{2} \right), \quad (4)$$

Table 2: Attributes for each cell c_{tr} in $\mathscr{C}^{w}_{\text{tr}}$ and $\mathscr{C}^{\text{nw}}_{\text{tr}}$ and DTA in $\mathscr{A}$ (except *users*).

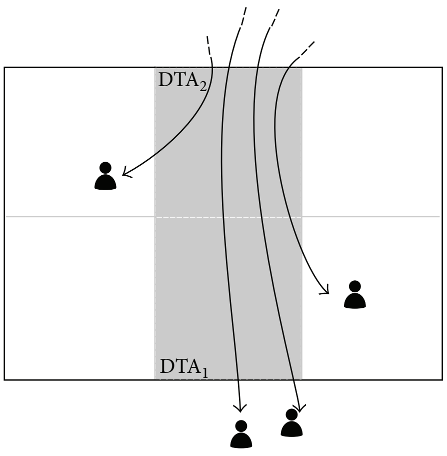
Attribute	Meaning
type	Type of movement within the cell, namely, *walking* or *nonwalking*
users	Number of users who have at least one route covering the cell c
n_routes	Number of different historical subroutes that have traversed c
route_len	Total length of all the subroutes that have covered c
speed	Average speed of the routes while traversing c
$routes^{s}_{\text{in}}$	Number of subroutes that have gone into c through a particular side s of the cell ($s \in \{upper\|bottom\|left\|right\}$). For example, $routes^{\text{left}}_{\text{in}}$ informs about how many routes have covered c by getting into its left side
$routes^{s}_{\text{out}}$	Number of subroutes that have left c through each side of the cell



(a) Nonrelated DTAs



(b) DTAs with high parallel-transit similarity



(c) DTAs with high common-transit similarity

Figure 6: Examples of DTAs similarities. Each DTA is shown as a grey square.

where s is the common side between a and b from a perspective whereas s' is the adjacent side from b point of view. For example, in Figure 6(a) s = bottom (for DTA$_2$) whereas s' = upper for (DTA$_1$). As we can see, sim$_{\text{com}}$ basically measures the subroutes that actually move between the two DTAs under consideration.

All in all, Algorithm 2 shows the mechanism applied by the *DTA composer* so as to fuse DTAs. Basically, this algorithm takes a set of DTAs to be merged, $\mathscr{A}_{\text{target}}$. Then, each pair of adjacent DTAs is compared to measure its parallel (lines 4–7) and common (lines 9–12) similarity. In that sense, only DTAs with the same *type* attribute (see Table 2) are compared. This way, we ensure that both contain similar routes in terms of speed. It is also important to note that both similarities are calculated with respect to the time criterion $\mathscr{T}_{\text{crit}}$. Thus, if any of these similarities is above its associated threshold sim$_{\text{min}}$, it means that both DTAs comprise a similar or common human dynamics for different time periods. As a result, the two DTAs can be merged.

```
Input: Set of DTAs $\mathscr{A}_{\text{target}}$
Output: Set of merged DTAs $\mathscr{A}_{\text{target}}^{\text{merged}}$
(1) $\mathscr{A}_{\text{target}}^{\text{merged}} \leftarrow \emptyset$
(2) for each $a \in A_{\text{target}}$ do
(3)     $\mathscr{A}_{\text{adj}}^{a} \leftarrow find\_adj\_DTAs(A_{\text{new}}, a)$
(4)     for each $a_{\text{adj}} \in A_{\text{adj}}^{a}$ do
(5)         if $\sum_{t\in\mathscr{T}_{\text{crit}}} \text{sim}_{\text{par}}(a, a_{\text{adj}}, t)/|\mathscr{T}_{\text{crit}}| \geq \text{sim}_{\text{min}}$ then
(6)             $a \leftarrow mergeDTA(a, a_{\text{adj}})$
(7)             $A_{\text{target}} \leftarrow A_{\text{target}} - a_{\text{adj}}$
(8)     $\mathscr{A}_{adj}^{a} \leftarrow find\_adj\_DTAs(A_{\text{new}}, a)$
(9)     for each $a_{\text{adj}} \in A_{\text{adj}}^{a}$ do
(10)        if $\sum_{t\in\mathscr{T}_{\text{crit}}} \text{sim}_{\text{com}}(a, a_{\text{adj}}, t)/|\mathscr{T}_{\text{crit}}| \geq \text{sim}_{\text{min}}$ then
(11)            $a \leftarrow mergeDTA(a, a_{\text{adj}})$
(12)            $A_{\text{target}} \leftarrow A_{\text{target}} - a_{\text{adj}}$
(13)    $\mathscr{A}_{\text{target}}^{\text{merged}} \leftarrow \mathscr{A}_{\text{target}}^{\text{merged}} \cup a$
(14) return $\mathscr{A}_{\text{target}}^{\text{merged}}$
```

Algorithm 2: DTA fusion algorithm.

Finally, Algorithm 2 is executed by the *DTA aggregator* by two different manners. In the first one, the algorithm is automatically launched when a new set of DTAs is delivered from the *DTA composer*. Then, the resulting set of fused DTA, $\mathscr{A}_{\text{target}}^{\text{merged}}$, is appended to the global set of DTAs, $\mathscr{A}$, in the *DTA global repository* (see Figure 2). Additionally, since this first type of execution only fuses the DTAs generated due to a single route, the fusion mechanism is also periodically launched over the DTA set, $\mathscr{A}$, so as to detect correlated DTAs in the whole city under study.

3.3. DTA Discovery-DTA Monitoring Transition.

3.3. DTA Discovery-DTA Monitoring Transition. At the same time the server composes the DTAs it controls their state so as to decide whether the system remains in the *DTA discovery* mode or it can move to the *DTA monitoring* phase.

In that sense, the system should transition from the discovery to the monitoring stage if a stable set of DTAs has been composed. In that sense, the *DTA global manager* module implements this decision process.

3.3.1. DTA Global Manager. This module defines a sampling time period $\mathscr{T}_s$ and for each period calculates the number of new DTAs n_{dta} and the number of received routes from all the users n_{routes}. If the ratio $n_{\text{dta}}/n_{\text{routes}}$ is below a decision threshold dt_{min} then it means that the system has not composed many new DTAs with respect to the incoming routes. Consequently, the module infers that the system has reached a consistent set of DTAs and eventually transitions to the *DTA monitoring* as it is depicted in Figure 3.

During this transition, the *DTA global manager* distributes the set $\mathscr{A}$ of DTAs uncovered by the server among all the contributors. In the client side, the *DTA Local Manager* receives such set and stores it in the *DTA local repository* (see Figure 2). As we will see in the next section, locally storing this data is of great help in the *DTA monitoring* mode.

On the whole, the system during the *DTA discovery* mode composes a set of DTAs by means of an approach that combines a multilayered partition of the space and its posterior fusion to come up with a reliable set of DTAs. Next, during *DTA monitoring* stage the system focuses on controlling the fact that the current set of DTAs actually represent the dynamics of the city under study.

4. DTA Monitoring

In the *DTA monitoring* mode, the system focuses on selecting a subset of users and uses their reported routes to compare the current state of the human transit in the city with respect to the one represented by the DTAs to see whether any discrepancy exists. This allows deactivating certain users and, thus, avoiding their continuous contribution to the system with the consequent resources saving.

As a result, the behaviour of the system changes with respect to the one described in the previous section so as to adapt to this new goal. In particular, the steps of the system in this mode are described next.

4.1. User Subset Selection. The first task the system does when it starts to operate in monitoring mode is to select the target set of monitoring users. This selective process is undertaken by the *DTA global manager* component.

4.1.1. DTA Global Manager. For the aforementioned goal, this module uses the sampling time period t_s and selects, for each period, the subset of users providing the best coverage of the detected DTAs. This selection process follows the following steps.

(1) For each DTA in $\mathscr{A}$, the module asks the mobile clients to report their *availability* to visit this DTA within the current sampling period.

(2) Among all the users reporting an availability score over a domain-dependant threshold av_{min}, it chooses the top k users according to this value.

(3) These top k users are committed to report their ongoing routes to the server during the current sampling interval.

4.1.2. DTA Local Manager. This module in the mobile client is in charge of calculating the *availability* score used by the aforementioned selection process.

To do so, each time the mobile clients switch to the *DTA monitoring* mode, this module reads the historical routes of the user stored in the *personal routes repository*. Then, it counts the number of visits to each DTA stored in its *DTA local repository*, for each time period in $\mathcal{T}_{\text{crit}}$. Then, it removes all these historical routes from the repository. This way, only the routes covered by the user during the previous *DTA discovery* cycle are used to generate the visit statistics. This avoids the usage of rather deprecated routes.

With this information, the module can easily calculate probability of visiting a DTA, *dta,* at particular time interval $t \in \mathcal{T}_{\text{crit}}$ as

$$p(dta)_t = \frac{nv(dta)_t}{nr_t}, \qquad (5)$$

where $nv(dta)_t$ is the number of visits to dta during t and nr_t is the number of routes that started during t. On the basis of such probability the module calculates the availability to visit a DTA at a particular time interval t, $\text{av}(dta)_t \in [0,1]$, as follows:

$$\text{av}(dta)_t = \text{contribution}_{\text{factor}} \times p(dta)_t, \qquad (6)$$

where $\text{contribution}_{\text{factor}}$ is a corrective factor to avoid the fact that a user contributes to the monitoring state during too many consecutive time intervals. To do so, the module counts the number of times the user has been selected by the server during the last $nc_{\max}$ instants, nc_m, and computes such factor as follows:

$$\text{contribution}_{\text{factor}} = 1 - \frac{nc_m}{nc_{\max}}. \qquad (7)$$

Finally, $\text{av}(dta)_t$ for each DTA is locally stored and sent to the central server in each selection process.

4.2. User Subset Routes Generation. As in the previous mode, this task is performed by the same two modules in the client side, *route composer* and *route deliverer* (see Figure 2).

4.2.1. Route Composer. This component in the client has the same functionality in both operational modes. Hence, it just composes the sequences of timestamped locations of the user's routes and sends them to the *route deliverer* as it has been described in Section 3.1.

4.2.2. Route Deliverer. The behaviour of this element changes slightly under the monitoring stage. In particular, it now forwards to the central server the ongoing route's sequence $r(p)_l$ instead of the completed routes. Therefore, each time the *route composer* appends a new location to such sequence, this module immediately delivers the new version to the server. This way, the central server is informed of how the users move in real time under this operation mode.

4.3. DTA Change Detection. The key tasks in this operation mode are to perceive potential changes of the urban dynamics in the target city. This process involves the *DTA change detector* component.

4.3.1. DTA Change Detector. This module processes all the ongoing routes reported by the subset of users so as to extract the current movement features inside each DTA in terms of direction and speed. Then, it periodically compares such current values with the historical ones stored in the *DTA global repository*. Algorithm 3 shows this process in more detail.

In short, the algorithm firstly maps each route to its cell-based representation (line 3). After that, it updates the current state of each DTA covered by the route's cells (lines 5–9).

Next, the similarity between the current and the historical state of each DTA in terms of transit direction and speed is measured (lines 11–13). In that sense, we use the parallel similarity for the directional features (see Section 3.2.2). This allows detecting whether the subroutes currently visiting the DTA enter and exit it in the same way and speed that is reflected in the historical data. If that is not the case, it means that people or vehicles are moving in an unusual direction or speed. This might be originated due to planned events (e.g., road works) or unplanned ones (e.g., sudden riots or car accidents).

Consequently, if either the speed or direction similarity is below a predefined threshold then the algorithm infers that the human dynamics inside the DTA under review might have moved with respect to its default state. As a result, it is appended to the list of DTAs with potentially changed DTAs, $\mathcal{A}_{\text{shift}}$ (line 16).

4.3.2. Low DTA Coverage Problem. It might occur that the selection process described in Section 4.1 is not capable of configuring a suitable set of top-k users for one or more DTAs. In order to cope with this limitation, the system indirectly calculates the potential dissimilarity of these uncovered DTAs by using the K-nearest neighbours (KNN) method.

In more detail, for each DTA with low coverage, dta_{lc}, the *DTA change detector* gets the speed and direction similarity, sim_{dir}, $\text{sim}_{\text{speed}}$ of its k closest DTAs with suitable coverage, $\mathcal{A}(\text{dta})_k$. Then, it extrapolates such measurements to infer the associated similarities of dta_{lc} as follows:

$$\begin{aligned} \text{sim}_{\text{dir}} &= \frac{\sum_{i\in|k|} \lambda^i_{\text{dist}} \times \text{sim}^i_{\text{dir}}}{\sum_{i\in|k|} \lambda^i_{\text{dist}}}, \\ \text{sim}_{\text{speed}} &= \frac{\sum_{i\in|k|} \lambda^i_{\text{dist}} \times \text{sim}^i_{\text{speed}}}{\sum_{i\in|k|} \lambda^i_{\text{dist}}}, \end{aligned} \qquad (8)$$

where λ^i_{dist} is a decay factor proportional to the distance between the ith DTA in $\mathcal{A}(\text{dta})_k$ and dta_{lc} whereas $\text{sim}^i_{\text{dir}}$ and $\text{sim}^i_{\text{dir}}$ are the direction and speed similarities of the ith DTA in $\mathcal{A}(\text{dta})_k$.

Lastly, these two inferred values are used like the ones calculated by Algorithm 3 for the anomaly detection in DTAs described above.

```
Input: Ongoing route $r(p)_l$, DTAs current features
       $\mathscr{A}_{\text{curr}}$, DTA historic features $\mathscr{A}$
Output: Set of DTAs with meaningful shifts $\mathscr{A}_{\text{shift}}$
(1)  $\mathscr{A}_{\text{shift}} \leftarrow \emptyset$
(2)  $t \leftarrow r(p)_c.t_{\text{init}}$
(3)  $r(p)_c \leftarrow map(\mathscr{C}, r(p)_{\text{seq}})$
(4)  for each $c \in r(p)_c$ do
(5)      $dta_{\text{curr}} \leftarrow \mathscr{A}_{\text{curr}}.get(c)$
(6)      $dta_{\text{curr}}.n\_routes$ ++
(7)      $dta_{\text{curr}}.get\_time(t).get\_side(c.side_{\text{in}}).routes_{\text{in}}$ ++
(8)      $dta_{\text{curr}}.get\_time(t).get\_side(c.side_{\text{out}}).routes_{\text{out}}$ ++
(9)      $dta_{\text{curr}}.get\_time(t).speed \leftarrow inc\_avg(dta_{\text{curr}}.n\_routes,$
(10)                                  $dta_{\text{curr}}.get\_hour(h).speedc.speed)$
(11)     $dta_{\text{hist}} \leftarrow \mathscr{A}.get(c)$
(12)     $\text{sim}_{\text{dir}} \leftarrow \sum_{t \in \mathscr{T}_{\text{crit}}} \text{sim}_{\text{par}}(dta_{\text{hist}}, dta_{\text{curr}}, t)/|\mathscr{T}_{\text{crit}}|$
(13)     if $\text{sim}_{\text{dir}} \leq \text{sim}_{\text{min}}$ then
(14)         $\mathscr{A}_{\text{shift}} \leftarrow \mathscr{A}_{\text{shift}} \cup dta_{\text{hist}}$
(15)     $\text{sim}_{\text{speed}} \leftarrow |(dta_{\text{hist}}.speed - dta_{\text{curr}}.speed)/\max(dta_{\text{hist}}.speed, dta_{\text{curr}}.speed)|$
(16)     if $\text{sim}_{\text{speed}} \leq \text{sim}_{\text{min}}$ then
(17)         $\mathscr{A}_{\text{shift}} \leftarrow \mathscr{A}_{\text{shift}} \cup dta_{\text{hist}}$
(18) return $\mathscr{A}_{\text{shift}}$
```

Algorithm 3: DTA monitoring algorithm.

4.4. DTA Monitoring-DTA Discovery Transition. If a DTA is included in $\mathscr{A}_{\text{shift}}$ during t_{shift} consecutive sampling periods, the module concludes that the flow of people/vehicles within such an area has actually changed. As a result, it alerts the *DTA global manager* (see Figure 2).

When this module receives such an alert, it automatically starts the transition of the whole system to go back to the initial *DTA discovery* mode. The goal in this case is to use again all the contributors' routes so as to accurately confirm the preliminary change detected by the *DTA change detector*.

4.5. Exploitation of the DTA Global Repository. As Figure 2 depicts, the whole set of DTAs that the system generates and updates in its two-operation-mode cycle is stored in the *DTA global repository*.

Such a repository is exposed to third-party location-based services that can make use of it. In that sense, it is important to recall that the transit information that each DTA contains and, thus, is used by the stakeholders is completely anonymized. According to Table 2, each DTA informs of the average velocity of usual directions of routes visiting such area but it does not comprise any type of personal information of the users. Consequently, it prevents a malicious third-party service from getting access to personal details of the contributors by means of the public DTAs.

5. Evaluation

In order to state a comprehensive view of our proposal, we evaluated it on a real-world dataset. Besides, we compared our proposal with an existing approach to perceive abnormal movements in the mobility mining domain.

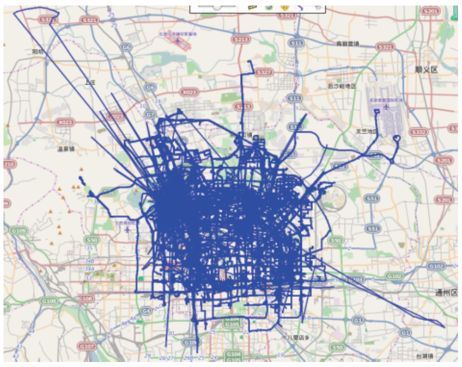


Figure 7: Digital trace of the GL dataset.

5.1. Experiment Setup

5.1.1. Dataset. In order to test our system we have used the *GeoLife dataset* (GL) [10], a public collection of human trajectories produced by 178 users carrying different GPS feeds in a period of over three years. Figure 7 depicts its trace that falls into the square of latitude 40.17 to 39.86 and longitude 116.14 to 116.34 covering a large area of Beijing city (China). More details about the dataset are provided in Table 3.

5.1.2. Settings. Table 4 summarizes the default configuration of the system for the evaluation.

Furthermore, a week-hour granularity has been chosen for $\mathscr{T}_{\text{crit}}$ so it takes the form of an array $\mathscr{T}_{\text{crit}} = \{0, 1, 2, \ldots, 167\}$ representing the 168 hours within a week. Consequently, the transit features of the DTAs are disaggregated with respect to such an array.

Table 3: GL dataset general information.

Users	Locations	Routes	Time period
178	23667828	17621	2007-04 → 2011-10

Table 4: System default configuration.

Parameter	Module	Value
$\mathcal{S}_{\text{gap}}$	*Route composer*	1000 m
$\mathcal{T}_{\text{gap}}$	*Route composer*	15 m
$speed_{walking}$	*DTA composer*	1.2 m/s
$\delta_{\min}$	*DTA composer*	2 m/m^2
$\mu_{\min}$	*DTA composer*	5
$\text{sim}_{\min}$	*DTA aggregator*	0.8
t_{shift}	*DTA change detector*	3
$av_{\min}$	*DTA global manager*	0.8
k	*DTA global manager*	10
$dt_{\min}$	*DTA global manager*	0.2
$nc_{\max}$	*DTA global manager*	4
t_s	*DTA global manager*	360 s

5.2. Effect of the Cell Size. When a grill-based partitioning is used to compose the representation of spatiotemporal routes, the cell size is one of the parameters that has an important impact on the whole system. In our case, this size also indicates the default size of the DTAs. As a result, Figure 8 shows, for different cell sizes, the number of DTAs composed by the system and the number of times the system detected a shift in these DTAs.

As we can see, setting a small cell size implies the generation of a large number of DTAs. This allows representing the transit information of the city with fine granularity. For example, Figure 8 shows that when a 100 m cell is used, the system generated 681 DTAs.

In certain scenarios, such large number of areas could not be convenient as stakeholders are more interested in perceiving the movement within a city in a more general way. In that sense, increasing the cell size involves, unsurprisingly, the generation of a reduced number of DTAs and, thus, a coarse-grained transit perception.

However, as Figure 8 also depicts, this increment comes with a remarkable side effect; larger DTAs become more *sensitive* to changes. Since they cover a large spatial region, they are also more likely to suffer movement shifts of its inner routes. For instance, according to Figure 8, given 3000 m cells, the system is only composed of 41 DTAs, but, on the contrary, the number of average number of changes per area was 123. In that sense, a large number of DTA changes imply that the system transitions from one operation mode to the other many times. In large deployments this can imply a large usage of resources from both the server's and contributors' point of view.

All in all, considering the existing trade-off among cell size, providing DTA granularity, and number of DTA changes, providing mode-transition *stability*, Figure 8 depicts that the configuration providing an acceptable granularity and stability is the one with a cell size of 1000 m. Given this size, 270 DTAs were generated where each one changed, on average, 69 times.

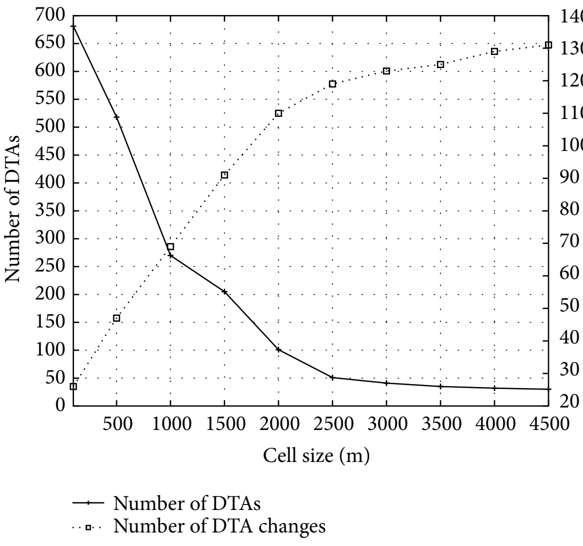


Figure 8: Number of DTAs and average number of changes per DTA with respect to the cell size of $\mathcal{C}$.

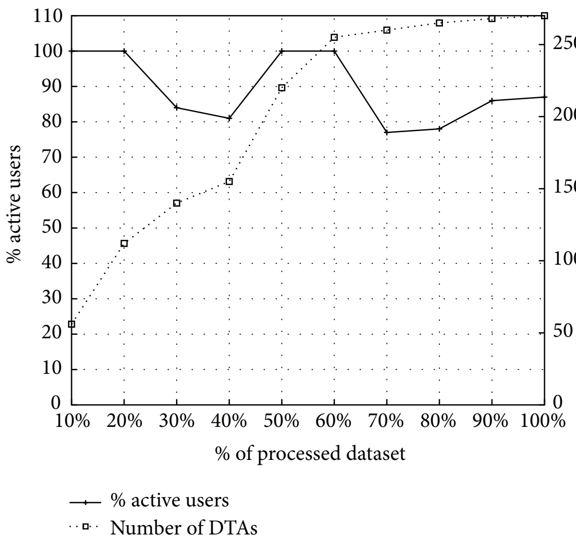


Figure 9: Percentage of active users and number of DTAs as the system proceeded.

5.3. System Evolution. One of the key features of the proposed system is its capability to compose the DTAs at the same time users are covering their routes without relying on any type of offline or previous data analytic stage. Therefore, Figure 9 shows the evolution of the system while it processed the GL dataset in terms of number of generated DTAs and percentage of active users reporting their routes at each moment.

According to this figure, we see that the system was able to reduce the number of active users about 20% with respect to the total number of contributors. Regarding the generation of DTAs, the system steadily generates them. Once

Table 5: Precision of DTA direction change detection with respect to a spatial distance approach.

	DSSIM				
	0.2	0.4	0.6	0.8	1.0
System precision	0.21	0.32	0.43	0.74	0.88

Table 6: Precision of the DTA speed change detection with respect to [6].

	% routes with speed alert within a DTA					
	10	20	30	40	50	60
System precision	0.21	0.32	0.76	0.81	0.91	0.93

it processed 60% of the dataset, the DTA generation was reduced in meaningful terms. At this point, the system was able to compose a map of DTA detailed enough to capture the whole urban dynamics represented by the contributors. As a result, a reduction of the number of active users is also perceived because the system spends more time in *DTA monitoring* mode.

On the whole, according to the stated results, the generation of the DTA-based model in real time is feasible in terms of accuracy. However, it is true that the time required by the system to compose such model could be a drawback when a fast detection of the whole model is required.

5.4. DTA Change Detection Accuracy. Finally, we also analysed the accuracy of the approach when it comes to detecting mobility shifts in a DTA. Since such changes are based on speed and directional features, we have compared the capability of the system to detect variations of these two features with respect to two different existing solutions in the spatiotemporal data mining field.

5.4.1. Direction-Based Change Accuracy. Regarding direction-based changes, we have compared our proposal with the DSSIM function [11]. This method measures the dissimilarity between two spatiotemporal trajectories during a time period $[t_1, t_n]$ as follows:

$$\text{DISSIM}(R, S) = \int_{t_1}^{t_n} D(R(t), S(t))\, dt, \tag{9}$$

where R and S are the two trajectories to compute and D is the Euclidean distance norm.

Given such a metric, we measured the average distance between the subroutes visiting each DTA in a particular sampling interval and the ones comprising the historical information of the DTA. The idea is that if long distances were detected then the direction of the current routes visiting the DTA and the historical ones is different.

Therefore, we correlated such average distances with the number of times the system perceived a transit change in a DTA so as to study whether a DTA change detection actually represents long dissimilarities between historical and current routes. The results of this study are shown in Table 5.

This table represents the DSSIM distances normalized with respect to the size of the DTA. Therefore, DSSIM values close to 1 indicate that there were large differences between the historical and the current routes in a DTA whereas smaller values stand for higher levels of similarity. In the light of these values, Table 5 shows the precision of the system to detect DTA changes. Such measurement is computed as follows:

$$\text{Precision} = \frac{\#\text{DTA changes}}{\#\text{DTA changes} + \#\text{miss DTA changes}}, \tag{10}$$

where #DTA changes is the number of times the system detected a change for DTAs having the DSSIM values under consideration whereas #miss DTA changes is the number of times the system did not consider that a DTA had changed given the DSSIM value under consideration.

As we can see, the system had a higher precision for large DSSIM values that indicate very evident differences between current and historical routes. In that sense, our proposal achieved acceptable precision results, about 0.8, when DSSIM ranged between 0.8 and 1.0. However, the system's precision decayed for DSSIM values below 0.6. This is because, at such point, certain differences of the routes do not imply changing their incoming or outgoing sides in the DTA which are the features used by the proposal to detect transit changes and, thus, become *invisible* to the system.

5.4.2. Speed-Based Change Accuracy. For the speed feature we followed a similar approach to the one for the direction. In this case, we used the event-based mechanism proposed in [6] to detect abnormally high or low speeds of moving objects in real time. Although it was initially designed for the maritime environment, it can be easily adapted to any kind of mobility domain.

In more detail, we counted the percentage of routes for which the aforementioned mechanism reported an abnormal speed within each DTA. Then, we calculated the precision of the system for each DTA as it was done for the direction. The comparison of these two values is shown in Table 6.

Results in this table confirm a similar behaviour of the system to the one for the direction; the larger the speed variations, the more precise the system. For example, when 60% of the routes within a DTA reported an abnormal speed, the precision of our system was 0.93. At the same time, when a small set of routes have unusual speed values then the precision of the system was affected with a remarkable decrease. For example, if the number of routes with abnormal speed moves from 30% to 20% then the system precision drops from 0.76 to 0.32. Nevertheless, the precision of the system in this case is slightly higher than the one based on direction.

5.5. Results and Conclusions. From the whole evaluation described above, we can draw up the following main conclusions.

(i) Firstly, the size of the cell has a great impact not only in terms of model granularity but also in terms of system stability. Consequently, its configuration should be carefully chosen in each deployment bearing in mind that large DTA could lead to an intensive usage of the contributors' device capabilities.

(ii) The generation from the scratch of the DTA map comes with the price of a long convergence period in case of large city deployments like the one used in this evaluation. However, this could be a minor problem in case of scenarios where it is not possible to collect a reliable dataset for offline mobility mining.

(iii) The accuracy of the system to detect routes' speed or direction changes within DTAs is slightly below that of the two approaches taken as reference, DSSIM and the event-based abnormal-speed detector [8]. Nonetheless, we should note that both approaches need the whole sequence of timestamped locations to operate. On the contrary, the present proposal only stores a few fields per DTA to compute such changes with the consequent saving of resources.

6. Related Work

In this section we review the state of the art of MCS within the mobility mining discipline with respect to our proposed solution. Moreover, due to their relationship with our proposal, we also provide a general overview of current efforts for human mobility modelling and the different techniques for regions of interest detection.

6.1. MCS-Based Mobility Mining Solutions. Several success stories demonstrate the suitability of MCS to support the development of mobility mining applications. In that sense, we can observe two major trends of MCS-based solutions.

On the one hand, an important course of action makes use of MCS for mapping activities by collecting the spatiotemporal traces of contributors [12, 13]. The idea here is to compose collaborative maps comprising not only road networks but also routes that are interesting in different domains like cycling or hiking.

On the other hand, due to the MCS potential for providing near real-time information, this sensing paradigm has been widely used for traffic monitoring. In that sense, a well-established line of work proposes the distributed architectures to keep track or predict road traffic congestions within an area by means of the mobility reports generated by the on-board units of vehicles [14, 15].

These MCS-based traffic monitoring architectures have recently profited from smartphones' sensing capabilities. As a result, now we can find solutions that combine static and vehicle-mounted and smartphone sensors to detect the road traffic in an area [16, 17].

Like the different lines of research described above, our mechanism also makes use of the MCS paradigm to uncover the transit state in a particular area of interest. However, the mobility knowledge extracted by the aforementioned solutions focuses on road traffic features whereas our solution intends to capture the human dynamics from a wider perspective as our model based on DTAs is independent of any road network topology.

6.2. Mobility Models Generation. In recent years, various works have considered the processing of spatiotemporal traces from users to extract relevant knowledge [18]. These digital breadcrumbs can be collected from several sources like location-based social networks [19], motion sensors [20], or smart cards [21]. In that sense, GPS traces have been one of the most popular datasources in this field.

In order to compose a model that represents the mobility of an area of interest, trajectory pattern mining has been widely studied, and works within this discipline can be classified into three lines of research, namely, frequent item mining, trajectory clustering, and graph-based trajectory mining [22].

However, during the last years a host of studies have put forward novel approaches to generate probabilistic models for mobility modelling that do not explicitly compose trajectory patterns. In these types of approaches, a database of historical routes is usually used to make up such models. In that sense, Bayesian networks [23], hidden Markov models [24], or Markov decision processes [25] have been some of the applied solutions.

In this case, our work provides a novel approach to represent the urban dynamics of an area by proposing an intermediate solution between trajectory pattern solutions and probabilistic modelling. For example, it does not explicitly generate individual or collective trajectory patterns but comprises information about how routes move within a DTA in terms of direction and speed. Furthermore, unlike the works cited above, our solution does not rely on any type of historical data as it composes the DTA map at the time it receives the routes.

6.3. Regions of Interest Detection. When it comes to detecting the regions of interest (ROIs), given a collection of spatiotemporal traces, a well-known approach consists of applying different types of clustering algorithms to such traces to uncover the target ROIs.

On the one hand, density-based clustering has been one of the prominent solutions in this area [26, 27]. These methods cluster using measures such as the distance between GPS points or density connectivity in a two-dimensional Cartesian space. In that sense, our proposal does not rely on any type of density-based clustering applied to trajectories.

Despite this high accuracy, density-based clustering algorithms need to keep the whole set of GPS traces so as to uncover the clusters and their complexity is exponential with respect to the available number of points. On the contrary, our system makes use of an aggregation method to compute certain features of the routes in a predefined space partitioning. This lightweight approach does not require keeping all the GPS traces in disk avoiding potential privacy leaks.

On the other hand, a different approach makes use of frequency map based spatial-temporal clustering methods

[28–31]. In this case, the area is partitioned into a fine grid of cells and assigns a weight to each cell around each GPS point based on the duration of the GPS staying at that point or its low speed. Next, a cell is considered a ROI when its weight is above a predefined threshold.

Our DTA detection mechanism also relies on a predefined spatial partitioning. However, unlike the aforementioned works, our approach does not intend to detect meaningful ROIs where people tend to remain stopped, but it centers on actually detecting ROIs (DTAs) related to the movement of people.

7. Conclusions

Opportunistic mobile crowdsensing has become a foremost parading in the mobile computing era. The endless improvement of the inner equipment of mobile phones and vehicle-mounted devices has made such a paradigm instrumental so as to uncover mobility phenomena in the urban domain.

In this context, the present work introduces a novel approach to leverage such paradigm and compose a map of DTAs within a city representing some of its mobility features. In that sense, a novel multilevel grill partitioning of the space to represent these DTAs has been put forward.

Besides, such a representation avoids storing any type of personal information of the contributors so that the generated map of DTAs can be disclosed without potential privacy leaks. Last but not least, a novel mechanism based on MCS to detect such DTAs based on two different operation modes has been stated. These two operation modes allow the system to select the best set of contributing users depending on whether the goal is to discover new DTAs or just control the existing ones.

Finally, future work will focus on enriching the sensing mechanism by incorporating context-aware features in the mobile clients so that contributors can proactively decide when and where to activate their sensing capabilities. This would reduce the dependency with the central server.

Abbreviations

DTA:	Dense transit area
$\mathscr{A}$:	Set of discovered DTAs
$r(p)$:	Ongoing route of the user p
$r(p)_l$:	Sequence of timestamped locations of the route $r(p)$
$r(p)^{\text{ended}}$:	Finished route of a user p
$r(p)_l^{\text{ended}}$:	Sequence of timestamped locations of the route $r(p)^{\text{ended}}$
$\mathscr{S}_{\text{gap}}$, $\mathscr{T}_{\text{gap}}$:	Temporal and spatial gaps between consecutive routes
$\mathscr{C}$:	Grilled partition of the city under study
c:	Squared cell in $\mathscr{C}$
$\mathscr{C}(c)^{\text{sub}}$:	Set of subcells of a cell c
$r(p)_c^{\text{ended}}$:	Sequence of cells c of the route $r(p)^{\text{ended}}$
$r(p)_c$:	Sequence of cells c of the route $r(p)$
$\mathscr{C}_{\text{tr}}^{w}$:	Historical data of walking routes in $\mathscr{C}$
$\mathscr{C}_{\text{tr}}^{\text{nw}}$:	Historical data of nonwalking routes in $\mathscr{C}$
c_{tr}:	Historical data of a cell in $\mathscr{C}_{\text{tr}}^{w}$ or $\mathscr{C}_{\text{tr}}^{\text{nw}}$.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This paper has been possible partially by the European Commission through the H2020-ENTROPY-649849 and the Spanish National Project CICYT EDISON TIN2014-52099-R (Grant BES-2015-071956) granted by the Ministry of Economy and Competitiveness of Spain (including ERDF support).

References

[1] Z. Yan and D. Chakraborty, "Semantics in mobile sensing," *Synthesis Lectures on the Semantic Web: Theory and Technology*, vol. 4, no. 1, pp. 1–143, 2014.

[2] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 29–35, 2014.

[3] C. Krner, M. May, and S. Wrobel, "Spatiotemporal modeling and analysis-introduction and overview," *KI-Künstliche Intelligenz*, vol. 26, no. 3, pp. 215–221, 2012.

[4] A. M. Hendawi and M. F. Mokbel, "Predictive spatio-temporal queries: a comprehensive survey and future directions," in *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (MobiGIS '12)*, pp. 97–104, ACM, 2012.

[5] A. Steinfeld, D. Manes, P. Green, and D. Hunter, "Destination entry and retrieval with the ali-scout navigation system," Tech. Rep. UMTRI-96-30, University of Michigan,Transportation Research Institute (UMTRI), 1996.

[6] F. Terroso-Saenz, M. Valdes-Vela, and A. F. Skarmeta-Gomez, "A complex event processing approach to detect abnormal behaviours in the marine environment," *Information Systems Frontiers*, vol. 18, no. 4, pp. 765–780, 2016.

[7] J. Zhang and M. F. Goodchild, *Uncertainty in Geographical Information*, Taylor & Francis, London, UK, 2002.

[8] F. Terroso-Saenz, M. Valdes-Vela, E. den Breejen, P. Hanckmann, R. Dekker, and A. F. Skarmeta-Gomez, "CEP-traj: an event-based solution to process trajectory data," *Information Systems*, vol. 52, pp. 34–54, 2015.

[9] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry*, Springer, Berlin, Germany, 2000.

[10] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: a collaborative social networking service among user, location and trajectory," *IEEE Data Engineering Bulletin*, vol. 33, no. 2, pp. 32–39, 2010.

[11] E. Frentzos, K. Gratsias, and Y. Theodoridis, "Index-based most similar trajectory search," in *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*, pp. 816–825, Istanbul, Turkey, April 2007.

[12] M. Haklay and P. Weber, "OpenStreetMap: user-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[13] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "Bikenet: a mobile sensing system for cyclist experience mapping," *ACM Transactions on Sensor Networks*, vol. 6, no. 1, article 6, 2009.

[14] L. Zhang, D. Gao, W. Zhao, and H.-C. Chao, "A multilevel information fusion approach for road congestion detection in VANETs," *Mathematical and Computer Modelling*, vol. 58, no. 5-6, pp. 1206–1221, 2013.

[15] J. Wan, J. Liu, Z. Shao, A. V. Vasilakos, M. Imran, and K. Zhou, "Mobile crowd sensing for traffic prediction in internet of vehicles," *Sensors*, vol. 16, no. 1, article 88, 2016.

[16] Á. Petkovics and K. Farkas, "Efficient event detection in public transport tracking," in *Proceedings of the International Conference on Telecommunications and Multimedia (TEMU '14)*, pp. 74–79, Heraklion, Greece, July 2014.

[17] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher, "SmartRoad: a crowd-sourced traffic regulator detection and identification system," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN '13)*, pp. 331–332, ACM, Philadelphia, Pa, USA, April 2013.

[18] Y. Zheng and X. Zhou, *Computing with Spatial Trajectories*, Springer Science & Business Media, New York, NY, USA, 2011.

[19] H. Gao and H. Liu, "Mining human mobility in location-based social networks," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 7, no. 2, pp. 1–115, 2015.

[20] T. Guo, Z. Yan, and K. Aberer, "An adaptive approach for online segmentation of multi-dimensional mobile data," in *Proceedings of the 11th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE '12)*, pp. 7–14, Scottsdale, Ariz, USA, May 2012.

[21] N. J. Yuan, Y. Wang, F. Zhang, X. Xie, and G. Sun, "Reconstructing individual mobility from smart card transactions: a space alignment approach," in *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM '13)*, pp. 877–886, IEEE, Dallas, Tex, USA, December 2013.

[22] M. Lin and W.-J. Hsu, "Mining GPS data for mobility patterns: a survey," *Pervasive and Mobile Computing*, vol. 12, pp. 1–16, 2014.

[23] J. Krumm, R. Gruen, and D. Delling, "From destination prediction to route prediction," *Journal of Location Based Services*, vol. 7, no. 2, pp. 98–120, 2013.

[24] J. Zhou, A. K. H. Tung, W. Wu, and W. S. Ng, "A 'semi-lazy' approach to probabilistic path prediction in dynamic environments," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*, pp. 748–756, ACM, 2013.

[25] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell, "Navigate like a cabbie: probabilistic reasoning from observed context-aware behavior," in *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp '08)*, pp. 322–331, ACM, Seoul, South Korea, September 2008.

[26] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen, "Discovering personal gazetteers: an interactive clustering approach," in *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems (GIS '04)*, pp. 266–273, ACM, 2004.

[27] B. W. Sharman and M. J. Roorda, "Analysis of freight global positioning system data: clustering approach for identifying trip destinations," *Transportation Research Record*, no. 2246, pp. 83–91, 2011.

[28] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell, "NextPlace: a spatio-temporal prediction framework for pervasive systems," in *Pervasive Computing: 9th International Conference, Pervasive 2011, San Francisco, USA, June 12–15, 2011. Proceedings*, K. Lyons, J. Hightower, and E. M. Huang, Eds., pp. 152–169, Springer, Berlin, Germany, 2011.

[29] Z. Li, J. Han, B. Ding, and R. Kays, "Mining periodic behaviors of object movements for animal and biological sustainability studies," *Data Mining and Knowledge Discovery*, vol. 24, no. 2, pp. 355–386, 2012.

[30] T. Bhattacharya, L. Kulik, and J. Bailey, "Extracting significant places from mobile user GPS trajectories: a bearing change based approach," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*, pp. 398–401, ACM, Redondo Beach, Calif, USA, November 2012.

[31] X. Xiao, Y. Zheng, Q. Luo, and X. Xie, "Inferring social ties between users with human location history," *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 1, pp. 3–19, 2014.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 2630439, 17 pages
http://dx.doi.org/10.1155/2016/2630439

Research Article

A Safety Enhancement Broadcasting Scheme Based on Context Sensing in VANETs

Chen Chen, Hongyu Xiang, Canding Sun, and Qingqi Pei

State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China

Correspondence should be addressed to Chen Chen; cc2000@mail.xidian.edu.cn

Received 14 April 2016; Revised 3 July 2016; Accepted 4 August 2016

Academic Editor: Carlos T. Calafate

The broadcasting plays a vital role for context awareness in VANETs (Vehicular Ad Hoc Networks) whose primary goal is to improve the driving safety depending on effective information exchanging. In this paper, based on the LQG (linear quadratic Gaussian) optimal control theory, a broadcasting control scheme named LQG-CCA is proposed to improve the network throughput thus increasing the opportunities for the safety-related events to be successfully handled. By predicting the network throughput with the Kalman filter model, our LQG model is envisioned to minimize the difference between the predicted and expected throughput through the adjustment of CCA (Clear Channel Assessment) sensing threshold. Numerical results show that our proposed model can significantly improve the network performance in terms of average throughput, average End-to-End delay, and average packets delivery ratio compared with a highly cited work D-FPAV and a latest published model APPR.

1. Introduction

VANETs are being developed for applications on road including mainly safety-related events, such as Cooperative Collision Warning (CCW) [1], traffic signal violation warning, and lane change warning. Although the critical applications usually rely on event-driven messages exchange in very emergent cases, the periodical broadcasting in VANETs also plays an important role in safety enhancement and guarantee. For those safety-related applications, the correctness and up-to-dateness of the messages greatly depend on the emergent broadcasts delivery ratio and periodical beacons frequency. For instance, in Cooperative Collision Warning or Cooperative Collision Avoidance (CCA) systems, the successful reception of braking notifications from neighboring vehicles directly influences the crash probability which depends on the intervehicle distance and/or packets transmission delay. An example is illustrated in Figure 1. Although the two following vehicles behind the front braking vehicle may have extra time to prepare for avoiding the potential crash with wireless broadcasting enabled, the packets collision from other beacons or event-driven messages may fail this emergency notification transmission. Another instance is shown in Figure 2. In intersections where approaching vehicles from different directions become blind spots in each other's horizon, at least one of the broadcasted messages should be received by the two potentially colliding vehicles to prevent severer accidents. In the traffic light violation warning scenario [2] as shown in Figure 3, reliable broadcast-based notification mechanism is crucial for collision avoidance before the specific vehicle approaches the danger area.

With above illustrations, it can be concluded that a highly efficient and reliable broadcasting mechanism is very necessary for VANETs to guarantee the driving safety. To make such mechanism practical in real cases, a model with lower communication overheads as well as complexity to greatly reduce the experienced delay is required. This scheme should also have the ability to avoid the traffic congestion especially for the safety-related events. Generally speaking, the primary cause of the network congestion is that the network loads from the nodes exceed the storage and processing capacity of the network. Therefore, the purpose of an effective congestion control is to reasonably control the channel loads and to ensure that the loads put on the network are

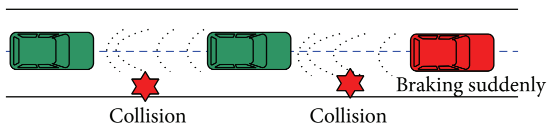


Figure 1: Cooperative Collision Avoidance scenario.

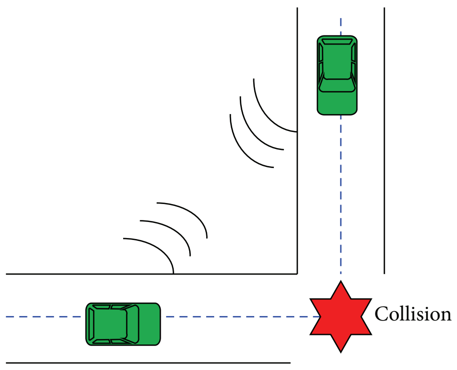


Figure 2: Blind spot detection scenario.

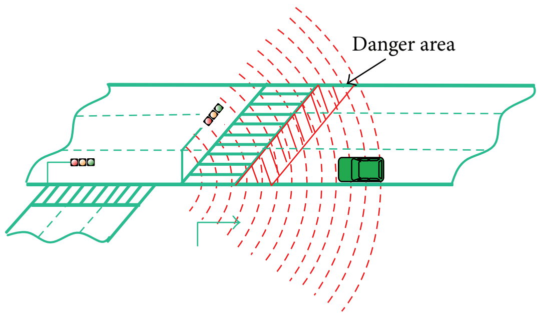


Figure 3: Traffic light violation scenario.

within the resource and processing capacity of the network while meantime making full use of the provided network resources, such as spectrum in our work. Fortunately, many achievements have been made on the congestion control issue in the research field of wireless Ad Hoc networks [3–5]. But due to the specificity of VANETs (i.e., rapid changing topology, specific mobility model, and variable vehicular density), the strategies used in the wireless Ad Hoc network cannot be directly applied to VANETs [6]. As a result, in this paper, a broadcasting congestion control framework has been proposed which takes the special characteristics of VANETs into account and attempts to fully utilize the wireless spectrum to maximize the network throughput while not making the channel congested at the same time, especially for the safety-related cases.

Note that our proposed model is adaptive to the network contexts such as traffic status and channel states through the adjusting of the CCA threshold. In this way, the nodes in a network can independently decide their channel sensing status through different local CCA configurations, thus making the global throughput maximized. In addition, by adaptively controlling the CCA threshold, the packets collisions due to hidden terminals could also be significantly alleviated [7]. Meanwhile, since we aim to control the channel congestion as well as maximize the network throughput, the requirements between transmission delay and network goodput especially for safety-related broadcasting messages could be well balanced in VANETs.

The rest of this paper is organized as follows. Section 2 describes some related works and gives their pros and cons. Section 3 provides the details of our proposed LQG-CCA model. Section 4 evaluates the performance of our model with numerical results generated via NS2. Section 5 concludes this paper.

2. Related Work

By looking at the safety- or non-safety-related applications and their data traffic in VANETs, as well as the current and foreseen possible multichannel strategies, there is a risk that the corresponding radio spectrum could be readily saturated if no congestion control algorithms are taken especially in the dense environment. This channel saturation would result in unstable vehicular communications thus failing the promised applications. Therefore, although the congestion control scheme now has not been proposed as a mandatory term in related standards or drafts, we think it is extremely necessary to include this feature in the VANETs standardization process and its future realization. Considering the high mobility and dynamic network topology in VANETs, a lot of congestion control algorithms have been executed in a centralized way which has been proved to be successful. P. Rani and M. Rani [8] proposed a centralized scheduling technique by controlling congestion for safety messages in the vehicular environment. Guan et al. [9] proposed a centralized adaptive congestion control method for the vehicular networks in road intersections. Wang et al. [10] presented a method for finding an optimal tradeoff between network congestion and the freshness of received information in a cellular-based vehicular network. Although previous studies showed that centralized control might be suitable and efficient in the vehicular environment, their results are actually only applicable to the case where delay requirement is not much strict. For our discussed safety-related events, the introduced communication overheads from/to the central control node may be huge compared to the notifications or context aware messages between vehicles and might finally make the designed model impractical and events fail. As a result, the distributed and self-organized capability is sometime preferable in VANETs especially for delay sensitive applications.

For decentralized congestion control schemes, there have been several works which can be mainly classified into 3 categories, that is, utility, power control, and rate adjustment based. Some works also seek methods to avoid information congestion by regulating beacon frame length, but we consider this sort of strategy to be not representative because packets are generally very small in VANETs especially for safety-related notifications.

For utility-based methods, the design principle is to assume a strictly concave utility function $u(r)$ of some given network parameters such as packet sending rate, channel capacity, transceiver power level, network throughput, or their specific combinations. The congestion control scheme is then a way to adjust the parameters to maximize total utility of the system. Wischhof and Rohling [11] proposed a Utility-Based Packet Forwarding and Congestion Control scheme (UBPFCC) that works on top of IEEE 802.11x series MAC protocols with emphasis on nonsafety applications. They designed an application-specific utility function and encoded the payoff information in each transmitted data packet from a node locally. Then, a decentralized algorithm was used to assign a share of the available data rate to nodes proportional to their relative priorities which were calculated based on their utilities to the "average utility value." However, as the authors stated in their paper, their work is only applicable to the comfort applications of VANETs whereas the safety-related applications, which have very stringent delay and reliability requirements, were not considered at all. Bouassida and Shawky [12] presented a cooperative and fully distributed congestion control approach in VANETs based on dynamic priority scheduling and transmission. Their priority is composed of two parts, that is, the static and dynamic part, and its assignment took the "messages" utility into consideration through counting the number of packets retransmissions in the neighborhood. By combining the static assignment depending on the application type and the dynamic determination upon the specific context of the VANETs (neighborhood density, node speed, and message utility and validity), the priorities of sent packets were designated and used for messages scheduling in order to avoid congestion in the overall network. However, their application type specific priority configuration is a little bit arbitrary where no metric or calculation formula is given to determine the exact value of the priority for a specific service. Additionally, how to determine "Service Channel Congestion Threshold" was not given in this paper. Zhou et al. [13] proposed a scheme jointly formulating the rate control, medium access control, and routing problem for cooperative VANETs in the framework of the utility function optimization. Although their utility function considered the tradeoff between users fairness, network cost, and users rewards, their work cannot be used in safety background where delay requirement and QoS (Quality of Service) should be taken into account.

For congestion control strategies relying on transmitting power adjustment, the design goal is to balance the tradeoff between power and packets generation rate. Generally, although a higher packet generation rate can increase the information accuracy with frequent updates, an uncontrolled strategy could also readily lead to a saturated medium. Likewise, a message sent with higher transmitting power can reach further distances, but it will also increase the level of interferences to other ongoing transmissions. Therefore, how to handle the tradeoff between power and sending rate will influence the up-to-dateness of sent messages as well as the channel congestion level. Torrent-Moreno et al. [14] proposed a fully distributed and localized power control algorithm called Distributed Fair Power Adjustment for Vehicular Networks for adaptive transmitting power adjustment which is formally proven to achieve max–min fairness. Although D-FPAV was proven effective through simulations under different radio propagation models, its major concern lies on the fairness in terms of channel busy time sensed by every node in the highway. Accordingly, D-FPAV is not applicable in urban environment for safety-related applications which care more about the successful completion ratio instead of fairness. Guo et al. [15] proposed a delay-aware and reliable broadcast protocol (DR-BP) based on transmitting power control. In their model, when emergency events occur, a local optimal relay selection mechanism is designed by which only the vehicle selected can forward warning messages. In this way, the extra redundant warning messages can be effectively restrained thus not making the network congestion and improving the delay performance for safety messages.

For generation rate control on periodic beacon messages, the design goal is to balance the tradeoff between the updating rate/interval and latency of received messages in order to avoid congestion. Mitra and Mondal [16] proposed two distributed channel congestion control algorithms by controlling the message generation rate in VANETs. Note that RSUs were actually introduced in this paper for access authentication and further congestion reduction which may make this work categorized as a centralized model. However, their proposed two algorithms for message generation rate control are fully distributed and can be locally executed on each vehicle. In their first approach APPR-1, the channel load is maintained to an estimated initial value calculated depending upon the message generation rate of different vehicles. For the second approach APPR-2, the channel load is continuously increased till the percentage of message loss lies below a predefined threshold. Sommer et al. [17] proposed a novel approach to dynamic beaconing which can provide low-latency communication (i.e., very short beaconing intervals), while ensuring not to overload the wireless channel, in the presence of radio signal obstructions caused by other vehicles and by buildings. It is worth noting that their designed dynamic beaconing (DynB) scheme allows more aggressive channel use in time-variant signal shadowing environment than in realistic case where no obstacle is considered. Although DynB introduces more practical propagation models into the design process of the information dissemination schemes in vehicular networks, their major concern actually does not focus on the safety applications along with their various safety requirements. Nevertheless, their evaluation of the impact of vehicles and buildings shadowing on the performance of beaconing protocols is really beneficial. In our work, we also take the obstacles effect into account and discuss our congestion control problem in a radio signal shadowing environment. Bai et al. [18] proposed a distributed beacon scheduling scheme CABS (Context Awareness Beacon Scheduling) which is based on the spatial context information to dynamically schedule the beacon by means of a TDMA-like manner. Their numerical results showed that CABS can efficiently use the limited network resources without leading to congestion and satisfy the requirements of safety applications as well. Although their work also introduces the time-slot-based scheduling, CABS is actually a TDMA-like protocol where channel contention has been resolved via slots assignment by the centralized management node. However, our work attempts to model the channel competitions in a full distributed environment where different vehicles randomly access the channel locally. Sahoo et al. [19] proposed a congestion-controlled-coordinator-based MAC (CCC-MAC) using time-slot-based medium access protocol to address the packets congestion and safety guarantee considerations in VANETs. By incorporating a pulse-based slot reservation mechanism, their proposal can ensure fast and reliable propagation of emergency messages over multiple hops under different vehicular densities. Even though CCC-MAC is confirmed by its ability to deliver safety-critical messages to around 95% of the intended recipients, it also needs centralized scheduling to ensure that all vehicles in a segment can receive time slots for their beacon transmissions. Chaabouni et al. [20] proposed a congestion control approach that uses the number of detected collisions as a metric to control the beacon generation frequency and therefore reduce the effect of congestion and improve e-Safety. Their works verified that they can achieve a balanced tradeoff between beacon information accuracy and beacon related overhead. However, in their work, the beacon generation rate is dynamically adjusted based on the number of collisions locally detected by each node, which may imply a misjudgment of the network congestion status since local information will not always correctly respond to the global state. In addition, how to determine the important threshold V_{thresh} used for initiating the rate adjustment process was not given. Tielert et al. [21] designed a protocol named PULSAR (Periodically Updated Load Sensitive Adaptive Rate control) which aims at controlling channel load and in the meantime satisfying safety applications' awareness requirements. Their reactive congestion control strategy adjusts a vehicle's transmission rate based on guidance by the application layer on transmission range as well as minimum and maximum transmission rate. Although PULSAR is a distributed algorithm and can accommodate different min/max transmission rate intervals and radio ranges, it mainly focused on access fairness but gave a little bit of consideration on differentiated traffic which was common in VANETs with different prioritized services. Javed and Khan [22] proposed a space-division multiple access (SDMA) technique combined with an adaptive rate control mechanism to improve the efficiency of BSM (Basic Safety Message) transmissions. Their SDMA can reduce the interference among vehicles and address the hidden-node problem by introducing space-division access opportunities. However, the efficiency of SDMA is highly dependent on the accuracy of the estimation of density, which is usually difficult to be collected on the fly in real time and be precise enough in a highly dynamic environment.

Actually, the congestion avoidance or control according to CCA threshold adjustment could be generalized to the power control category since such adjustment will directly influence the determination of the channel busy states based on carrier sense range modification, which is usually altered by the transmitting power. For the CCA threshold adjustment based on adaptive schemes, there are already many previous existent works. Zeng et al. [23] designed an EWM (emergency warning message) dissemination algorithm that uses a CCA threshold ladder setting mechanism to deal with the road vehicle's abnormal event. In this paper, message ranking is combined with CCA adjusting which is judged by Dissemination Successful Rate (DSR) and Packet Delivery Delay (PDD). However, this method is not valid for networks with IEEE 802.11p radios. Meanwhile, systematic evaluation of this conjecture is not given in the paper. Similarly, Han et al. [24] proposed a CCA threshold selection method to imitate the behaviors of IEEE 802.11 MAC in multihop networks. By calculating the interference and transmitting probability of senders, the performance of EWMs can be improved through CCA adjustment to ensure both real time and reliability of EWM transmission. However, their experiments are only tested on a stationary test bed without mobility considered, which may suffer performance degradation when mobility model is enabled. Cho et al. [25] proposed a method to set the carrier sensing threshold by computing the self-interference at a secondary user based on the distance separation in a cognitive radio oriented wireless network. In their work, the carrier sensing threshold is presented as a common parameter to control the activity of the secondary network. The proposed method has low complexity and makes it possible to compute the carrier sensing threshold in real time. However, since cognitive radio is introduced in this work, the complexity in terms of time and computation on "spectrum hole" detection and awareness is nontrivial. Schmidt et al. [26] also proposed a stepwise CCA Threshold Adaptation (CTA) scheme depending on how long a packet has been waiting for medium access. Numerical results showed that their approach can mitigate significantly the problem of local message drops and hence local congestion.

Different from the aforementioned CCA-based works, by introducing LQG optimization control model, our algorithm aims at maximizing the global throughput by controlling the CCA threshold for each node locally. In addition, through the distributed transmission opportunities control, all nodes in a network could cooperatively schedule their transmitting thus maximizing the global profit.

3. LQG-CCA

In this section, the LQG optimization algorithms used to control our broadcasting scheme are given in the vehicular environment. The Linear Quadratic Regulator (LQR) [27] targets the linear system that takes the form of the state space in the modern control theory, and its objective function is a quadratic function of the object's status and the control input. In LQR, a state feedback controller S is designed to minimize the quadratic objective function, where S is uniquely determined by the weight matrices Q and R. Although the LQR theory is the oldest method for designing the state space, it is capable of providing the optimized control rule according to the linear feedback on the state, which is very helpful to our discussed problem.

Since high mobility inevitably introduces random disturbance into the practical vehicular network, our congestion control issue can be transformed to an optimal control problem to maximize the network throughput in the presence of the random disturbance. Note that we did not take the latency as our optimization objective since different services have different delay requirements and a global minimization of delay might not guarantee the success even for the most emergent event. In addition, the random disturbance is assumed to be the Gaussian noise in our work for the sake of analysis. Therefore, our control problem can then be formulated as an LQG optimal control issue. Fortunately, a famous law of segregation [28] already exists for our discussed case, enabling us to study noise reduction and congestion control separately. In this way, while we study the congestion control problem, it can be assumed that the network is free of noise; thus, all state variables could be accurately obtained.

Our algorithm mainly consists of two parts, that is, the Kalman filter model based prediction and LQG optimization congestion control. Let us discuss them one by one in the following.

3.1. Constructing the LQG Model for Our Congestion Control Problem. Actually, our congestion control problem could be formulated as a feedback control issue in the domain of the Networked Control Systems (NCSs) [29], in which the performance of a system is iteratively adjusted in a closed-loop manner according to the generated feedback from the system. In addition, as commented in almost all the literatures on NCSs, the motivations to construct such a control system via networks are its low installation and maintenance costs, high reliability, increased system flexibility, and decreased wiring. On the other hand, a networked system also introduced some other issues to influence the output performance of a system such as uncertain delay caused by traffic congestion, packets loss due to medium collision or channel failure, and throughput degradation due to limited bandwidth. In this paper, to alleviate the impact of traffic congestion on the performance of VANET broadcasting, an LQG-based optimal control model is applied to maximize the system throughput according to the measured throughput and distributed CCA threshold control.

Generally, an LQG controlled system could be further divided into the state estimation and control part [30], respectively. In our work, the state of the next interval is estimated using the Kalman filter model according to the presently measured state considering the disturbance from the system noise. Note that the packets transmissions on the shared wireless medium are vulnerable to the channel failure and packets collisions as well as noise interference. Therefore, we compensate this performance loss in advance by intelligently adjusting the coefficients of the observed state and outputted control vectors. In addition, during the control procedure, to minimize the output error between the predicted and expected throughput, our throughput maximization problem is transformed to an optimal control issue, which in return reduced the packet loss ratio and improved the system throughput.

The notations used for constructing and updating the state space model are listed at the end of the paper for convenience.

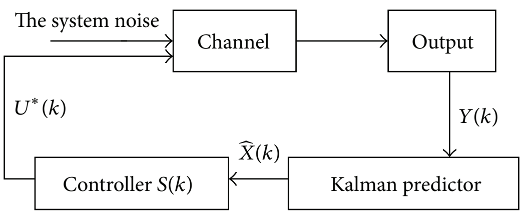


FIGURE 4: Network configuration.

Next, let us propose our LQG model regarding our congestion control problem in detail. With reference to literature [31], a state space could be described with the following discrete linear time-invariant stochastic system:

$$X(k+1) = AX(k) + BU(k) + GW(k), \tag{1}$$

$$Y(k) = CX(k) + V(k), \tag{2}$$

where (1) and (2) are the state and observation equations, respectively. $X(k)$ denotes the state variable (i.e., network throughput), $U(k)$ denotes the input control (i.e., CCA threshold), and $Y(k)$ denotes the measured output. A and B are the coefficient matrix of the state equation. C is the coefficient matrix of the observation equation. $W(k)$ and $V(k)$ are two uncorrelated white additive Gaussian noises with variances $Q(k)$ and $R(k)$, which will be discussed in detail later. G denotes the gain of the Kalman filter.

The network configuration for our throughput maxmization problem through dynamic CCA adjustment could be framed as shown in Figure 4. The detailed description regarding each component is given as follows.

3.1.1. Channel. The component "channel" is used here to model the transmisson medium in a VANET. The feedback control vector is taken as the input and the measured througput is taken as the output of the channel, respectively.

3.1.2. Kalman Predictor. A "Kalman predictor" is an estimator for the linear quadratic Gaussian (LQG) problem, which is the problem of estimating the instantaneous "state" of a linear dynamic system by using measurements linearly related to the state but corrupted by Gaussian white noise. For a dynamic system, it is not always possible or desirable to measure every variable that one wants to control. The Kalman filter provides a means for inferring the missing information from indirect (and noisy) measurements. In such situations, the Kalman filter is used to estimate the complete state vector from partial state measurements and is called an observer. The resulting estimator is statistically optimal with respect to any quadratic function of estimation error.

3.1.3. Controller. The "controller" module generates the control signal by minimizing the error between the predicted and exptected state. Specifically, given a positive time horizon, the aim is to find the optimal control policy $(U^*) = \{U_1^*, \dots, U_M^*\}$, to minimize the linear quadratic cost functional:

$$J(U(k)) = E\left(\sum e^T(k)Q(k)e(k) + U^T(k)R(k)U(k)\right), \quad (3)$$

where $J(U(k))$ is the cost function, $e(k)$ denotes the deviation of the predicted value from the expectation, $U(k)$ is the control variable, and the matrices $Q(k)$ and $R(k)$ are, respectively, the weighting matrices indicating the state and control cost penalties. $Q(k)$ and $R(k)$ are determined through multiple numerical tests in our work which will be discussed later.

By minimizing (3), the optimal control vector could be given as follows:

$$U^*(k) = -S(k)\widehat{X(k)}, \quad (4)$$

where $S(k)$ is the coefficient matrix, $\widehat{X(k)}$ is the input, and $U^*(k)$ is the output of the optimal controller, respectively.

3.2. Modeling the Relationship between the CCA Threshold and Throughput. To construct our LQG model and solve the congestion problem according to effective CCA control, the relationship between the CCA and finalized throughput should be figured out at first.

As the de facto standard of VANETs, the IEEE 802.11p [32] enables the CSMA/CA scheme to coordinate nodal transmission of messages. The CSMA/CA specifies that the node first senses the channel before transmitting, and it only sends the data when it senses that the channel is idle. If it finds that the channel is busy, which means another node is occupying the channel, the node should back off for a random duration for retransmission. For the CSMA/CA mechanism, the CCA threshold actually corresponds to a carrier sense area, with the node's sensing range being in reverse proportion to the CCA threshold. In other words, a higher CCA means that the node's sense area is smaller, resulting in an increased probability that the node attempts to transmit, even if another node is already transmitting on this channel. If the CCA is reduced, then the node's sensing range will instead increase, making it more likely to sense the channel being busy. As a result, the node will have a lower probability of trying to transmit. In conclusion, the higher the CCA threshold, the higher the collision probability. In addition, with the dropping of the CCA threshold, not only the collision probability but also the throughput and channel utilization decline.

In fact, the network capacity is an important measure of the network performance which mainly depends on two indexes, that is, the channel throughput and spatial multiplexing level. In general, the channel throughput refers to the amount of data transmitted on the channel per unit time, and it mainly relies on the SINR (Signal to Interference and Noise Ratio) of the current channel. The spatial multiplexing level indicates the number of node pairs that can transmit in parallel across the network and it is determined by the transmission power and CCA threshold. In our work, by adjusting the CCA threshold, some nodes will be restrained from transmitting thus increasing the opportunities of concurrent transmissions for others, improving the network throughput to a great extent through spatial multiplexing.

Next, according to the framework in Figure 4, our congestion control problem could be further formulated in detail as follows. As defined in (1) and (2), $X(k)$ denotes the network throughput and $U(k)$ denotes the CCA threshold in our work, respectively. We predict the throughput and compare it with the expected one, that is, the optimal throughput, and minimize the difference between the predicted and expected throughput by controlling the CCA threshold using our presented LQG controller. Actually, by adjusting the CCA threshold, some nodes will be restrained from transmitting thus increasing the opportunities of concurrent transmissions for others and improving the network throughput to a great extent.

As a result, before constructing the LQG model, we should first derive the relationship between the CCA threshold and throughput. The notations mentioned later to deduce their relationship are listed at the end of the paper for convenience.

In our envisioned model, a dense network is assumed and wireless stations are uniformly and independently distributed in an area of Area. A common and fixed transmission power P is used by each transmitter.

Assume that a vehicle produced an emergency warning message (EWM). Let us define the interferences to this source node within the investigated area Area:

$$I_s = \sum_{m\in\text{Area}} \sum_{i\in\text{Area}} \pi_m \text{Pow}(i,k), \quad (5)$$

where π_m is a constant. $\text{Pow}(i,k)$ is the signal strength on the receiver for packets from i at time k; that is,

$$\text{Pow}(i,k) = \text{Pow}(i) - [92.44 + 20\log f\ (\text{GHZ}) + \log d_{ik}\ (\text{km})], \quad (6)$$

where d_{ik} indicates the distance between i and its intended receiver at time k. $\text{Pow}(i)$ is the transmitting power of i.

To make the EWM successfully decoded, the SINR of the received signal should at least meet the SINR threshold at the receiver, say, tolerate an interference no smaller than I_s. With the CCA definition, we have

$$\text{CCA}_{\text{th}} \geq I_s. \quad (7)$$

As a result, to make the throughput maximized, the CCA will be

$$\text{CCA}_{\text{th}} = I_s. \quad (8)$$

Correspondingly, the SINR for this case can be expressed as follows:

$$\text{SINR} = \frac{\text{Pow}(i,k)}{N_0 + I_s} = \frac{\text{Pow}(i,k)}{N_0 + \text{CCA}_{\text{th}}}, \quad (9)$$

where N_0 is the noise power.

```
(1) Input: $\mathrm{Pow}(i,k)$, $\pi_m$, $N_0$
(2) Output: Throughput
(3) while ($i \in$ Area){
(4) If obtain the optimal CCA then
(5) CCA = $I_s$;
(6) $I_s$ is obtained by formula $I_s = \sum_{m\in \mathrm{Area}} \sum_{i\in \mathrm{Area}} \pi_m \mathrm{Pow}(i,k)$;
(7) Endif
(8)      **Insert** ($\mathrm{Pow}(i,k)$, $N_0$, $I_s$, CCA);
(9) SINR is obtained by formula $\mathrm{SINR} = \mathrm{Pow}(i,k)/(N_0 + I_s) = \mathrm{Pow}(i,k)/(N_0 + \mathrm{CCA})$
(10)      $i = i + 1$;
(11) } //end of while
(12) while ($\mathrm{SINR} \to 0$);
(13) Throughput = $C_0/X^2 * (1 + \mathrm{Pow}(i,k)/(N_0 + \mathrm{CCA}))$;
```

Algorithm 1: The derivation of the relation between throughput and CCA.

With the above analysis, the relation between throughput and CCA could be given as follows [33]:

$$\begin{aligned}\text{Throughput} &= \text{ChannelRate} * M \\ &= C_0 * \frac{\ln(1+\text{SINR})}{X^2},\end{aligned} \tag{10}$$

where ChannelRate is the achievable channel rate expressed with Shannon capacity; that is, ChannelRate $= W \log_2(1 + \mathrm{SINR})$. W is the channel bandwidth in Hertz. M accounts for the total number of concurrent transmissions. C_0 is a constant. $X = D/R$, where D is the carrier sense range and R denotes the maximum radio range.

When SINR $\to$ 0, throughput can be simplified as follows:

$$\text{Throughput} = \frac{C_0}{X^2} * \left(1 + \frac{\mathrm{Pow}(i,k)}{N_0 + \mathrm{CCA}}\right). \tag{11}$$

The pseudocode used to deduce the relationship between throughput and CCA can be described as shown in Algorithm 1.

Let $X(k) = \mathrm{THR}(k) = [\mathrm{thr}_1(k)\ \ \mathrm{thr}_2(k)\ \cdots\ \mathrm{thr}_M(k)]^T$, where $\mathrm{thr}_i(k)$ is the throughput of vehicle i at time k. Then, $\mathrm{thr}_i(k)$ can be derived as follows:

$$\mathrm{thr}_i(k) = \frac{C_0}{X_i^2} * \left(1 + \frac{\mathrm{Pow}(i,k)}{N_0 + \mathrm{CCA}_i}\right). \tag{12}$$

Let $U(k) = [\mathrm{CCA}_1(k)\ \ \mathrm{CCA}_2(k)\ \cdots\ \mathrm{CCA}_M(k)]^T$, where CCA_i is the CCA of vehicle i, and then (1) can be further derived as follows:

$$\begin{aligned}\mathrm{THR}(k+1) &= A * \mathrm{THR}(k) + B * \mathrm{CCA}(k) \\ &\quad + GW(k) = \begin{bmatrix} \mathrm{thr}_1(k+1) \\ \mathrm{thr}_2(k+1) \\ \vdots \\ \mathrm{thr}_M(k+1) \end{bmatrix} \\ &= A \begin{bmatrix} \mathrm{thr}_1(k) \\ \mathrm{thr}_2(k) \\ \vdots \\ \mathrm{thr}_M(k) \end{bmatrix} + B \begin{bmatrix} \mathrm{CCA}_1(k) \\ \mathrm{CCA}_2(k) \\ \vdots \\ \mathrm{CCA}_M(k) \end{bmatrix} \\ &\quad + G \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}.\end{aligned} \tag{13}$$

Since nodes share the wireless medium, the throughput of each node in the network is dependent on each other. Therefore, the state transition matrix A could be calculated as follows.

For an M-node network, the Markov chain is characterized by the $M\times M$ transition matrix $A = [p_{nk}]$ with p_{nk} being the probability that the network state goes from n to k in one transition. By [34], the Markov chain transition matrix A can be obtained as $A = [p_{nk}]$, where

$$\begin{aligned} p_{nk} &= \begin{cases} \displaystyle\sum_{y=n-k}^{n} \sum_{x=0}^{M-n} r_{(x+y)[x+(n-k)]} Q_r(y,n)\, Q_t(x,n), & 0 \le k \le n \\ \displaystyle\sum_{x=k-n}^{M-n} \sum_{y=0}^{n} r_{(x+y)[x-(k-n)]} Q_t(y,n)\, Q_r(y,n), & n \le k \le M, \end{cases} \\ Q_t(k,n) &= \binom{M-n}{k} (1-p_t)^{M-n-k} p_t^k, \\ Q_r(k,n) &= \binom{n}{k} (1-p_r)^{n-k} p_t^k, \end{aligned} \tag{14}$$

where $Q_t(k,n)$ denotes the probability of k transmissions from successful nodes given that there are n backlogged nodes. $Q_r(k,n)$ denotes the probability of k transmissions from backlogged nodes given that there are n backlogged nodes. p_t and p_r denote the packet transmission probability of successful and backlogged nodes.

Then, we can obtain the expression of the control coefficient B in (1) as follows:

$$B = \frac{\text{THR}(k+1) - A * \text{THR}(k) - GW(k)}{\text{CCA}(k)}. \quad (15)$$

Further, take $X(k)$ and $U(k)$ into (15) and simplify it; we can get the expression of B as

$$\begin{aligned} B &= \left[B_1(k) \;\; B_2(k) \;\; \cdots \;\; B_M(k)\right]^T, \\ B_i(k) &= \frac{C_0}{X_i^2 * \text{CCA}(k)} \cdot \left[\text{Pow}(i,k) \right. \\ &\quad \left. * \left(\frac{1}{N_0 + \text{CCA}(k+1)} - \frac{\sum_{j=1}^{M} p_{1j}}{N_0 + \text{CCA}(k)}\right)\right] \\ &\quad - \frac{C_0}{X_i^2 * \text{CCA}(k)} \cdot \left(\sum_{j=1}^{M} p_{1j} - G_1 W_1(k)\right), \end{aligned} \quad (16)$$

where $\sum_{j=1}^{M} p_{1j}$ is the sum of the first row of matrix A.

In [35], a strategy for dynamic iterative Kalman filtering has been proposed which proved that the output of every node in a Kalman filter is independent of each other. According to this conclusion, we assume the M nodes have identical measurement coefficient; that is, $C_1(k) = C_2(k) = \cdots = C_M(k) = 1$. Then, we could thereby take the measurement coefficient $C(k) = I$. Further, for the noise variables $W(k)$ and $V(k)$ in (1) and (2), they are two uncorrelated white Gaussian noises with covariances $Q(k)$ and $R(k)$; that is, they are independent of each other. Meanwhile, in $W(k) = [w_1 \;\; w_2 \;\; \cdots \;\; w_M]$, w_i $(i = 1, 2, \ldots, M)$ is distributed as a multivariate Gaussian with expectation 0 and covariance $Q(k)$ and, in $V(k) = [v_1 \;\; v_2 \;\; \cdots \;\; v_M]$, v_i $(i = 1, 2, \ldots, M)$ is distributed as a multivariate Gaussian with expectation 0 and covariance $R(k)$.

Then all variables in (1) and (2) have been deduced now.

3.3. Optimal Estimation. To accurately estimate the network throughput in the next discrete interval, the Kalman filter model is introduced to remove the noise from the estimation. With the network throughput at previous interval and the observation of throughput on the current interval, the Kalman filter can eliminate the impact of the noise on the state variable and iteratively compute the estimation of the network throughput on the current interval. To remove the noise with the Kalman filter, two stages are usually necessary, that is, estimating and updating. At the estimating stage, the filter forecasts the network throughput of the current interval using the estimated throughput at the previous interval according to the relation between carrier sense threshold and network throughput. At the updating stage, the filter uses the observation on the network throughput of the current interval to amend the estimation of the network throughout obtained at the estimation stage, resulting in an accurate observation on the throughput.

Let $k-1$ denote the previous interval and k denote the current interval, and suppose that system's network throughput and covariance at $k-1$ are known as $X(k-1 \mid k-1)$ and $Z(k-1 \mid k-1)$, respectively. Thereupon, the network throughput at k can be computed as follows:

$$X(k \mid k-1) = AX(k-1 \mid k-1) + BU(k), \quad (17)$$

where $X(k \mid k-1)$ is the estimated throughput the Kalman filter provides based on the network throughput at $k-1$. The covariance corresponding to $X(k \mid k-1)$ is computed as follows:

$$Z(k \mid k-1) = AZ(k-1 \mid k-1)A^T + Q(k-1), \quad (18)$$

where $Q(k-1)$ indicates the variance of $W(k-1)$. The $Q(k-1)$ is a specific value and it is determined through multiple numerical tests in our work which will be discussed in Section 3.4.

The Kalman gain $G(k)$ at k is computed as follows:

$$G(k) = \frac{Z(k \mid k-1)C^T}{CZ(k \mid k-1)C^T + R(k)}, \quad (19)$$

where R denotes the covariance of the measurement noise V. With the parameters obtained, we can correct the estimation of network throughput $X(k \mid k-1)$ at k made by Kalman filter in (10) to yield an accurate value of the throughput denoted by $\widehat{X(k)}$; that is,

$$\begin{aligned} \widehat{X(k)} &= X(k \mid k-1) \\ &\quad + G(k)(Y(k) - CX(k \mid k-1)). \end{aligned} \quad (20)$$

To enable the Kalman filter to operate iteratively, the covariance $Z(k \mid k)$ corresponding to $X(k \mid k)$ is given as follows:

$$Z(k \mid k) = (1 - CG(k))Z(k \mid k-1). \quad (21)$$

With (17)~(21), the predicted network throughput after the elimination of the noise can be worked out.

The prediction process is given as shown in Algorithm 2.

3.4. Computation of the Optimal Control Quantity. The linear quadratic optimal control model is used in our work to make the network throughput fulfill its expectation. In the case of network congestion, the network throughput will decrease and deviate from the expectation. By comparing the predicted network throughput with the expected one, we can compute the CCA threshold that enables the network throughput to meet the expectation using the linear quadratic optimal control model.

Next, according to the discussed LQG model in Section 3.1, our quadratic optimization control model is established as follows. First, we should define an appropriate expected network throughput $Y_r(k)$, which is dependent on the number of vehicles and road conditions. For the expected throughput, we establish a Markov chain to get the optimal throughput between the users with reference to the literature

```
(1) Kalman Initialization
(2) Input: $X(k-1 \mid k-1)$, $Z(k-1 \mid k-1)$
(3) Output: $\widehat{X(k)}$
(4) while $(k-1 < t \leq k)$ {
(5) State prediction
(6)       $X(k \mid k-1) \leftarrow AX(k-1 \mid k-1) + BU(k)$;
(7)       $Z(k \mid k-1) \leftarrow AZ(k-1 \mid k-1)A^T + Q(k-1)$
(8)       obtain the measure state value
(9)       $Y(k) \leftarrow CX(k) + V(k)$
(10)         **Insert** $(Z(k \mid k-1), C, R(k))$;
(11) state revision
(12) gain is obtained by the formula $G(k) = Z(k \mid k-1)C^T/(CZ(k \mid k-1)C^T + R(k))$;
(13) } //end of while
(14)   Output $(\widehat{X(k)} = X(k \mid k-1) + G(k)(Y(k) - CX(k \mid k-1)))$;
```

Algorithm 2: The optimal estimation.

[36]. If the network congestion occurs due to high density of vehicles, the predicted throughput $Y(k)$ will be less than $Y_r(k)$. Let $e(k) = Y_r(k) - Y(k)$ denote the deviation of the predicted throughput from the expectation. Based on this deviation, the cost function is defined as follows:

$$J(k) = E\left(\sum e^T(k)Q(k)e(k) + U^T(k)R(k)U(k)\right), \quad (22)$$

where $J(k)$ is the cost function, $E(\cdot)$ represents the mean value, $e(k)$ denotes the deviation of the predicted throughput from the expectation, $U(k)$ is the control variable, and the matrices $Q(k)$ and $R(k)$ are the weighting matrices, respectively, indicating the state and control cost penalties.

To obtain the value of $R(k)$, a numerical test is launched with different configurations of $R(k)$. According to [37], we set the order of $R(k)$ to 10^{-4} and choose its value as $0.1 * 10^{-4}$, $1 * 10^{-4}$, $5 * 10^{-4}$, $10 * 10^{-4}$, and $20 * 10^{-4}$, respectively. The corresponding results are listed in Table 1 for illustration.

As illustrated in Table 1, it can be concluded that the radio minimum is observed when

$$R(k) = 0.1 * 10^{-4} \times \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}_{M\times M}. \quad (23)$$

$Q(k)$ indicates the variance of Gaussian noise $W(k)$. In our work, $Q(k)$ is also determined through a numerical test. At first, $Q_0(k)$ is initialized with the elements on its diagonal at order of 10^{-4} [37]; that is,

$$Q_0 = 10^{-4} \times \operatorname{diag}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}_{(M+1)\times(M+1)}. \quad (24)$$

Then, we choose $Q(k)$ as $0.1 * Q_0$, Q_0, $10 * Q_0$, $100 * Q_0$, and $1000 * Q_0$ to test and the results are listed in Table 2.

It can be noticed from Table 2 that, for the same state transition process, the bigger the $Q(k)$, the bigger the disturbance to the state variable. At the same time, when $Q(k)$ begins to increase, the speed of the convergence of $Q(k)$ becomes slow. As a result, we choose $Q(k) = 0.1 * 10^{-4} \times \operatorname{diag}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}_{M\times M}$ in our work.

Using LQG benchmark, the achievable performance of $J(k)$ is given by a tradeoff curve between $\operatorname{var}(Y(k))$ and $\operatorname{var}(U(k))$ as shown in Figure 5 and this curve can be further obtained by solving the LQG problem [38].

Table 1: Kalman filtering error table with different matrix $R(k)$.

Parameters for filtering $R(k)$	Predicted value $\widehat{X(k)}$	The ratio of predicted value to the measured value
$0.1 * 10^{-4}$	30.96	41.25%
$1 * 10^{-4}$	23.81	31.72%
$5 * 10^{-4}$	34.60	46.10%
$10 * 10^{-4}$	53.54	71.34%
$20 * 10^{-4}$	93.97	95.20%
Measured value $Y(k)$	75.06	100%

Table 2: Kalman filtering error table with different matrix $Q(k)$.

Parameters for filtering $Q(k)$	$e(k) = Y_r(k) - Y(k)$	The ratio of $e(k)$ to $Y_r(k)$
$0.1 * Q_0$	16.68	39.69%
Q_0	19.87	47.26%
$10 * Q_0$	25.04	59.57%
$100 * Q_0$	33.58	79.89%
$1000 * Q_0$	39.92	94.97%
Expected value $Y_r(k)$	42.04	100%

The LQG problem is set up to control the system by keeping the output error small via small control energy. Therefore, our problem can be formulated as the problem of minimizing the cost function to work out the optimal feedback control rule $U^*(k)$, that is, making $Y(k)$ equal to $Y_r(k)$.

With reference to [39], definition 1, the LQG system is said to be stable in the mean square sense if all initial states $X(0)$ yield

$$\lim_{k\to\infty} E\left(X(k)X(k)^T\right) = 0. \quad (25)$$

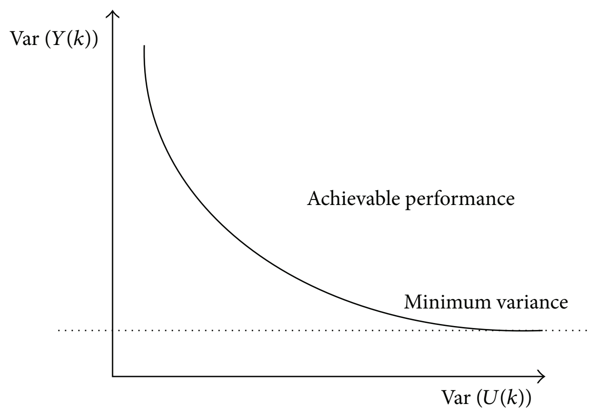


Figure 5: Optimal performance curve obtained through LQG benchmark.

The LQG system is said to be stabilizable in the mean square sense if there exists a matrix $S(k)$ such that

$$J(U(k)) = E\left(\sum e^T(k)\,Q(k)\,e(k) + U^T(k)\,R(k)\,U(k)\right) \quad (26)$$

is stable in the mean square sense. We use the concept of stability here as what would in other contexts usually be called asymptotic stability. Note that stability in the mean square sense implies stability of the mean (i.e., $\lim_{k\to\infty}E\{X(k)\}=0$) for all $X(0)$ and almost sure stability (i.e., $\lim_{k\to\infty}X(k)=0$, for all $X(k)$). In particular, this implies that stability of the deterministic system

$$X(k+1) = AX(k) + BU(k), \quad (27)$$

is a necessary condition for mean square stability of LQG system. (This condition is satisfied if (27) is, say, controllable [40].)

Finally, according to the framework of LQG in Figure 4, our optimal controller could be expressed as (28). With reference to [38], the LQG controller computing an optimal state feedback control law by minimizing the quadratic cost function J can be expressed as follows:

$$U^*(k) = -S(k)\,\widehat{X(k)}, \quad (28)$$

where $\widehat{X(k)}$ denotes the predicted throughput with Kalman filtering and $S(k)$ is the optimal feedback gain matrix which can be solved via MATLAB LQ (linear quantifier) function. According to Figure 4, when the state estimation $\widehat{X(k)}$ is obtained from the Kalman filter, the optimal control vector $U^*(k)$ will be obtained by $\widehat{X(k)}$ multiplied by $S(k)$. The optimal state feedback gain matrix $S(k)$ is available from solving the associated discrete algebraic Riccati equation [41]; that is,

$$S(k) = -\left(R(k) + B^T Q_{k+1}(k)\,B\right)^{-1} B^T Q_{k+1}(k)\,A, \quad \forall k \in \{0,\ldots,T-1\}. \quad (29)$$

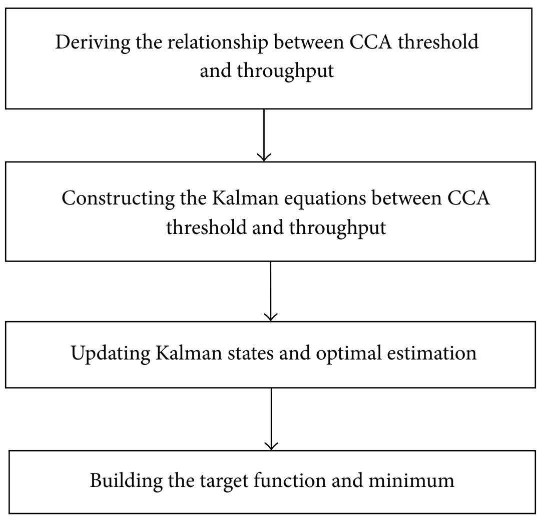

Deriving the relationship between CCA threshold and throughput
Constructing the Kalman equations between CCA threshold and throughput
Updating Kalman states and optimal estimation
Building the target function and minimum

Figure 6: An LQG optimized information congestion control scheme.

So the corresponding minimum cost is

$$J\left(U^*(k)\right) = \begin{bmatrix} X(k) \\ U^*(k) \end{bmatrix}^T \begin{bmatrix} Q(k) & S(k) \\ S^T(k) & R(k) \end{bmatrix} \begin{bmatrix} X(k) \\ U^*(k) \end{bmatrix}. \quad (30)$$

When the optimal CCA threshold CCA $= U^*(k)$ is obtained, we can figure out the optimal throughput of all vehicles as follows:

$$\text{Throughput}(k) = \sum_i \text{thr}_i(k). \quad (31)$$

Finally, our proposed scheme in this paper can be summarized as a flow chart as shown in Figure 6.

4. Simulation

The simulation in this paper is performed in a Linux system which combines MATLAB with NS2. The topology is created with the popular Vanetmobisim toolset [42] as shown in Figure 7. The street layouts used for simulation, that is, the selected area of Washington DC, are loaded from TIGER database. The complexity of the selected area of Washington DC is 42, 80, and 125; that is, there are 42 junctions and 80 streets and the average length of streets is 125 meters. The labels with numbers in Figure 7(b) indicate different vehicles and they are a little bit overlapped due to limited screen space especially when they are stopped by traffic lights. The lines represent avenues or streets. Since traffic lights are enabled in simulation, the colored line means there is a traffic light on the junction. The red color line indicates that the traffic on this line is stopped by a red light and vice versa. Note that the configuration of the position and number of traffic lights are not the real case, but it can be adjusted during simulations to reflect the practical situation. In our cases, we set the number of traffic lights to 10. In addition, although TIGER can describe land attributes such as roads, buildings, rivers, and lakes, it is still difficult to draw obstacles on output traces by Vanetmobisim till now. However, to reflect the influences from obstacles which are common in urban environment, we extracted the coordinates of obstacles from the investigated parts of real maps and input them into NS2. Besides, since there is no height information in TIGER database, a modification to NS2 is needed to reflect the impact of obstacles on channel fading and power attenuation. To support obstacle modeling, a two-dimension obstacle object "Obstacle Class" is introduced which represents a wall of 1 meter deep and has the length indicated by the distance between two coordinates extracted from the real maps, that is, $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$. By this way, a building could be expressed by four connected walls. When the line of sight (LOS) of a communication pair intersects with the outline of the building, the power attenuation could be calculated by the following equation, which combines the generic free space path loss model with the obstacle model presented in [43]; that is,

$$P_r\ [\text{dBm}] = P_t\ [\text{dBm}] + 10\log\left(\frac{G_t G_r \lambda^2}{16\pi^2 d^\alpha}\right) - \beta n - \gamma d_m, \tag{32}$$

where $P_r, P_t, G_t, G_r, \lambda, d$ are the receiving power, transmitting power, sender antenna gain, receiver antenna gain, wave length, and the distance between sender and receiver, respectively. n is the number of times that the border of the obstacle is intersected by the line of sight. d_m here is the total length of the obstacle's intersection. β and γ are two constants. β is given in dB per wall and represents the attenuation a transmission experiences due to the (e.g., brick) exterior wall of a building. γ is given in dB per meter and serves as a rough approximation of the internal structure of a building. The general values of β and γ in most cases are 9 dB and 0.4 dB/m, respectively.

The performance of our model is evaluated by varying the density, the packet generation speed, and the intervehicle distance according to the following metrics:

(1) Average throughput, defined as the average number of successfully transmitted payloads per second for the overall network.

(2) Average End-to-End delay (average E2E delay), defined as the delay averaged among all the transmitted packets by all the active vehicles during the simulation period.

(3) Average packets delivery ratio (average PDR), defined as the average ratio of the packets successfully delivered to the destinations with respect to those generated by the sources during the simulation period.

In our simulated highway scenario, all vehicles keep sending and receiving periodic beacon messages. The maximum radio range of each vehicle is set to 1,000 m. Actually, according to the IEEE 802.11p, the fast-moving vehicles can cover a transmission range over 1,000 m [44]. Our scenario is envisioned comprising 2 lanes and 10 km long in total, where there are 50 (at least) to 400 (at most) vehicles in two directions and the vehicle moves using the IDM_LC (intelligent driver model with lane changing) mobility model with the parameters listed in Table 4. Detailed simulation parameters are listed in Table 3. The frequency band is set to 5.92 GHz which is configured to serve for the "High Power Public Safety" services in IEEE 802.11p. The channel data rate is chosen to be the lowest rate supported by IEEE 802.11p, namely, 3 Mbps [45]. The Nakagami radio propagation model, whose parameters were adjusted to match actual measurements reported in [46], has been used. The Access Class is configured to AC_VO, say, the highest priority for voice transmission. This is because the packets exchanged in our discussed scenario are safety-related where the highest priority is necessary to guarantee their exclusive transmissions. The beacon size is set to 500 bytes [47]. The propagation delay is fixed to 1 μs for simplicity. The simulation duration is 50 seconds and all the results are the average of 100 runs.

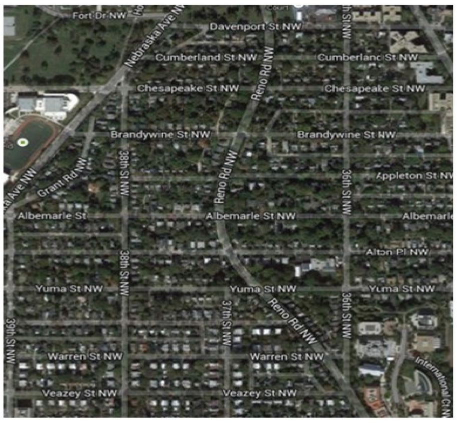
Table 3: Simulation parameters.

Parameters	Values
Frequency band (GHz)	5.92
Channel data rate (Mbps)	3
Transmission model	Nakagami
Maximum communication range (m)	1000
SNR threshold for frame reception (dB)	5
Slot time (μs)	16
AIFS (μs)	9
Access class	AC_VO
$aCW_{\min}$	15
$aCW_{\max}$	1023
$CW_{\min}$(AC_VO)	$(aCW_{\min}+1)/4-1$
$CW_{\max}$(AC_VO)	$(aCW_{\min}+1)/2-1$
Beacon size (bytes)	500
Total road length (Km)	10
Propagation delay (μs)	1
Number of traffic lights	10
Simulation time (s)	50

Table 4: Scenario generation parameters for IDM_LC.

Description	Value
Traffic light interval (s)	3
Min speed (m/s)	6.66
Recalculating movement step (s)	1
Number of lanes	2
Max speed (m/s)	24.44
Min stay (s)–max stay (s)	5–30

To evaluate the performance of our proposed LQG-CCA algorithm, two other protocols, that is, D-FPAV and APPR, are introduced for comparisons. Note that D-FPAV is a highly cited work in the vehicular communication research filed to address the channel congestion issue through transmission power control to improve the network performance in terms of dissemination delay and amount of message retransmissions. Similarly, our LQG-CCA also tends to improve the


(a) Satellite snapshot

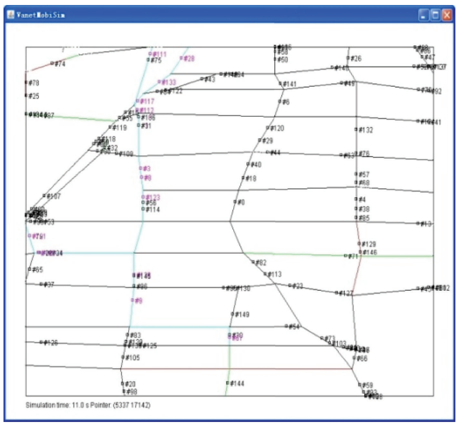

(b) Topology generated by Vanetmobisim

Figure 7: Simulation topology.

network performance by dynamically adjusting the CCA threshold for each node to maximize the network throughput whereas not making the channel congested. In addition, D-FPAV and LQG-CCA are both designed for time-critical safety-related events in a vehicular environment. As a result, we consider D-FPAV to be a better candidate for performance comparisons with our proposed LQG-CCA. As for APPR, it is also a channel congestion reduction scheme by varying the length of beacon message, controlling the transmission power and message generation rate, and removing the duplicate messages from the message queue. Additionally, APPR is also applicable to the safety-related scenarios in which an emergency message or a warning message will be propagated according to the aforementioned congestion control strategies. Note that, in our work, only APPR_II is implemented for comparison since it is more reasonable in practice which attempts to increase the channel load till the percentage of message loss lies below a predefined threshold. Instead, APPR_I just maintains the channel load to a fixed value depending upon the message generation rate of different vehicles in a network. The important parameter in APPR_II, that is, the threshold of the message loss ratio, is set to 2% according to [16].

Figure 8 shows the network average throughput performance with the number of nodes varying from 50 to 400. The packets generation speed for this case is 5 packets/s. It is worth noting that our LQG-CCA always outperforms the other two in this case. Before point 200, the average throughput of LQG-CCA continuously increases. And after 200, there is a slow decrease of average throughput for LQG-CCA and finally it outputs approximately 2.3 Mbps which is even bigger than the throughput corresponding to the 50 vehicles. At first, this result indicates that the impact of the growth of senders on the throughput is larger than that of the more introduced collisions from more vehicles. This conclusion is also valid for APPR. However, D-FPAV shows a lower throughput at point 400 compared to that at point 50. The reason behind this is the transmission restraining mechanism adopted by LQG-CCA and APPR. For LQG-CCA, by dynamically computing the appropriate CCA threshold for each active vehicle, the throughput could be maximized by forcing some vehicles to stop sending according to the channel load and contention level. In APPR, it also will stop the generation of some messages from vehicles if the packets loss ratio exceeds a predefined threshold. In this way, more transmission opportunities could be protected by suppressing extra interferences. However, in D-FPAV, the research emphasis falls on how to achieve the fairness of transmission opportunities among different active nodes. Every node starts with the minimum transmitting power assigned and then increases their transmitting power simultaneously with the same amount as long as the condition on the beaconing network load (MBL) is satisfied. In this

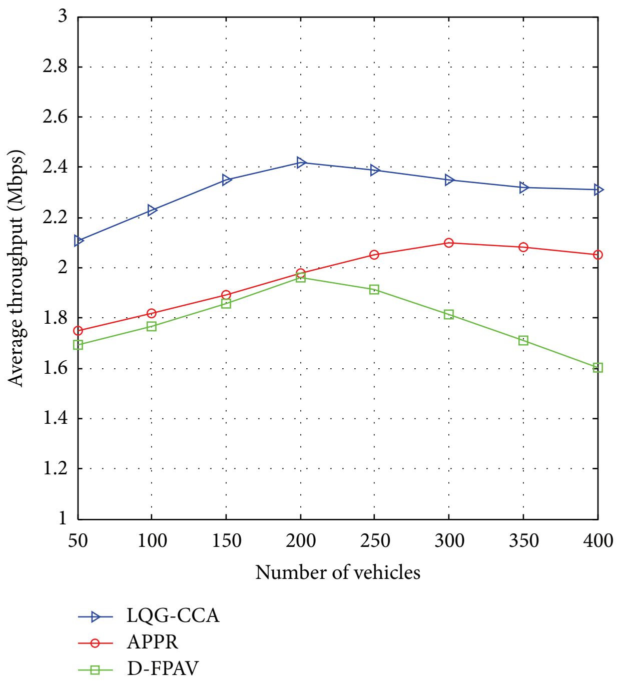


Figure 8: Average throughput versus number of vehicles when the packets generation speed is 5 packets/s.

way, the fairness is guaranteed but the total throughput in a network is damaged. As a result, D-FPAV outputs a lower average throughput when there are too many nodes simultaneously increasing their sending power, by which mutual interferences are also increased thus leading to a lower transmission opportunity for each node. In addition, the superiority of our LQG-CCA over APPR and D-FPAV is actually attributed to the distributed interference cancellation mechanism. In this way, the global throughput could be maximized by transmission opportunities assignment among nodes locally.

The performance of average E2E delay is shown in Figure 9 with the number of nodes varying from 50 to 400. Note that the safety-related events are given the highest sending priority, that is, AC_VO, with smaller contention window and interframe space. It can be concluded that all three protocols meet the delay requirements for safety-related events according to "TABLE I" in [48], where a maximum 100 ms latency is required for both periodical and emergency warning messages. Since a larger density of vehicles will definitely increase the average E2E delay considering the packets collisions or transmission restraining, extra time is needed to finish the transmission attempts of active nodes. Our LQG-CCA also outperforms the other two protocols with a maximum latency approximately 5 ms at point 400. The APPR ranks second while the D-FPAV is the worst. In addition, by checking the slope of three curves, it can be noticed that D-FPAV shows a faster growth of delay with the density increasing compared to the other two. This is actually because D-FPAV will easily saturate the channel load when there are too many nodes concurrently increasing their transmitting power. On the other hand, due to adaptive control of transmission opportunities or packet generation speed, LQG-CCA and APPR both show a more flat curve reflecting their better adaptability to the network load.

Figure 10 shows the average PDR of three algorithms with the number of nodes varying from 50 to 400. Consistent with Figures 8 and 9, our LQG-CCA also ranks first considering its ability to adaptively adjust the medium sensing threshold thus leading to a better PDR. Note that the restrained nodes are not taken into account in the computation of PDR since they are no longer active. With the vehicular density increase, the average PDR of three protocols all drop. By checking the NS2 trace file, there are lots of "DROP_MAC_COLLISION", "DROP_MAC_BUSY", and "DROP_MAC_RETRY_COUNT_EXCEEDED" occurring; that is, the packets dropped due to collisions, channel being busy, and exceeding the retry limit, respectively. Therefore, we could say that the packets retrying and backing off contribute to the PDR falling when the density is increasing. However, since intelligent transmission scheduling is enabled in LQG-CCA and APPR, their PDR dropping is relatively slight with a minimum of 0.6 compared to D-FPAV which has a minimum PDR below 0.5. In addition, the smaller slope of LQG-CCA implies its better load adaptive capability compared to the other two.

The impact of packet generation speed on the average PDR is shown in Figure 11 with the packet generation speed varying from 0 to 10 packets/s. Note that, in this simulation case, the number of nodes is fixed to 100, say, a relatively slight density considering the total 10 km long road segment. With the packet generation speed increases, all three protocols show decreases of packets delivery ratio. This result is reasonable as a larger data sending rate will readily saturate the channel and bring more packets collisions thus

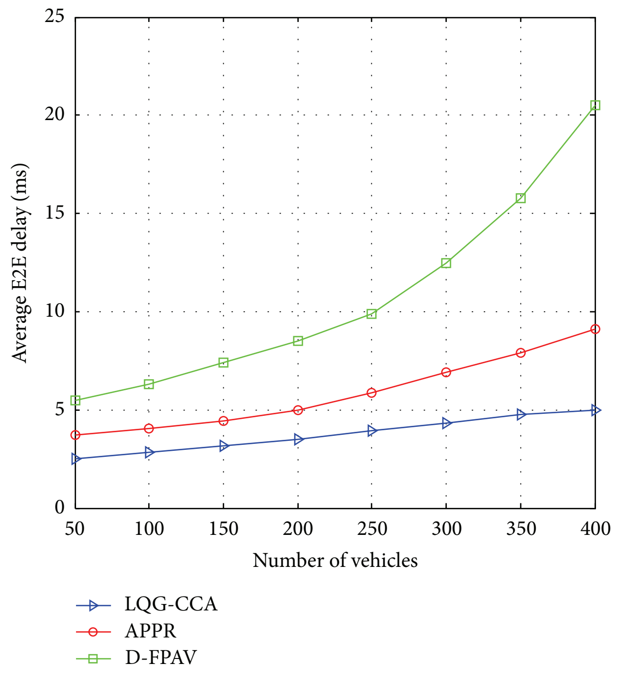


FIGURE 9: Average E2E delay versus number of vehicles when the packets generation speed is 5 packets/s.

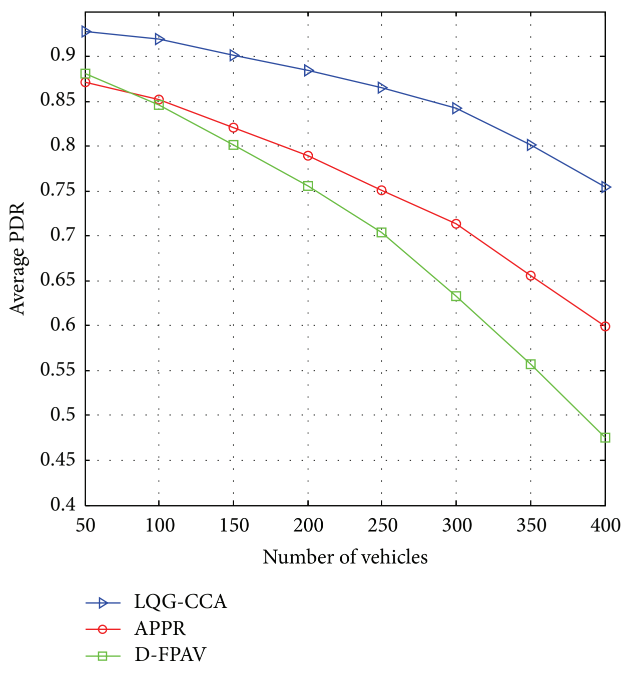


FIGURE 10: Average PDR versus number of vehicles when the packets generation speed is 5 packets/s.

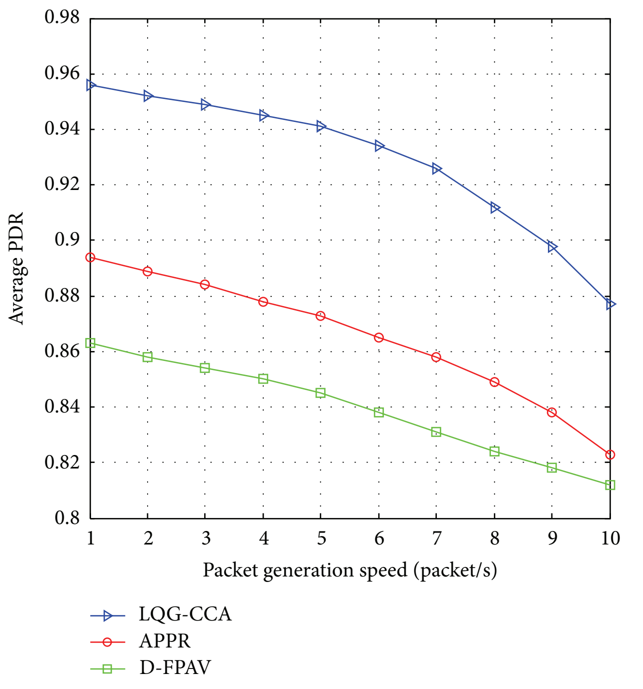


FIGURE 11: Average PDR versus packet generation speed when there are 100 nodes.

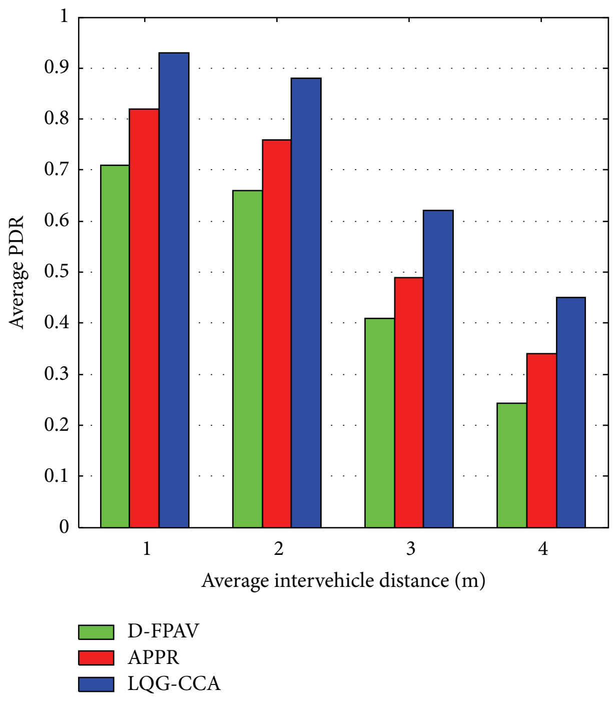


FIGURE 12: Average PDR versus average intervehicle distance when there are 100 nodes.

reducing the packets delivery ratio. What is noteworthy is that both LQG-CCA and APPR introduced the transmitting restraining scheme which will adaptively schedule the packets sending instead of just sending them out upon generation. As a result, although the packet generation speed is continuously increasing, the decreasing speed of three protocols is quite different according to their various medium access manners. For D-FPAV, there is no strategy to counteract the adverse impact on channel congestion from packet generation speed growing. Therefore, D-FPAV shows the worst performance consistent with previous numerical results. To compare APPR with our LQG-CCA, since there are no traffic predictions in APPR, their packet generation speed control mechanism will only take effect after the receiving of the feedback regarding a congestion that already occurred. In this way, our LQG-CCA will act more efficiently with accurate estimations of the channel state thus leading to a better average PDR.

The impact of average intervehicle distance on the average PDR is shown in Figure 12. Note that although a 1000-meter radio range is assumed, the packets are actually difficult to be received by their destinations 1000 meters away considering the channel fading and obstacles which are common in the urban environment such as skyscrapers, big trucks, and giant trees. As a matter of fact, the wireless signal will be attenuated by the equivalent walls according to our introduced obstacle model expressed by (31). To control the intervehicle distance, one parameter in IDM_LC, that is, the desired dynamical distance [49], has been changed to generate a required average intervehicle distance in the network. It is noteworthy that, with the increasing of average intervehicle distance, the average PDR drops quickly. This is because a larger intervehicle distance will bring more risks for the signal to be sheltered by the obstacles. For our LQG-CCA, because its throughput maximization procedure considers the interferences from others using the SINR reception model, its performance degradation is the smallest compared to the other two. Since the reception model in APPR and D-FPAV is not given explicitly, we just determine whether a packet is successfully received according to the SNR threshold for frame reception listed in Table 3. However, due to the generation speed control according to the packet loss ratio in APPR, its performance is better than D-FPAV. As for D-FPAV, which puts its emphasis on access fairness rather than channel congestion and total throughput, it performs the worst among three models.

5. Conclusion

In this paper, we propose an LQG-based congestion control scheme in vehicular networks. Our model can dynamically adjust the CCA threshold according to the difference between the actual network throughput and the expected network throughput. In this way, the determined CCA threshold for each node can maximize the network throughput without leading to channel congestion at the same time. Numerical results show that our LQG optimized congestion control algorithm can improve the network throughput and packets delivery ratio and can reduce packet transmission delay significantly. Our future work will attempt to efficiently utilize the capture effect to make concurrent reception available on hardware and model the CCA threshold adjustable transmission model with MU-MIMO (Multiuser Multiple Input Multiple Output) enabled.

Nomenclature

Variables

$X(k)$:	Predicted throughput
A:	Coefficient matrix of the state variable
B:	Coefficient matrix of the control variable
$U(k)$:	The control variable
W:	The system noise
$Y(k)$:	The actual measured throughput
C:	The coefficient matrix of measured variable
V:	The measured noise
Q:	The covariance of noise
R:	The weight of state variable
$X(k \mid k-1)$:	The predicted throughput at time k
$X(k-1 \mid k-1)$:	The throughput at time $k-1$
$Z(k \mid k-1)$:	The covariance of predicted throughput
$Z(k-1 \mid k-1)$:	The covariance of throughput at time $k-1$
$\widehat{X(k)}$:	The estimated throughput at time k
$Z(k \mid k)$:	The covariance of estimated throughput
$Y_r(k)$:	The expected throughput
$G(k)$:	The Kalman gain at time k
H:	The covariance of measured noise
$e(k)$:	The deviation of throughput from the actual value
J:	The objective function of quadratic form
$U^*(k)$:	The optimal control variable
$S(k)$:	The optimal output feedback gain matrix.

Notations

I_s:	The received signal strength
P:	The transmission power
r:	The distance between the transmitter and receiver
θ:	The path loss coefficient
CCA_{th}:	The Clear Channel Assessment sensing threshold
R:	The maximum transmission range
W:	The channel bandwidth
n:	The number of concurrent transmissions
Area:	The network range.

Competing Interests

The authors declare that they have no conflict of interests regarding the publication of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61201133 and 61571338), the National Science and Technology Major Project of the Ministry of Science and Technology of China (2015zx03002006-003), the Natural Science Foundation of Shaanxi Province (2014JM2-6089), the National High-Tech R&D Program of China (863 Program, 2015AA015701), the Research Collaboration Innovation Program of Xi'an (BD31015010001), the Hong Kong, Macao and Taiwan Science and Technology Cooperation Special Project (2014DFT10320 and 2015DFT10160), the Program for New Century Excellent Talents in University (NCET-11-0691), state grid cooperation project (HX0115014015), the Key Program of NSFC-Guangdong Union Foundation (U1401251), and the "111 Project" of China (B08038).

References

[1] D. Gruyer, S. Demmel, B. D'Andrea-Novel, A. Lambert, and A. Rakotonirainy, "Simulation architecture for the design of cooperative collision warning systems," in *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems (ITSC '12)*, pp. 697–703, IEEE, Anchorage, Alaska, USA, September 2012.

[2] Y.-K. Park, Y.-J. Moon, Y.-S. Cho, and K.-J. Kum, "Field tests for evaluating cooperative intersection signal violation warning system (CISVWS)," *International Journal of Automotive Technology*, vol. 14, no. 2, pp. 275–281, 2013.

[3] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *Proceedings of the 25th IEEE International Conference on Computer Communications (IEEE INFOCOM '06)*, pp. 1–13, Barcelona, Spain, April 2006.

[4] L. Chen, S. H. Low, and J. C. Doyle, "Joint congestion control and media access control design for ad hoc wireless networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '05)*, vol. 3, pp. 2212–2222, Miami, Fla, USA, March 2005.

[5] C. Lochert, B. Scheuermann, and M. Mauve, "A survey on congestion control for mobile ad hoc networks," *Wireless Communications and Mobile Computing*, vol. 7, no. 5, pp. 655–676, 2007.

[6] K. Kamini and R. Kumar, "VANET parameters and applications: a review," *Global Journal of Computer Science and Technology*, vol. 10, no. 7, pp. 72–77, 2010.

[7] J. Zhu, B. Metzler, X. Guo, and Y. Liu, "Adaptive CSMA for scalable network capacity in high-density WLAN: a hardware prototyping approach," in *Proceedings of the INFOCOM 25th IEEE International Conference on Computer Communications*, pp. 1–10, Barcelona, Spain, April 2006.

[8] F. Xia, H. Bin Liaqat, A. M. Ahmed et al., "User popularity-based packet scheduling for congestion control in ad-hoc social networks," *Journal of Computer & System Sciences*, vol. 82, no. 1, pp. 93–112, 2016.

[9] W. Guan, J. He, L. Bai, and Z. Tang, "Adaptive congestion control of DSRC vehicle networks for collaborative road safety applications," in *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks (LCN '11)*, pp. 913–917, IEEE, Bonn, Germany, October 2011.

[10] S. Wang, L. Le, N. Zahariev, and K. K. Leung, "Centralized rate control mechanism for cellular-based vehicular networks," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM '13)*, pp. 4914–4920, IEEE, Atlanta, Ga, USA, December 2013.

[11] L. Wischhof and H. Rohling, "Congestion control in vehicular ad hoc networks," in *Proceedings of the IEEE International Conference on Vehicular Electronics and Safety*, pp. 58–63, October 2005.

[12] M. S. Bouassida and M. Shawky, "A cooperative and fully-distributed congestion control approach within VANETs," in *Proceedings of the 9th International Conference on Intelligent Transport Systems Telecommunications (ITST '09)*, pp. 526–531, IEEE, Lille, France, October 2009.

[13] L. Zhou, B. Zheng, B. Geller, A. Wei, S. Xu, and Y. Li, "Cross-layer rate control, medium access control and routing design in cooperative VANET," *Computer Communications*, vol. 31, no. 12, pp. 2870–2882, 2008.

[14] M. Torrent-Moreno, P. Santi, and H. Hartenstein, "Distributed fair transmit power adjustment for vehicular ad hoc networks," in *Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad hoc Communications and Networks (SECON '06)*, pp. 479–488, Reston, Va, USA, September 2006.

[15] W.-J. Guo, L.-S. Huang, Q. Sun, H.-L. Xu, and H.-R. Zhang, "Delay-aware reliable broadcast scheme based on power control for VANETs," *The Journal of China Universities of Posts and Telecommunications*, vol. 21, no. 1, pp. 26–35, 2014.

[16] S. Mitra and A. Mondal, "Joint congestion control strategy during V2V communication among authentic vehicles in VANET," *Wireless Personal Communications*, vol. 79, no. 1, pp. 43–67, 2014.

[17] C. Sommer, S. Joerer, M. Segata, O. K. Tonguz, R. L. Cigno, and F. Dressler, "How shadowing hurts vehicular communications and how dynamic beaconing can help," *IEEE Transactions on Mobile Computing*, vol. 14, no. 7, pp. 1411–1421, 2015.

[18] S. Bai, J. Oh, and J.-I. Jung, "Context awareness beacon scheduling scheme for congestion control in vehicle to vehicle safety communication," *Ad Hoc Networks*, vol. 11, no. 7, pp. 2049–2058, 2013.

[19] J. Sahoo, E. H.-K. Wu, P. K. Sahu, and M. Gerla, "Congestion-controlled-coordinator-based MAC for safety-critical message transmission in VANETs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1423–1437, 2013.

[20] N. Chaabouni, A. Hafid, and P. K. Sahu, "A collision-based beacon rate adaptation scheme (CBA) for VANETs," in *Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS '13)*, pp. 1–6, IEEE, Kattankulathur, India, December 2013.

[21] T. Tielert, D. Jiang, Q. Chen, L. Delgrossi, and H. Hartenstein, "Design methodology and evaluation of rate adaptation based congestion control for Vehicle Safety Communications," in *Proceedings of the IEEE Vehicular Networking Conference (VNC '11)*, pp. 116–123, Amsterdam, The Netherlands, November 2011.

[22] M. A. Javed and J. Y. Khan, "Performance analysis of an adaptive rate-range control algorithm for VANET safety applications," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC '14)*, pp. 418–423, IEEE, Honolulu, Hawaii, USA, February 2014.

[23] L. Zeng, L. Yang, Q. Han, X. He, L. Ye, and B. Yang, "Experimental analysis of CCA threshold self-adjusting method for EWM dissemination," in *Proceedings of the 17th IEEE International Conference on Intelligent Transportation Systems (ITSC '14)*, pp. 3040–3045, Qingdao, China, October 2014.

[24] Q. Han, Y. Liu, L. Yang, and L. Zeng, "Experimental analysis of multi-hop vehicle node CCA threshold selection for EWM transmission," in *Proceedings of the 2nd IEEE International Conference on Connected Vehicles and Expo (ICCVE '13)*, pp. 857–862, Las Vegas, Nev, USA, December 2013.

[25] B. Cho, K. Koufos, and R. Jantti, "Interference control in cognitive wireless networks by tuning the carrier sensing threshold," in *Proceedings of the 8th International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM '13)*, pp. 282–287, July 2013.

[26] R. K. Schmidt, A. Brakemeier, T. Leinmüller, F. Kargl, and G. Schäfer, "Advanced carrier sensing to resolve local channel congestion," in *Proceedings of the 8th ACM International Workshop on Vehicular Inter-Networking (VANET '11)*, pp. 11–20, 2011.

[27] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics (ICINCO '04)*, pp. 222–229, Setúbal, Portugal, August 2004.

[28] C. L. Black, "The lawfulness of the segregation decisions," *The Yale Law Journal*, vol. 69, no. 3, pp. 421–430, 1960.

[29] L. Zhang, H. Gao, and O. Kaynak, "Network-induced constraints in networked control systems—a survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 403–416, 2013.

[30] T. Schuhmann, W. Hofmann, and R. Werner, "Improving operational performance of active magnetic bearings using Kalman filter and state feedback control," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 821–829, 2012.

[31] K. You, N. Xiao, and L. Xie, "LQG control with quantized innovation Kalman filter," in *Analysis and Design of Networked Control Systems*, pp. 205–221, Springer, London, UK, 2015.

[32] S. U. Eichler, "Performance evaluation of the IEEE 802.11p WAVE communication standard," in *Proceedings of the IEEE 66th Vehicular Technology Conference (VTC-Fall '07)*, pp. 2199–2203, Baltimore, Md, USA, October 2007.

[33] X. Yang and N. Vaidya, "On physical carrier sensing in wireless ad hoc networks," in *Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 2525–2535, Miami, Fla, USA, March 2005.

[34] J. Q. Bao and L. Tong, "A performance comparison between ad hoc and centrally controlled CDMA wireless LANs," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 829–841, 2002.

[35] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Approximate distributed kalman filtering in sensor networks with quantifiable performance," in *Proceedings of 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pp. 133–139, April 2005.

[36] G. Bianchi, "IEEE 802.11-saturation throughput analysis," *IEEE Communications Letters*, vol. 2, no. 12, pp. 318–320, 1998.

[37] V. A. Bavdekar, A. P. Deshpande, and S. C. Patwardhan, "Identification of process and measurement noise covariance for state and parameter estimation using extended Kalman filter," *Journal of Process Control*, vol. 21, no. 4, pp. 585–601, 2011.

[38] Z. Chao, S. Hongye, G. Yong, and C. Jian, "A pragmatic approach for assessing the economic performance of model predictive control systems and its industrial application," *Chinese Journal of Chemical Engineering*, vol. 17, no. 2, pp. 241–250, 2009.

[39] J. L. Willems and J. C. Willems, "Feedback stabilizability for stochastic systems with state and control dependent noise," *Automatica*, vol. 12, no. 3, pp. 277–283, 1976.

[40] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, vol. 1, Wiley-Interscience, New York, NY, USA, 1972.

[41] P. C. Young and J. C. Willems, "An approach to the linear multivariable servomechanism problem," *International Journal of Control*, vol. 15, pp. 961–979, 1972.

[42] J. Härri, F. Filali, C. Bonnet, and M. Fiore, "VanetMobiSim: generating realistic mobility patterns for VANETs," in *Proceedings*

of the 3rd International Workshop on Vehicular ad hoc Networks (VANET '06), pp. 96–97, Los Angeles, Calif, USA, September 2007.

[43] C. Sommer, D. Eckhoff, R. German, and F. Dressler, "A computationally inexpensive empirical model of IEEE 802.11p radio shadowing in urban environments," in *Proceedings of the 8th International Conference on Wireless On-Demand Network Systems and Services (WONS '11)*, pp. 84–90, Bardonecchia, Italy, January 2011.

[44] V. D. Khairnar and K. Kotecha, "Performance of vehicle-to-vehicle communication using IEEE 802.11p in vehicular ad-hoc network environment," *International Journal of Network Security & Its Applications*, vol. 5, no. 2, pp. 143–170, 2013.

[45] K. Bilstrup, E. Uhlemann, E. G. Strom, and U. Bilstrup, "Evaluation of the IEEE 802.11p MAC method for vehicle-to-vehicle communication," in *Proceedings of the IEEE 68th Vehicular Technology Conference (VTC-Fall '08)*, pp. 1–5, Calgary, Canada, September 2008.

[46] V. Taliwal, D. Jiang, H. Mangold, C. Chen, and R. Sengupta, "Empirical determination of channel characteristics for DSRC vehicle-to-vehicle communication," in *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, p. 88, Philadelphia, Pa, USA, October 2004.

[47] S. Eichler, "Performance evaluation of the IEEE 802.11p WAVE communication standard," in *Proceedings of the IEEE 66th Vehicular Technology Conference (VTC '07)*, pp. 2199–2203, IEEE, Baltimore, Md, USA, October 2007.

[48] E. C. Eze, S. Zhang, and E. Liu, "Vehicular ad hoc networks (VANETs): current state, challenges, potentials and way forward," in *Proceedings of the 20th International Conference on Automation and Computing (ICAC '14)*, pp. 176–181, Cranfield, UK, September 2014.

[49] M. Fiore, J. Harri, F. Filali, and C. Bonnet, "Vehicular mobility simulation for VANETs," in *Proceedings of the 40th Annual Simulation Symposium (ANSS '07)*, pp. 301–309, Norfolk, Va, USA, March 2006.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 2097243, 10 pages
http://dx.doi.org/10.1155/2016/2097243

Research Article

Representation Learning from Time Labelled Heterogeneous Data for Mobile Crowdsensing

Chunmei Ma,[1] Qing Zhu,[2] Shuang Wu,[3] and Bin Liu[2]

[1]*School of Computer and Information Engineering, Tianjin Normal University, Tianjin, China*
[2]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China*
[3]*College of Computer Science, Zhejiang University, Hangzhou, China*

Correspondence should be addressed to Chunmei Ma; mcmxhd@163.com

Received 5 May 2016; Revised 18 July 2016; Accepted 4 August 2016

Academic Editor: Francisco Martínez

Mobile crowdsensing is a new paradigm that can utilize pervasive smartphones to collect and analyze data to benefit users. However, sensory data gathered by smartphone usually involves different data types because of different granularity and multiple sensor sources. Besides, the data are also time labelled. The heterogeneous and time sequential data raise new challenges for data analyzing. Some existing solutions try to learn each type of data one by one and analyze them separately without considering time information. In addition, the traditional methods also have to determine phone orientation because some sensors equipped in smartphone are orientation related. In this paper, we think that a combination of multiple sensors can represent an invariant feature for a crowdsensing context. Therefore, we propose a new representation learning method of heterogeneous data with time labels to extract typical features using deep learning. We evaluate that our proposed method can adapt data generated by different orientations effectively. Furthermore, we test the performance of the proposed method by recognizing two group mobile activities, walking/cycling and driving/bus with smartphone sensors. It achieves precisions of 98.6% and 93.7% in distinguishing cycling from walking and bus from driving, respectively.

1. Introduction

The smartphone has become extremely popular recently. The development of the smartphone with various sensors and powerful capabilities (computing, storage, and communication) motivates a popular computing and sensing paradigm, *crowdsensing*. As a result of the explosion of sensor-equipped mobile phones, we can sense the environment, infrastructure, and even social activities [1]. For example, in [2], the authors proposed using a smartphone with a built-in triaxial accelerometer to recognize physical activities, which can provide valuable information regarding an individual's degree of functional ability and life style. Besides using a single type sensor of smartphones, most often, we use multitypes sensors of the smartphone to obtain more comprehensive sensory data for a variety of applications [3, 4]. However, the sensory data from various data sources are usually heterogeneous, representing different granularity and diverse quality. In addition, the data are usually time labelled. Because of the two characteristics of sensory data, how to "understand" heterogeneous data with time labels correctly becomes a new challenge for data analyzing.

Some existing solutions would prefer to analyze a single type of data sensor by sensor [4–6]. For instance, in [5], authors focus specifically on traffic monitoring by using accelerometer, microphone, GSM radio, and/or GPS sensors of the smartphone to detect potholes, bumps, braking, and honking. They separately analyze data generated from each of these sensors one by one. The disadvantage of these methods is that they can only obtain a unidimensional characteristic of sensory data. Some other researchers proposed sensor fusion methods to learn the sensory data [3, 7]. This is accomplished by a feature extraction approach in which features from each sensor are computed independently. Then, the extracted features are integrated for fusion of information from multisensors. Although these methods derive comprehensive characteristics of sensory data, they cannot denote the internal relations of the heterogeneous data. Furthermore, all of these methods never consider time labels of sensory data, which could lead to some typical features being neglected. For this reason, some works try to learn the time characteristics of sensory data using Hidden Markov Model (HMM) [8, 9]. But HMM-based algorithm can only obtain features of neighboring time points of sensory data rather than the overall time features.

Due to the limitations of existing methods of dealing with sensory data separately, we propose a new *representation learning* method of heterogeneous data with time labels to extract typical features using deep learning. In our model, multitype sensors are set to the same sampling frequency. Then, the sensory data are labelled by "sequence tag" according to sequencing the collection of data. Thus, the collected data and their sequence tags can be combined together as an integral feature. Then, such global data integration can be accepted as input by deep learning network. With multiple layers, deep learning is more powerful and flexible. It is able to combine many layers to generate an integrated feature. In crowdsensing, we believe that the integrated feature abstracted from multiple sensory data can well recognize a corresponding context. Besides, due to the sensory data tagged by time labels, we can learn the temporal knowledge from raw data as well in our model. In a word, we not only integrate heterogeneous data from multiple sensors, but also combine it with temporal information. We named the combination as *context fingerprint*.

In this paper, we propose and demonstrate our method to analyze sensory data in an overall view. We group all data collected at time t_0 from multiple sensor sources and their sequence tag T_0 as a vector. Suppose the vector generated at time t_0 is denoted by $\mathbf{x}_1'$ and the length of sampling window is τ. Then, we can get a vector $\mathbf{x}_2'$ with sequence tag T_1 at time $t_0 + \tau$ in the same way. Repeating this sampling process for n times, we can get the sample $\mathbf{X}'$, where $\mathbf{X}' = (\mathbf{x}_1', \mathbf{x}_2', \dots, \mathbf{x}_n')$ is a matrix with n columns. Since the sensory data are from different granularity data sources, it is necessary to refine the raw sample $\mathbf{X}'$ by data preprocessing. With the preprocessing, we get same size sample $\mathbf{X}$ from $\mathbf{X}'$. The sample $\mathbf{X}$ will be set as an input for our deep learning model. Experimental results show that the context fingerprint reconstructed by deep learning can efficiently represent an invariant feature for a crowdsensing context. Our proposed method has the following innovative features: (i) it integrates the overall feature (*context fingerprint*) of raw data as input, (ii) it captures and learns, in addition to sensory data itself, the tagged time information of data and utilizes both of them for context inferring, and (iii) with the deep learning model, we do not have to do orientation correction; in other words, we need not care about the problem of phone orientation.

The main contributions of this paper are multifold, which include the following:

(1) We propose integrating features of multiple sensors and their sequence tags as an overall fingerprint for data analyzing in crowdsensing.

(2) We consider the factor of time information to improve the efficiency of mobile activities recognizing.

(3) With deep learning model, we make the smartphone data analysis independent of smartphone orientation.

(4) We evaluate our proposed scheme with real data collected from multiple sensors of smartphone.

The rest of the paper is structured as follows. Section 2 presents a brief overview of related works. In Section 3 we introduce the basic architecture of the time-delay multilayer perception model. We explain network training in Section 4. Section 5 evaluates our schemes in human mobile activity inferring by the data we collected in realistic scenarios, and Section 6 summarizes this paper.

2. Related Work

Due to the popularity of the smartphone and multitypes sensors with which it is equipped, there is a growing interest in mobile application researches [10–13]. They leverage the sensors of smartphone to sense our physical environment or individuals' physiological parameters and so forth. The sensory data are always multimodal, representing different granularity and diverse quality. In order to well understand the potential meanings of the collected data, many researchers devote themselves to learning representation of the data that make it easier to extract useful information. In [14], authors proposed calculating resultant vectors of accelerometer, gyroscope, and magnetometer of smartphones, respectively. Then, individual defined thresholds of the three sensors are used for fall detection. The independent representation mechanism of the sensory data of multitypes sensors is used in [15, 16] as well. Although these methods are lightweight, they can only obtain a unidimensional characteristic of sensory data and cannot form a discriminant feature. For example, an accelerometer for downstairs and upstairs has similar change characteristics.

In view of the insufficiency of the independent representation mechanism of the sensory data, some researches put forward sensor fusion based schemes to learn representation of sensory data. Sensor fusion is combining of sensory data or data derived from disparate source such that the resulting information has less uncertainty than what would be possible when these sources were used individually [17]. With a fusion process, we can get more accurate and more dependable result from the disparate raw data source. For example, in [18], in order to improve localization service, the author manipulates at least four sensors including microphone, camera, WiFi radio, and accelerometer. The aim is to combine multiple features for reliable localization service. In [19], authors presented a hierarchical algorithm for the heterogeneous data representation. In the lower level, it extracts feature vectors of accelerometer and microphone for the motion and environment. In the higher level, it combines the extracted two features to get an integration feature for human activity recognition. Similarly, in [20], Zeng et al. proposed a dynamic heterogeneous sensor fusion framework to incorporate various sensory data. It learned the weights of sensors to form an integrated feature for activity recognition. The drawback of these schemes is that they only simply integrate heterogeneous data, which do not consider the influence of different sensors. In addition, some of the works have to implement coordinate reorientation of sensors to obtain the meaningful sensory data that indicate physical activities of objects [3, 6, 21], which increases the complexity of system implementations.

Since the sensory data may present different temporal characteristics for various sensing events, some studies try to explore the time characteristic in learning sensory data. To the best of our knowledge, the method used to analyze the temporal characteristics of sensory data is Hidden Markov Model-based algorithm [22, 23]. However, HMM-based algorithm requires prior knowledge to define its structure, which limits its feasibility. In addition, it analyzes transfer features of data of neighboring time points such that it cannot extract an integrated time feature of sensory data.

3. Model Review

3.1. Why to Choose Deep Learning. Theoretical results suggest that, for a complicated extraction process, the results can be further improved by applying a "deeper" structure [24, 25]. In this paper, we propose learning representation of the sensory data tagged by time labels to extract typical features using deep learning, which is a generative model that consists of multiple layers of hidden stochastic latent variables of feature. There are two advantages of our method. First of all, we consider temporal information in our algorithm for analyzing the time labelled sensory data. Secondly, different from traditional ways that think each sensor presents one feature (subfeature) separately, we believe that all sensors with which smartphone is equipped will represent a unique feature together corresponding to a context. Namely, we integrate all of the subfeatures as an overall feature, which is the *context fingerprint* we named before. We plan to explain more details of these two considerations as follows.

3.1.1. Time Information of Sensory Data. Usually, the sensory data generated by smartphone is time labelled. For example, if we sample sensory data at the time t_0 with sampling window length of τ, then we can collect data sequences like this: $\{\mathbf{x}_i', y^{(i)},\ i = 1, 2, \ldots, N\}_{[t_0+(i-1)\tau]}$, where $\mathbf{x}_i'$ denotes sample data that is sequenced according to the collecting order. Thus, $\mathbf{x}_i'$ can be time labelled by "sequence tag" T_i. $y^{(i)}$ is the class label that $\mathbf{x}_i'$ belongs to. For mobile crowdsensing, time labels are valuable information that can be used to extract changing characteristics of the sensory data with time. The time labels should be considered in the algorithm design as mucg possible as we can. However, existing methods usually cannot deal with the temporal information effectively. In this paper, we introduce a deep learning model to extract typical features from time labelled sensory data.

3.1.2. Data Integration. In order to achieve typical features extraction using deep learning, it is necessary to determine the data integration which is as input data of our deep learning. Data integration is to combine data residing at different sources and to provide users with a unified view of these data [26]. As discussed before, we combine all kinds of sensory data and sequence tags together to obtain a *context fingerprint*. In our model, rather than considering different sensors as different subfeatures separately, the data integration representation is an invariant feature, namely, the *context fingerprint*. For example, if there are four kinds of sensors, we can manipulate accelerometer, gyroscope, magnetometer, and compass. There must be a special fingerprint vector generated from a special context $\mathbf{c}$ and a time point it corresponds to. For each context $\mathbf{c}$, there must be one and only one $\mathbf{f}$ corresponding to it. And the fingerprint vector $\mathbf{c}$ is orientation invariant.

$$\mathbf{f} = \{Accx, Accy, Accz, Gyrx, Gyry, Gyrz, Magx, Magy, Magz, Com, T\}. \tag{1}$$

3.2. The Deep Learning Model. In order to extract typical features from sensory data with time labels, we use the deep learning model which consists of many layers. Up to now, there are various deep learning architectures, such as convolutional neural networks, recursive neural networks, and deep belief networks. The convolutional neural network (CNN) is suitable for processing visual and other two-dimensional data [27]. The recursive neural network (RNN) uses a tensor-based composition function and its structure is very complex [28]. RNN is suitable for natural language processing [29]. The deep belief networks can be efficiently trained in an unsupervised, layer-by-layer manner, where the layers are made of Restricted Boltzmann Machine (RBM) [30]. Thus, DBN can greatly reduce the training samples. Through the comparative analysis, we select the deep belief network (DBN) as our deep learning model. In this paper, we use four-layer DBN structure which contains a visible layer and three hidden layers. The four layers form three RBM groups as shown in Figure 1. Suppose the input data vector for our network is m-dimensional, which is collected and integrated from accelerometer, gyroscope, magnetometer, compass, and sequence tags (in this paper we only consider 4 sensors at all; for more sensors the network could be enlarged in the same way).

There are l_1 units in the visible layer of our DBN, which is responsible for accepting input samples. The samples data are time labelled. Suppose each sample contains n sampling time points; then it is easy to know that each input sample $\mathbf{X}$ is a matrix with $m \times n$, $\mathbf{X} = [x_{ij}]_{m\times n}$. The visible layer should accept every element of one sample as shown in Figure 1. Until now, the number l_1, which is linearly correlated with both m and n, can be calculated as $l_1 = mn$. As we mentioned before, the sensory data are from different granularity data sources. Thus, the samples we used here are not the raw data collected by smartphone but have been preprocessed. The data preprocessing is further discussed in this paper. The following three layers are hidden layers. The lowest hidden layer has h_1 hidden units, the next one has h_2, and the top layer has h_3 hidden units. The hidden units of the first RBM get inputs from the visible layer and then forward their well-trained outputs to the second RBM. At this time, the units of the first hidden layer become visible units in the second RBM. This process will be repeated until the top layer hidden

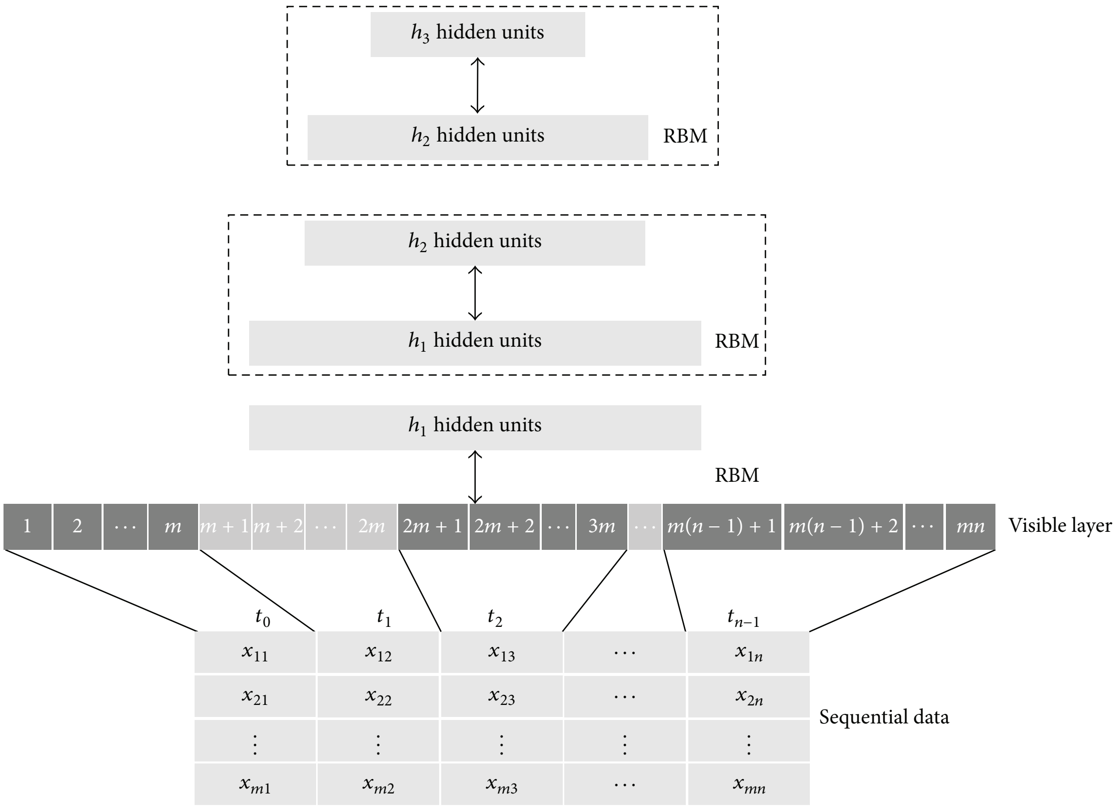

h_3 hidden units
h_2 hidden units
RBM
h_2 hidden units
h_1 hidden units
RBM
h_1 hidden units
RBM
1 2 ⋯ m $m+1$ $m+2$ ⋯ $2m$ $2m+1$ $2m+2$ ⋯ $3m$ ⋯ $m(n-1)+1$ $m(n-1)+2$ ⋯ mn
Visible layer
t_0 t_1 t_2 t_{n-1}
x_{11} x_{12} x_{13} ⋯ x_{1n}
x_{21} x_{22} x_{23} ⋯ x_{2n}
x_{m1} x_{m2} x_{m3} ⋯ x_{mn}
Sequential data

FIGURE 1: The architecture of the deep belief network for mobile crowdsensing. It contains four layers. The input data is a vector that is collected and integrated from accelerometer, gyroscope, magnetometer, compass, and time sequence.

units are determined. The number of each hidden layer unit should be carefully chosen, and we can tune and search an appropriate one by experiments.

3.3. Data Sampling and Preprocessing. In this subsection, we explain how to define and get samples from the raw data we collected from smartphones. As we discussed before, the raw data is m-dimensional that contains sensory data and sequence tags. For every sampling time point, one kind of an m-dimensional vector would be generated. In our model, we select successive sequences data as our training or test sample rather than only one sampling point, because only long enough sequential data can capture a pattern; in other words, only a successive sampling sequence can represent a special context correctly. Now, the problem is how to explore an appropriate length of time frame of sampling, n, as we discussed before.

In our model, we make the length of sample n related to a specific situation, such as human daily activity recognition [2, 14, 31] or transportation mode recognition [32–34]. For different application purpose, we choose different length of time frame n. For example, in [33], it is necessary to determine whether the people are on the bus. So, we should use longer sensory data to achieve this objective; 20~120 seconds may be an appropriate length for sampling time frame according to our experiments. However, for recognizing human daily activity, such as cycling, 5~8 seconds is enough. A reasonable value of n of different scenarios will be selected from experiments. As we discussed before, there are n times samplings for each raw sample $\mathbf{X}'$; $\mathbf{X}' = [x'_{ij}]_{m\times n}$. Since the granularity of the raw sample is different, we do not input the raw sample $\mathbf{X}'$ into our model directly. Actually, we propose doing preprocessing for $\mathbf{X}'$ and getting the refined sample $\mathbf{X} = [x_{ij}]_{m\times n}$ for training and testing as follows:

$$x_{ij} = \frac{x'_{ij} - \overline{x'_i}}{\sigma_i}, \quad i = 1, 2, \ldots, m, \ j = 1, 2, \ldots, n, \qquad (2)$$

where $\overline{x'_i} = (1/n)\sum_{j=1}^{n} x'_{ij}$ and σ_i is the variance of the ith row of the raw sample $\mathbf{X}'$. The refined sample $\mathbf{X} = [x_{ij}]_{m\times n}$ is smooth and it can also represent a *context fingerprint*.

4. The Deep Belief Network Training

After preprocessing, the samples with size of $m \times n$ could be accepted by the visible layer of DBN. However, different from image data, which is pixel matrix, our samples are time-delay data sequences. In order to integrate the sensory data, forming a typical feature, the deep belief network should be well trained. Therefore, our purpose is to find the parameters of DBN to minimize the network errors. This procedure is divided into two phases: (1) pretraining phase and (2) fine-tuning phase. In the following sections, we will describe the two phases in detail.

4.1. Pretraining Phase. As shown in Figure 1, our DBN consists of three RBM groups, which are separated from

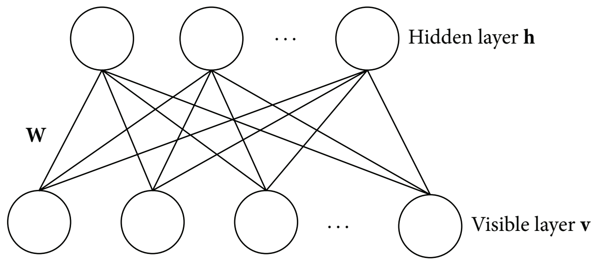


Figure 2: The model of Restricted Boltzmann Machine (RBM).

each other. Therefore, we train each RBM group individually. For each RBM, it is an undirected graph that consists of two layers: visible layer used to denote the observations and hidden layer used to denote the feature detectors. **W** is the weight of the connection between the visible layer and hidden layer. The structure of RBM is shown in Figure 2.

Let vectors **v** and **h** denote the state of visible unit and hidden unit, in which v_i denotes the state of the ith visible unit and h_j denotes the state of the jth hidden unit. For a given state of $(\mathbf{v}, \mathbf{h})$, the energy of the joint configuration in RBM is

$$E(\mathbf{v}, \mathbf{h} \mid \theta) = -\sum_{i\in\mathbf{v}} a_i v_i - \sum_{j\in\mathbf{h}} b_j h_j - \sum_{i\in\mathbf{v}}\sum_{j\in\mathbf{h}} v_i w_{ij} h_j, \tag{3}$$

where $\theta = \{w_{ij}, a_i, b_j\}$ is the parameter that needs to be trained in RBM. w_{ij} is the weight of the connection between the ith visible unit and jth hidden unit and a_i and b_j are their bias. Based on the energy function, the joint probability distribution of $(\mathbf{v}, \mathbf{h})$ is given as

$$P(\mathbf{v}, \mathbf{h} \mid \theta) = \frac{e^{-E(\mathbf{v},\mathbf{h}|\theta)}}{Z(\theta)}, \tag{4}$$

where $Z(\theta) = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h}|\theta)}$ is the partition function. For a practical problem, the aim of the pretraining algorithm is to determine the distribution of the observation data $P(\mathbf{v} \mid \theta)$, that is, the marginal probability of $P(\mathbf{v}, \mathbf{h} \mid \theta)$. It can be given as

$$P(\mathbf{v} \mid \theta) = \frac{1}{Z(\theta)} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h}|\theta)}. \tag{5}$$

Since the energy of a training sample could be lowered by raising the probability of the sample, the optimal parameter θ can be computed by maximizing the likelihood function of $P(\mathbf{v} \mid \theta)$. It can be computed by taking the derivation of the likelihood function of $P(\mathbf{v} \mid \theta)$ with respect to the parameters:

$$\begin{aligned} \frac{\partial \log P(\mathbf{v} \mid \theta)}{\partial w_{ij}} &= \left\langle v_i h_j \right\rangle_{\text{data}} - \left\langle v_i h_j \right\rangle_{\text{model}} \\ \frac{\partial \log P(\mathbf{v} \mid \theta)}{\partial a_i} &= \left\langle v_i \right\rangle_{\text{data}} - \left\langle v_i \right\rangle_{\text{model}} \\ \frac{\partial \log P(\mathbf{v} \mid \theta)}{\partial b_j} &= \left\langle h_j \right\rangle_{\text{data}} - \left\langle h_j \right\rangle_{\text{model}}, \end{aligned} \tag{6}$$

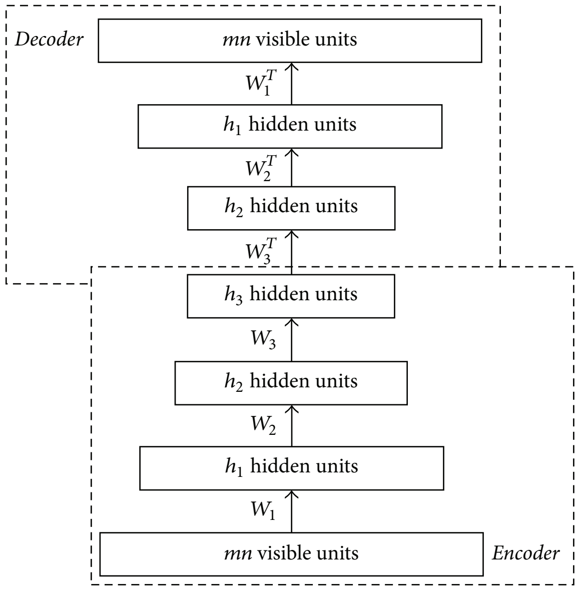


Figure 3: The model of unrolling deep belief network (DBN).

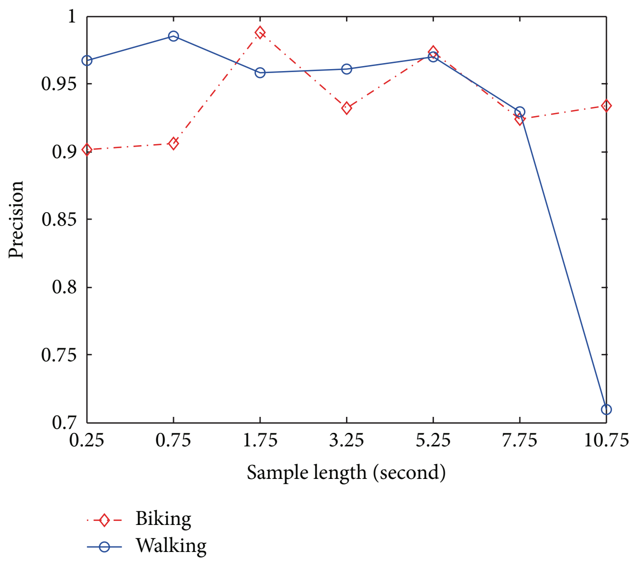
where $\langle\cdot\rangle_{\text{data}}$ is the expectation of the product of the parameters and observed data and $\langle\cdot\rangle_{\text{model}}$ is the expectation for the model observations that is generated according to the model. When the training data and generated data are similar, we obtain the optimal performance. Thus, the parameters can be updated by

$$\begin{aligned} \Delta w_{ij} &= \epsilon\left(\left\langle v_i h_j \right\rangle_{\text{data}} - \left\langle v_i h_j \right\rangle_{\text{model}}\right) \\ \Delta a_i &= \epsilon\left(\left\langle v_i \right\rangle_{\text{data}} - \left\langle v_i \right\rangle_{\text{model}}\right) \\ \Delta b_j &= \epsilon\left(\left\langle h_j \right\rangle_{\text{data}} - \left\langle h_j \right\rangle_{\text{model}}\right), \end{aligned} \tag{7}$$

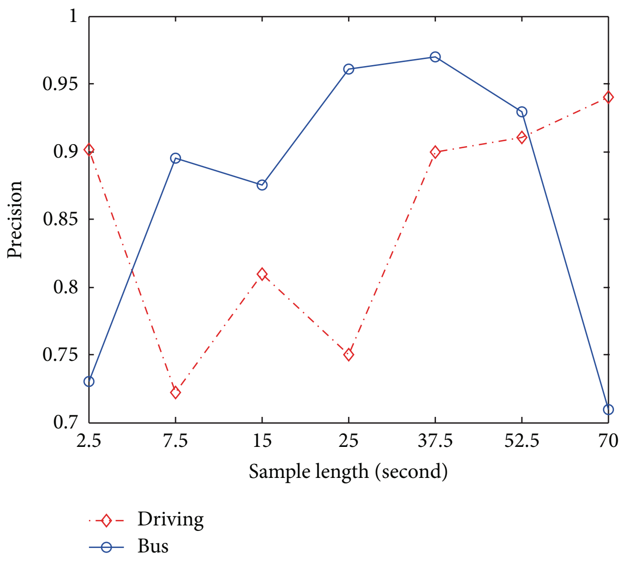
where ϵ is a learning rate. Through experimental test, ϵ is set at 0.01. After the first RBM is well trained, the hidden units in this RBM become visible unit for learning the second RBM. The layer-to-layer learning will repeat until the last RBM is well trained. At this time, we obtain coarse grain optimal values of the parameters. To further improve the result, a fine-turning process is implemented in the next phase.

4.2. Fine-Tuning Phase. The phase described above is a bottom to top unsupervised learning process to achieve the network pretraining. After that, the models unfold (as shown in Figure 3) to produce encoder and decoder network. Then, we implement a fine-turning process to optimize the parameters of the deep belief network. In this step, the process is a top to bottom supervised learning.

In order to achieve fine-tuning of the network, in this paper, we use backpropagation (BP) method, which calculates gradient descent of the mean-squared error as a function of the weights [35]. Specifically, backpropagation procedure performs two stages through the unrolling network, forward and backward. During the forward stage, we forward the training data to the input of the network and calculate the difference between the inferred hidden unit and the learned



(a) Cycling versus walking



(b) Driving versus bus

Figure 4: Precision statistics of seven different sample lengths n. (a) The precision statistics of walking/cycling with sample length as 0.75 s, 1.75 s, 3.25 s, 5.25 s, 7.75 s, and 10.75 s. (b) The precision statistics of driving/bus with sample lengths 2.5 s, 7.5 s, 15 s, 25 s, 37.5 s, 52.5 s, and 70 s.

hidden unit. In this way, an error can be computed by comparing output with desired output. For the backward stage, we can evaluate the derivatives of the error function with respect to the weights and then use them to adjust weights among all the connections. This process will repeat many times for each of the training data until the network converges. During the whole process, the initial weights are the same weights that are well trained in pretraining phase.

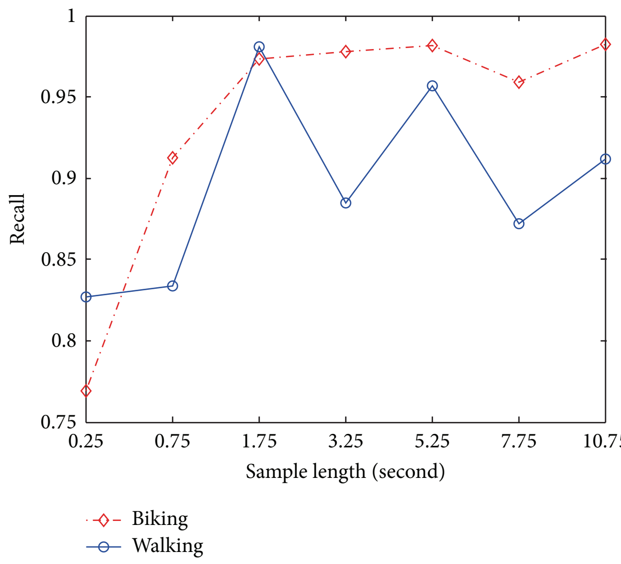
5. Evaluation

5.1. Sample Sets. Analyzing data generated by multiple sensors of smartphone to design and develop a mobile application is not the goal of this paper. The most important thing in this paper is proposing and demonstrating a novel solution for integrating and analyzing multiple time labelled sensory data effectively. As mentioned in Section 3.1, we plan to integrate four kinds of sensors, accelerometer, gyroscope, magnetometer, and compass. We can start our evaluation by explaining how to do preprocessing firstly. The testing sample sets we gathered are corresponding to two groups of mobile activities, *walking/cycling* and *driving private car/taking bus*. We have six volunteers to collect data, including 4 males and 2 females. In two weeks, the volume of sensory data they collected is over 180 hours. They carry six different Android smartphones (different handset makers) equipped with the four sensors we mentioned before. The sampling frequency is 4 Hz. And we do never restrict smartphone orientation during data collecting. It means all volunteers could do sample with the most comfortable gestures as they prefer. Usually, the orientation for woman carrying phone is different from man. But the only thing is that they have to keep one gesture constantly during one sampling period. It means that the gesture with smartphones cannot change during the sampling period. And we do the training and testing with the cross validation method. We grouped the samples into four parts and randomly choose three of them as the training set. The left part is testing set.

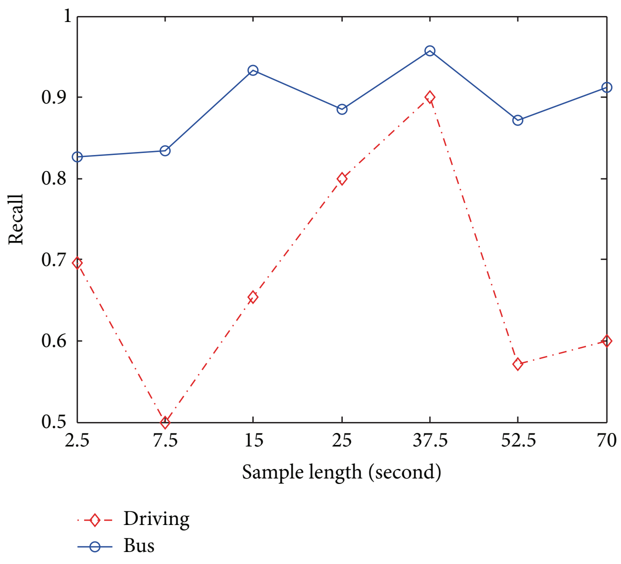
5.2. Experimental Results

5.2.1. Running DBN with Different Sample Length n. We do all the tests with two groups of human mobile activities, *walking/cycling* and *driving/bus*. Based on the integrated features, then we use an SVM classifier to distinguish the activities. Precision and recall are the most widely used quality measurements. We observe and compare the precision and recall when tuning sample length n. Figures 4 and 5 compare the precision and recall with different value of n. We firstly test our model on *walking/cycling* testing set with sample lengths 0.25 s (only one sampling point), 0.75 s, 1.75 s, 3.25 s, 5.25 s, 7.75 s, and 10.75 s, respectively. In classifying these two kinds of mobile activities from each other, we firstly define *cycling* as positive class (1) and *walking* as negative class (0). So, the precision of recognizing *cycling* can be calculated. Then, we change *walking* as positive class and get the precision for classifying *walking* as shown in Figure 4(a). From Figure 4(a), we can find that the integrated features achieve an excellent performance with different sample lengths. And a sample length around 1.75 s~5.25 s achieves a stable precision above 92% and with a peak value 98%.

The testing for *driving/bus* works acts in the same way. However, for recognizing *driving* from *bus*, we need to enlarge the sample length because only long enough sample



(a) Cycling versus walking



(b) Driving versus bus

FIGURE 5: Recall statistics of seven different sample lengths n. (a) The recall statistics of walking/cycling with sample length as 0.75 s, 1.75 s, 3.25 s, 5.25 s, 7.75 s, 10.75 s. (b) The recall statistics of driving/bus with sample lengths 2.5 s, 7.5 s, 15 s, 25 s, 37.5 s, 52.5 s, 70 s.

can capture the characteristic of driving or taking bus. Therefore, in searching an appropriate sample length, we choose the candidate sample length as 2.5 s, 7.5 s, 15 s, 25 s, 37.5 s, 52.5 s, and 70 s, respectively. As shown in Figure 4(b), the sample length n plays a more important role in distinguishing *driving/bus* than *cycling/walking*. An effective range of sample length is 37.5 s~52.5 s. Too small or too large sample length would never achieve a satisfying precision. Actually, not only n but also the number of the hidden layer unit for the two tests is also different. We choose $h_1 = 100$, $h_2 = 60$, and $h_3 = 3$ for the experiment of recognizing *walking/cycling*. For *driving/bus*, we choose $h_1 = 900$, $h_2 = 300$, and $h_3 = 4$. For the visible unit, because the sampling frequency is 4 Hz, $l_1 = 1650(11 * 37.5 * 4)$ if we select 37.5 s as the sample length.

The recall of statistics is shown in Figure 5, which reveals almost the same phenomenon with different sample lengths. However, in the test of classifying driving and bus, it is much easier to distinguish bus than to distinguish driving from testing sample set. As shown in Figures 4(b) and 5(b), both of the two quality measurements, precision and recall, achieve a higher result in distinguishing *bus* from our testing samples. One possible explanation is that bus usually runs more regularly than *driving* a private car or taxi.

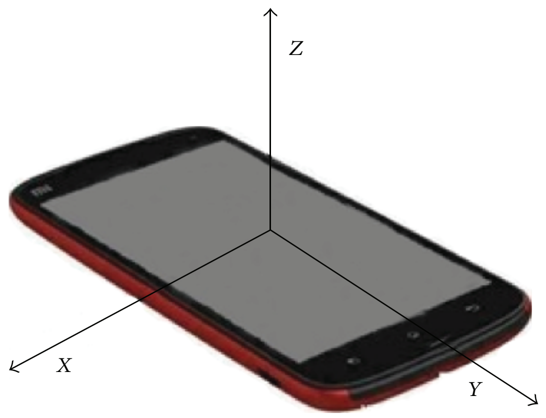
5.2.2. The Overall Performance of DBN. To evaluate the overall performance of our DBN model, we compare DBN with an Activity Recognition System (ARS) with mobile phones [36]. ARS manipulated three kinds of sensors, accelerometer, magnetometer, and gyroscope. There are some differences between DBN and the ARS. Firstly, ARS gets a constant sample length, which is 2 seconds but with a sampling frequency of 50 Hz. Therefore, there are 100 sampling points. Then, ARS calculates mean of temporal differences for the 100 sampling data sets named *Intensity*, where $Intensity := (1/100)\sum_{t=1}^{100}((x'_t - x'_{t-1})/\tau)$. For the second, ARS's network has 9 fully connected neurons in the input layer, which is less than our DBN model.

TABLE 1: Experiment performance compare ($F1$-measurement).

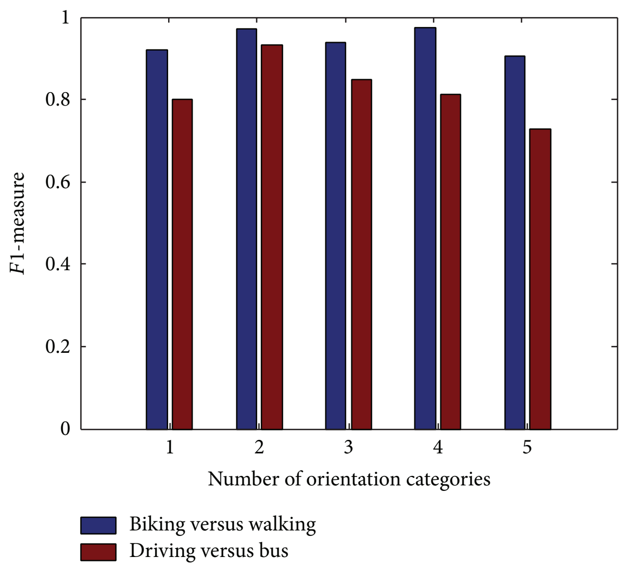
Activity	ARS	DBN
Walking	92.35%	96.36%
Cycling	75.96%	97.77%
Bus	70.10%	93.75%
Driving	58.33%	90.10%
Average	74.19%	94.50%

We conduct experiments on classifying the four mobile activities of walking, cycling, driving private car, and taking bus using DBN and ARS. For each activity, ARS uses the same sampling length, whereas DBN selects different sampling lengths for different activities. As tested above, DBN uses 2 s, 5 s, 40 s, and 35 s sampling length with 4 Hz sampling frequency, respectively, for classifying walking, cycling, driving private car, and taking bus. In order to evaluate performance overall, we introduce $F1$-score as a new quality measurement [37]. It is a measure of a test's accuracy. $F1$-score is defined as $2 * precision * recall/(precision + recall)$. The $F1$-score results of ARS and DBN are shown in Table 1.

As shown in Table 1, we can see that the integrated features extracted from our DBN perform better than ARS in recognizing all of the four mobile activities, especially for the last three activities. The reason is that the long enough temporal sequential data can capture more features of some mobile activities. Besides, DBN also considers time labels of the sensory data. Although ARS has higher sampling frequency, in a short time, there will be no significant change of the characteristic of sensory data. And it will increase the computational load.



(a) Phone orientation



(b) F1-measure of different number of orientations

Figure 6: The robustness testing for smartphone orientation invariance.

5.2.3. Evaluation of Orientation Invariance. Some of the sensors with which smartphone is equipped are orientation related as shown in Figure 6. For different orientations, data will present differently in the same context. For example, during walking, different gestures of carrying smartphone will generate different data records of accelerometer. Traditionally, in dealing with those different representations of data, we have to determine the orientation by some rules firstly [5]. However, in our model, we do not need to do this kind of job of adjusting orientations. It can learn context's different data representations for the same context effectively. In other words, it is orientation invariant.

In evaluating the orientation invariance of our method, we test it with a number of orientations and observe the corresponding performances. As discussed before, we never restrict smartphone orientation during sampling. And all volunteers could do sample with the most comfortable gestures as they prefer. Actually, there are totally five gestures volunteers used, putting their phones in coat pockets, trouser pockets, backpacks, lady's handbags, and their hands. We firstly do test on the testing data collected from only one gesture of phone carrying. Then, we increase the category of gestures to renew our testing. After that we use the extracted features to observe the varieties of performance. We also do the same experiments on cycling/walking and driving/bus. As shown in Figure 6(b), our method achieves stable results on both of the two classifying experiments. Despite data with multiple orientations, it could also be recognized with good performance.

6. Conclusion

In this paper, we propose and demonstrate a novel model to analyze multiple time labelled sensory data using deep learning in an overall view. Our method tries to integrate feature of each sensor into a combined feature (context fingerprint) and then set it as input for the DBN model. Besides, it captures and learns not only the sensory data itself but also tagged time information of data and utilizes both of them to do context inferring. When analyzing data extracted using our method, we even need not care about the orientation of smartphone during sampling. We demonstrate our model with capturing a reliable fingerprint of four sensory data sets in inferring two categories of mobile activities, *walking/biking* and *driving/bus*.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported in part by the Application Foundation and Advanced Technology Research Project of Tianjin (no. 15JCQNJC01400) and the Scientific Special Commissioner Project of Tianjin (no. 15JCTPJC58300).

References

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[2] A. M. Khan, Y.-K. Lee, S. Y. Lee, and T.-S. Kim, "Human activity recognition via an accelerometer-enabled-smartphone using Kernel discriminant analysis," in *Proceedings of the 5th International Conference on Future Information Technology (FutureTech '10)*, pp. 1–6, Busan, South Korea, May 2010.

[3] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *Proceedings of the 14th IEEE International Intelligent Transportation Systems*

Conference (ITSC '11), pp. 1609–1615, Washington, DC, USA, October 2011.

[4] C.-W. You, N. D. Lane, F. Chen et al., "CarSafe app: alerting drowsy and distracted drivers using dual cameras on smartphones," in *Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*, pp. 13–26, ACM, Taipei, Taiwan, June 2013.

[5] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys '08)*, pp. 323–336, November 2008.

[6] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, "Sensing vehicle dynamics for determining driver phone use," in *Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*, pp. 41–54, Taipei, Taiwan, June 2013.

[7] R. K. Ganti, S. Srinivasan, and A. Gacic, "Multisensor fusion in smartphones for lifestyle monitoring," in *Proceedings of the International Conference on Body Sensor Networks (BSN '10)*, pp. 36–43, Singapore, June 2010.

[8] H. Kuehne, J. Gall, and T. Serre, "An end-to-end generative framework for video segmentation and recognition," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV '16)*, pp. 1–8, Lake Placid, NY, USA, March 2016.

[9] H. Xu, Y. Lee, and C. Lee, "Activity recognition using Eigenjoints based on HMM," in *Proceedings of the 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI '15)*, pp. 300–305, IEEE, Goyang, Reoublic of Korea, October 2015.

[10] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The Pothole Patrol: Using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pp. 29–39, Breckenridge, Colo, USA, June 2008.

[11] S. K. Vashist, E. M. Schneider, and J. H. Luong, "Commercial smartphone-based devices and smart applications for personalized healthcare monitoring and management," *Diagnostics*, vol. 4, no. 3, pp. 104–128, 2014.

[12] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pp. 29–39, ACM, Breckenridge, Colo, USA, June 2008.

[13] J. Cherian, J. Luo, H. Guo, S.-S. Ho, and R. Wisbrun, "Poster: Parkgauge: gauging the congestion level of parking garages with crowdsensed parking characteristics," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pp. 395–396, ACM, Seoul, Republic of Korea, November 2015.

[14] S. Madansingh, T. A. Thrasher, C. S. Layne, and B. Lee, "Smartphone based fall detection system," in *Proceedings of the 15th International Conference on Control, Automation and Systems (ICCAS '15)*, pp. 370–374, Busan, South Korea, October 2015.

[15] J.-H. Hong, B. Margines, and A. K. Dey, "A smartphone-based sensing platform to model aggressive driving behaviors," in *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*, pp. 4047–4056, Toronto, Canada, April 2014.

[16] Q. T. Huynh, U. D. Nguyen, L. B. Irazabal, N. Ghassemian, and B. Q. Tran, "Optimization of an accelerometer and gyroscope-based fall detection algorithm," *Journal of Sensors*, vol. 2015, Article ID 452078, 8 pages, 2015.

[17] Sensorfusion, http://en.wikipedia.org/wiki/Principal_component_analysis.

[18] M. Azizyan, I. Constandache, and R. Roy Choudhury, "SurroundSense: mobile phone localization via ambience fingerprinting," in *Proceedings of the 15th Annual ACM International Conference on Mobile Computing and Networking (MobiCom '09)*, pp. 261–272, ACM, Beijing, China, September 2009.

[19] G. Filios, S. Nikoletseas, C. Pavlopoulou, M. Rapti, and S. Ziegler, "Hierarchical algorithm for daily activity recognition via smartphone sensors," in *Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT '15)*, pp. 381–386, IEEE, Milan, Italy, December 2015.

[20] M. Zeng, X. Wang, L. T. Nguyen, P. Wu, O. J. Mengshoel, and J. Zhang, "Adaptive activity recognition with dynamic heterogeneous sensor fusion," in *Proceedings of the 2014 6th International Conference on Mobile Computing, Applications and Services (MobiCASE '14)*, pp. 189–196, IEEE, November 2014.

[21] C. Song, J. Wu, M. Liu, H. Gong, and B. Gou, "RESen: sensing and evaluating the riding experience based on crowdsourcing by smart phones," in *Proceedings of the 8th International Conference on Mobile Ad Hoc and Sensor Networks (MSN '12)*, pp. 147–152, Chengdu, China, December 2012.

[22] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh, "Activity recognition and abnormality detection with the switching hidden semi-Markov model," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, pp. 838–845, IEEE, San Diego, Calif, USA, June 2005.

[23] L. Xie, S.-F. Chang, A. Divakaran, and H. Sun, "Unsupervised discovery of multilevel statistical video structures using hierarchical hidden Markov models," in *Proceedings of the International Conference on Multimedia and Expo (ICME '03)*, vol. 3, pp. III-29–III-32, Baltimore, Md, USA, July 2003.

[24] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[25] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning-a new frontier in artificial intelligence research," *IEEE Computational Intelligence Magazine*, vol. 5, no. 4, pp. 13–18, 2010.

[26] M. Lenzerini, "Data integration: a theoretical perspective," in *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '02)*, pp. 233–246, ACM, Madison, Wis, USA, June 2002.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[28] R. Socher, A. Perelygin, J. Y. Wu et al., "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, pp. 1631–1642, Citeseer, October 2013.

[29] R. Socher, C. C.-Y. Lin, C. D. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML '11)*, pp. 129–136, Bellevue, Wash, USA, June 2011.

[30] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[31] Q. Zhu, Z. Chen, and Y. C. Soh, "Smartphone-based human activity recognition in buildings using locality-constrained linear coding," in *Proceedings of the IEEE 10th Conference on Industrial Electronics and Applications (ICIEA '15)*, pp. 214–219, IEEE, Auckland, New Zealand, June 2015.

[32] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, p. 13, ACM, 2013.

[33] P. Zhou, Y. Zheng, and M. Li, "How long to wait? Predicting bus arrival time with mobile phone based participatory sensing," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, pp. 379–392, ACM, Lake District, UK, June 2012.

[34] Z. Zhang and S. Poslad, "A new post correction algorithm (PoCoA) for improved transportation mode recognition," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '13)*, pp. 1512–1518, IEEE, Manchester, UK, October 2013.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," DTIC Document, 1985.

[36] N. Győrbíró, Á. Fábián, and G. Hományi, "An activity recognition system for mobile phones," *Mobile Networks and Applications*, vol. 14, no. 1, pp. 82–91, 2009.

[37] "F1 score," https://en.wikipedia.org/wiki/F1_score.

Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 3247903, 12 pages
http://dx.doi.org/10.1155/2016/3247903

Research Article

AirCache: A Crowd-Based Solution for Geoanchored Floating Data

Armir Bujari and Claudio Enrico Palazzi

Department of Mathematics, University of Padua, Padova, Italy

Correspondence should be addressed to Claudio Enrico Palazzi; cpalazzi@math.unipd.it

Received 10 June 2016; Accepted 1 August 2016

Academic Editor: Carlos T. Calafate

The Internet edge has evolved from a simple consumer of information and data to eager producer feeding sensed data at a societal scale. The crowdsensing paradigm is a representative example which has the potential to revolutionize the way we acquire and consume data. Indeed, especially in the era of smartphones, the geographical and temporal *scopus* of data is often local. For instance, users' queries are more and more frequently about a nearby object, event, person, location, and so forth. These queries could certainly be processed and answered locally, without the need for contacting a remote server through the Internet. In this scenario, the data is alimented (sensed) by the users and, as a consequence, data lifetime is limited by human organizational factors (e.g., mobility). From this basis, data survivability in the Area of Interest (AoI) is crucial and, if not guaranteed, could undermine system deployment. Addressing this scenario, we discuss and contribute with a novel protocol named AirCache, whose aim is to guarantee data availability in the AoI while at the same time reducing the data access costs at the network edges. We assess our proposal through a simulation analysis showing that our approach effectively fulfills its design objectives.

1. Introduction

The technological revolution is palpable now more than ever as we delve into the era of the Internet of Everything (IoE). Through the IoE paradigm, every object, social network profile, person, interest, and process will be part of an interconnected scenario so as to create new intelligence and services [1]. Our lives have already been enriched by the success of smartphones which have brought into our pockets any sort of sensors. Every user is now a producer of a great quantity of data that, when combined, generate new information that can then be redistributed to others. In essence, every user becomes both a producer and a consumer of information [2]; even more, data generation and consumption are two inextricable processes as the latter also provides information about user preferences, location, interest, actions, and so forth (i.e., data generation). This people-centric sensing paradigm leads to a scenario with a diminishing presence of centralized server-based services [3]. Rather, digital services will be more and more based on crowdsensing and the ability to share the information in a distributed, decentralized way [4].

In this context, it is interesting to notice how in recent years, despite the continuous improvements of the connectivity means, users have been more and more looking for location-based services. A person connects to the Internet to find out local events and restaurants, friends in proximity, touristic information, offers, news, and much more related to the environment she/he is immersed in. It could be hence more efficient to just let these kinds of information float in a certain Area of Interest (AoI), while users generate, consume, and maintain them in the AoI. The latter, in particular, could be achieved by having the AoI-related data replicated among some of the users moving in the AoI and stationing there and removed from storage if a user walks out of the AoI. In our case study, we assume that the data is initially generated by one of the users or brought from elsewhere. Nodes in the AoI can exchange data (in an opportunistic way, when in proximity of each other) for two main reasons: (i) in case one user is interested in the available floating data and

(ii) as nodes with the floating data in their storage may exit the AoI or be simply turned off, the data need sometimes to be replicated among the nodes to ensure the survivability in the AoI [5].

This creates a sort of a layer of data which is supported by the presence and participation of users and that is strictly connected to a specific AoI, hence the name *floating data* [6]. Clearly, the resilience of this system depends on the number of users that circulate at any time in an AoI but it could be certainly effective in crowded areas of any town. Furthermore, the data could appear and vanish considering the actual interest they could raise and their temporal and geographical relevance. This is in contrast with the current data model, being available globally and endlessly regardless of their consumption profile [7].

In such a system, the main features that need to be provided are accessibility and survivability in the AoI. One naive, simple but not efficient, way to generate floating data could be that of replicating the data on all the nodes in the network through data sharing at any node encounter in the AoI. Although this strategy could increase the chances for the survivability of some data in the AoI, it is also limiting and inefficient as it will consume more storage and energy than actually needed.

Addressing the issues, we discuss AirCache (AC), a distributed protocol aimed at guaranteeing data survivability in the AoI. AirCache carefully replicates the content into the network by taking into consideration energy and storage availability of nodes. Nodes in our system periodically and autonomously send beacon messages announcing their respective capabilities and content they possess. This information is then exploited in order to enforce the replication policy required to augment data availability in the AoI. Furthermore, AC is capable of controlling the spatial distribution of the data, easing data access at the network edges. The contribution of this paper is hence twofold: (i) presenting the concept of AC, its *modus operandi,* and applicability; (ii) proposing possible algorithms and protocols that can be used to generate and maintain an AC in certain locations.

The outline of the paper is as follows: in Section 2, we provide the necessary background information needed to better comprehend the context under scrutiny. Section 3 discusses state-of-the art proposals in the context of mobile networks addressing a problem similar to ours. Next, through Section 4 we introduce our proposal, discussing its *modus operandi* and key built-in features required to satisfy our goals. Section 5 discusses the simulation environment and the evaluation strategy adopted to assess the proposal, while Section 6 discusses the evaluation outcome. Finally, Section 7 concludes the paper.

2. Background

The advances on wireless technology have created many new possibilities for communication, giving rise to new and decentralized networking paradigms such as the Mobile Ad Hoc Networks (MANETs [8]). Nodes in this context dynamically self-organize in a wireless overlay providing networking support for the upper layers. This networking paradigm is inherently decentralized and communication in these settings follows the well-known peer-to-peer (P2P) communication model rather than a client-server one. In these settings, networking functions must be designed to contemplate for disruptions due to mobility [9, 10].

In contrast with MANETs, the OPPortunistic NETworking (OppNet [11]) paradigm in addition exploits unplanned communication opportunities to move data between endpoints. In MANETs mobility is a disruptive force hindering networking performance, while OppNets leverage on mobility in order to create additional communication opportunities. To achieve this goal, networking functions need to contemplate for a high degree of heterogeneity of the entities involved in communication in terms of capabilities and mobility dynamics [12, 13].

Nevertheless, the critical mass for opportunistic communications and the ever increasing device capabilities are two important factors in favor of this networking paradigm. A representative example embracing this opportunity is the people-centric sensing (crowdsensing) concept. The pervasive coverage of mobile sensing devices producing and collecting sensory data has provided the basis for new applications and services which can benefit our community as a whole [14]. Indeed, cooperative environmental monitoring or public sensing has created a cost-effective way of collecting a vast variety of data which could be exploited in several ways.

In this spirit, Haggle is a first project of its kind aiming to bring a content-centric solution for opportunistic networks comprised of mobile handheld devices [15]. Haggle presents a modular framework which can be extended to support a variety of applications and services embracing the ad hoc networking paradigm. At the core of the framework stands the search-based module which transparently matches data received during opportunistic communications servicing them to the application layer. Search is hence identified as a first-class operation, relying upon lower layer functionalities to achieve its application-specific goal.

ICEMAN follows in these steps even though it was designed for use in tactical scenarios [16]. It is seen as an evolution of the Haggle architecture with a rich set of capabilities aiming to provide reliable communication in sensitive and disruptive environments. This is achieved by introducing networking coding techniques and utility-based content caching at the transmission layer counteracting the disruptive nature of dynamic environments.

While the above projects are an interesting next step for AirCache, our aim is to analyze and study the underlying primitives needed to sustain the concept of floating data. Indeed, AC can be deployed to support a vast variety of crowdsensing applications, ranging from infotainment to urban security. For instance, an urban security application whereby users announce information pertaining to this context could be sustained by a nearby deployed AirCache. Information related to the context might involve a warning about a nearby hole which is announced and sustained within the AC and of interest to a blind person passing through. Free-speech manifestations might employ AirCache given its inherent benefits of being decentralized and operator-free. On the other hand, infotainment information could involve the dissemination of local events and goods which might expire later on. To this end, in the following section we review state-of-the-art work in the context of mobile networks similar in spirit to ours. Next, we delve into the technical details of our solution.

3. Related Work

We argued that data lifetime and accessibility in the AoI are ingredients of paramount importance which demand attention in order to enable our envisioned scenario. Once data is created or injected into the AC, a mechanism guaranteeing its immediate availability needs to be in place. A trivial approach whereby data is replicated upon each opportunistic encounter might seem as a solution to both requirements. However, this epidemic dissemination of data is paid in terms of device energy consumption and network storage as a whole.

To tackle the issues, we reviewed literature material in the context of content placement and replication techniques to check if prior work could serve our purposes. Such techniques have been vastly investigated in the realm of infrastructure networks where topological information is stable in time and is exploited to position data at fixed storage locations [17]. In our scenario, network topology is volatile and transient in time; hence these techniques cannot be adopted as they are.

Networking techniques relying on the assumption of a stable and global knowledge of network topology are unfeasible for generic OppNets. Tackling these issues in mobile, dynamic environments are HybridCache in [18] and Hamlet in [19] which rely on local information exchange among neighboring nodes to fulfill their respective objectives. Both approaches rely on local knowledge employing opportunistic caching techniques. Nodes in HybridCache achieve this by caching either the relayed data or the entire path toward that specific data content. The strategy employed depends on the size of the data being relayed which is a system parameter configured at system bootstrap.

Hamlet builds on this idea bringing it a step further. The caching strategy aims to create content diversity in the local vicinity and this is achieved by estimating the cached items in the neighborhood. Differently from HybridCache, where each node is an autonomous entity making decisions, in Hamlet there is a distributed, collective neighborhood effort trying to optimize resource allocation. Both these techniques rely on local information to perform their duty and, hence, lead to suboptimal, best-effort solution in achieving their goal.

Instead of binding to physical nodes, we can anchor (bind) data to geographical location. This way we are decoupling the content placement problem from the volatile network topology. In this direction, the authors of [20] propose a strategy whereby data is anchored at geographical locations and content replication degree inside an area is based on content popularity. The authors show through simulations that their proposal does outperform prior techniques in terms of data access costs. The proposals idea is to define multilevel grids covering a physical place, allocating popular contents to fine-grained grids. In this way popular content is more likely to be available nearby when compared to less popular content.

In order for the above scheme to reach its full potential, content popularity needs to be known in advance (a closed-world assumption) in order to compute the content anchoring strategy. Data in our scenario is produced and maintained by the network nodes itself. Considering this, the proposal above is reduced to simply maintaining an arbitrary number of replicas for each data in a bounded geographical area. Moreover, in our scenario data is produced locally and is volatile. If no mechanism guaranteeing data survivability (availability) is in place the data vanish.

Proposals [21, 22] address a scenario similar to ours. Through a fluid model, the authors in [21] derive a sufficient condition for the content to float with very high probability. The model takes into consideration the average number of nodes, the average node contact rate, and the average sojourn time of nodes in the AoI. The authors validate their model against simulation analysis showing that the derived condition is actually a necessary condition for the content to float with a high probability. To keep the model tractable, the authors make unrealistic assumptions regarding, for example, the dissemination strategy, assuming an epidemic approach to data dissemination. Following these steps, the authors in [23] provide evidence from a field trial employing a proof-of-concept implementation of a floating data application.

In the following, we introduce and discuss AirCache, a decentralized algorithm whereby nodes cooperate toward a common goal in order to guarantee data survivability inside an AoI. Our solution lends itself to another interesting feature of controlling the spatial distribution of data through the network, hence the possibility to control the data access costs. AirCache has a conservative approach on the data replication, taking into consideration energy and memory consumption of nodes.

4. AirCache: A Floating Data Network

We aim to devise a decentralized solution for the data anchoring problem whereby the data are binded to a geographical location of interest. The data are either produced locally or brought from elsewhere and need to be kept alive until some conditions are met. Indeed, the data might expire, for example, due to a time attribute attached to it expiring or as a consequence of node mobility and/or the absence of the necessary critical mass capable of sustaining an AC. From a practical point of view, a node possessing the data (e.g., the source node itself or a node which has taken this responsibility) performs a search for a nearby existing AC to which it can hook the data. If an AC is not present in the surroundings, the node takes the responsibility to create a new one. Independent of the starting conditions, once an AC is available, it is crucial to start replicating the data so as to have it persisting in the area regardless of node swarming.

Our solutions are designed to guarantee that a minimum number (Min) of replicas are always present in the system.

Of course, these criteria can only be enforced if the number of nodes required to fulfill is present in the anchoring zone. On the other hand, avoiding the negative effects of an epidemic dissemination, a good design choice is to also enforce an upper threshold (Max) of replicas. In specifics, the system allows a number of average (Avg) replicas in the AoI and exploits hysteresis in order to decide whether the replicas have to be augmented (Avg $<$ Min) or diminished (Avg $\geq$ Max) so as to reach a stable situation where the number of replicas equals Avg. In this way, our system can provide guarantees that a balance of replicas between an upper and a lower number is always present.

Nodes in the system periodically broadcast their own capabilities translated in terms of energy, mobility, and memory space. This information is later on exploited whenever a content replication event is triggered. In these settings, a distributed algorithm establishing the grounds for node cooperation needs to be put in place. To this end, we delved into scientific literature and found an interesting solution that could suit our purposes. The Reliable R-Aloha (RR-ALOHA [24]) protocol is a distributed collision-free protocol employing a reservation mechanism, establishing the grounds for node cooperation. Through the rest of this section we detail our proposal built on top of the RR-Aloha protocol, discussing the added features and overall system *modus operandi*.

4.1. Joining the AC and Data Replication Policy. Reliable R-Aloha is a distributed MAC layer protocol employing a reservation mechanism for assigning dedicated communication slots to nodes in a distributed fashion. The reservation mechanism employs a *Frame Information* (FI) structure comprised of a certain number of *Basic Channels* (BCH) or communication slots of a certain duration which are contended by nodes. The protocol is collision-free and has been shown to solve the *hidden terminal problem*, providing the means for a reliable single-hop broadcast channel among neighboring nodes. In essence, RR-Aloha builds a slotted shared channel guaranteed to be accessed in mutual exclusion among nodes. Though the protocol is particularly apt to the slotted physical layers it could be ported on top of any physical layer, in particular on top of the 802.11 stack.

A node upon entering an AoI, a geographical location where some data needs to be anchored, goes into listening mode. While in listening mode, the node verifies whether there is an existing local AC nearby. If no local FI announcements are heard during this period, the node can proceed with creating a fresh AC; otherwise it enters a contention period (join-mode). In both cases, a node listens to nearby transmissions for a Frame Information Time (FIT) in order to reduce the probability of nearby collisions in case an AC is present. The current implementation considers a fixed FI length with a default value set to 100 BCHs. However, this is a system configuration parameter which can be set at system bootstrap. The consequence of a fixed FI length poses an upper bound on the number of nodes that can contend the slots in a FI. Also, as in the original design we further limit the number N of contending nodes for a FI to 50, leaving the rest of the communication slots for use in cases when additional bandwidth is required (later on).

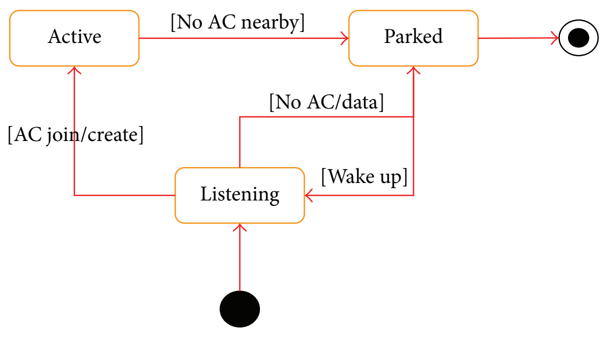


FIGURE 1: Diagram representing node state transitions.

After a FIT listening period, if the node has not sensed the presence of an AC and has data that require anchoring, it proceeds with the creation of the AC; otherwise it enters a contention period. The contention resolution period, which we discuss later in its fullness, ends with the node acquiring its own communication slot referred to as basic channel. Once the BCH is acquired, the node becomes active and participates in the system by broadcasting its view of the surroundings along with its chosen basic channel identifier appended at the *end* of the FI. Otherwise, if a node does not have any data content that needs anchoring and no local AC is sensed nearby, it is considered as parked. Parked nodes are not considered as part of the system but are eligible candidates at a later moment. Also, nodes can enter a parking mode whenever the AC has exhausted the available communication slots. Figure 1 depicts this entire process by showing the nodes state transition diagram.

Having introduced the basics of the AC mechanism, we need to discuss how the system enforces the replication policy. Each node, after Frame Information Time has gone by, has all the necessary information about the data replication level within the neighborhood or cluster (refer to Section 4.2). If the number of replicas for a particular data content is less than a threshold (Avg $\leq$ Min) and the system has the capability to further replicate it, a replication event is triggered. Contemplating for resource-consumption, the most capable nodes are involved in the replication process. The node eligible to broadcast a replica of the data is chosen based on the battery index (higher energy index) whereas the receiver(s) election is based on the buffer occupancy index (lower buffer occupancy). If the triggered event did not have any effect on the number of replicas the process repeats.

Similarly, if the maximum threshold is exceeded (Avg $\geq$ Max), the redundant replicas are discarded. The nodes eligible in this case are those with the lowest battery and buffer occupancy index and the resulting index is computed giving equal weight to both indexes.

4.2. Slot Reservation Mechanism. Before discussing the salient features of the AC protocol and delving into its algorithmic details, we start by providing a formal definition of cluster coined in the original RR-Aloha proposal.

Definition. Consider Cluster(C) = {Node(A) $\wedge$ Node(B) | Node(A) $\neq$ Node(B) $\wedge$ Node(A) $\in$ C $\wedge$ Node(B) $\in$ C $\Leftrightarrow$ Node(A) hears Node(B) $\wedge$ Node(B) hears Node(A)}.

That is to say, nodes participating within a cluster are nodes that can all hear and be heard by each other. In other words, a cluster is a strongly connected component of the temporal connection graph with vertexes where the nodes and the edges are bidirectional connections among these nodes.

A basic channel in the FI is either reserved or free. Nodes in the system periodically, after a FIT, confirm their reservation by broadcasting their perception of the surrounding neighborhood. Upon receiving this FI transmission, each node checks if its position as overheard by others is consistent with its view. If this is not the case, the node enters in contention mode for another slot. RR-Aloha adopts a sliding frame mechanism in order to compute this outcome whereby each node verifies the status of the BCH comprising the FI based on previous transmitted information from its single-hop neighboring nodes. We adopt a slightly different approach in processing the FI slot status occupancy as shown in Algorithm 1 and exemplified in Figure 2.

The original algorithm accounts only for a partial information of the overheard FIs, those parts of information falling inside the sliding frame of N slots. Differently from the original proposal, we exploit a node's local view in its fullness. The idea behind this design choice is due to the different mobility dynamics of vehicular network, context for which RR-Aloha was initially conceived, and human comprised networks. The former being more dynamic, demanding a more reactive algorithm for slot status computation. Our approach allows for a more complete picture of the slot reservation status, information which is not propagated but is instead used locally to reserve additional bandwidth whenever needed (later on). In this way, we are considering information which is at most 2 $*$ FIT old. However, taking into account the human mobility in most scenarios and the 802.11 transmission range, this amount of time is negligible. As a future work, we plan to evaluate our approach when combined with optimal forwarding algorithms proposed in the context of vehicular networks [25, 26].

To implement the discussed features, we have devised some application level data structures as evidenced in Figure 3. The Frame Information is a data structure comprised of basic channels (BCHs), denoting the slot occupancy status of the 1-hop neighborhood. The BCH on its turn is also a data structure comprised of the following fields:

(1) A node identifier represented by an integer value holding the node identifier who has reserved that particular BCH, otherwise −1.

(2) The slot occupancy status represented by a flag with a true value if the BCH is *Free* or false if it is *Reserved* by some node.

(3) Another flag variable which is used to denote if a BCH is eligible for use in case some extra bandwidth is required for a transmission.

The *ReservationBoard* used by Algorithm 1 has structure similar to the FI with the difference that its contents denote the slot status occupancy in a 2-hop coverage area. The reason for holding a 2-hop slot occupancy status in the ReservationBoard is so as to implement the extra bandwidth reservation mechanism whereby nodes reserve additional transmission slots whenever the data to be transmitted cannot not fit inside a single BCH. If conditions are met and a node requires extra transmissions slots, it unilaterally decides and reserves the additional slots after consulting its local reservation board. Its intention is then notified in the outgoing FI. The additional slot reservations expire as soon as they are not needed anymore and this control is contemplated through lines 23–26 of Algorithm 1. This information is sent in broadcast once and only by the node requiring the slots.

Referring to Figure 3, *TableInfo* is yet another data structure holding the device capabilities and a list of content identifiers cached by the node. The identifier in practice might consist of the resulting hash value of the raw data content. Nodes broadcast this information in their reserved BCH along with the FI status. The replication mechanism exploits these data to compute and enforce the replication policy.

4.3. Controlling the Spatial Distribution of the Data. Data accessibility inside the Area of Interest is another important feature we deem worth pursuing. Depending on the data entry point in the AoI, data may get confined and replicated only in limited portions of the AoI. This happens because nodes within the AC start enforcing the replication policy only after having heard about the existence of the data. While it might happen that the data gets replicated in the entire AC, for example, due to mobility dynamics, this event is accidental rather than intentional.

Hence, it is necessary to provide a mechanism which can effectively control the spatial distribution of the data. The rationale being that a homogeneous data distribution in the area reduces data access costs, easing data availability in the AoI. Cluster intersection nodes are the best candidates which can be exploited to fulfill this objective. Intuitively, they are in the position that exposes the data to new areas which once introduced can be replicated if the conditions are met. To this end, after a FIT, nodes autonomously check whether they are positioned in a cluster intersection. This step involves partitioning the set of confirmed slots in the outgoing FI into disjoint clusters as detailed by Algorithm 2.

After completing the steps involved in Algorithm 1, nodes autonomously verify their position in their neighborhood in order to find out whether they are positioned at cluster intersections. The neighborhood connectivity graph is built by exploiting the received FIs transmissions. Through Algorithm 2 each node checks the received FIs whether they have an entry with a slot assigned to him (line 1).

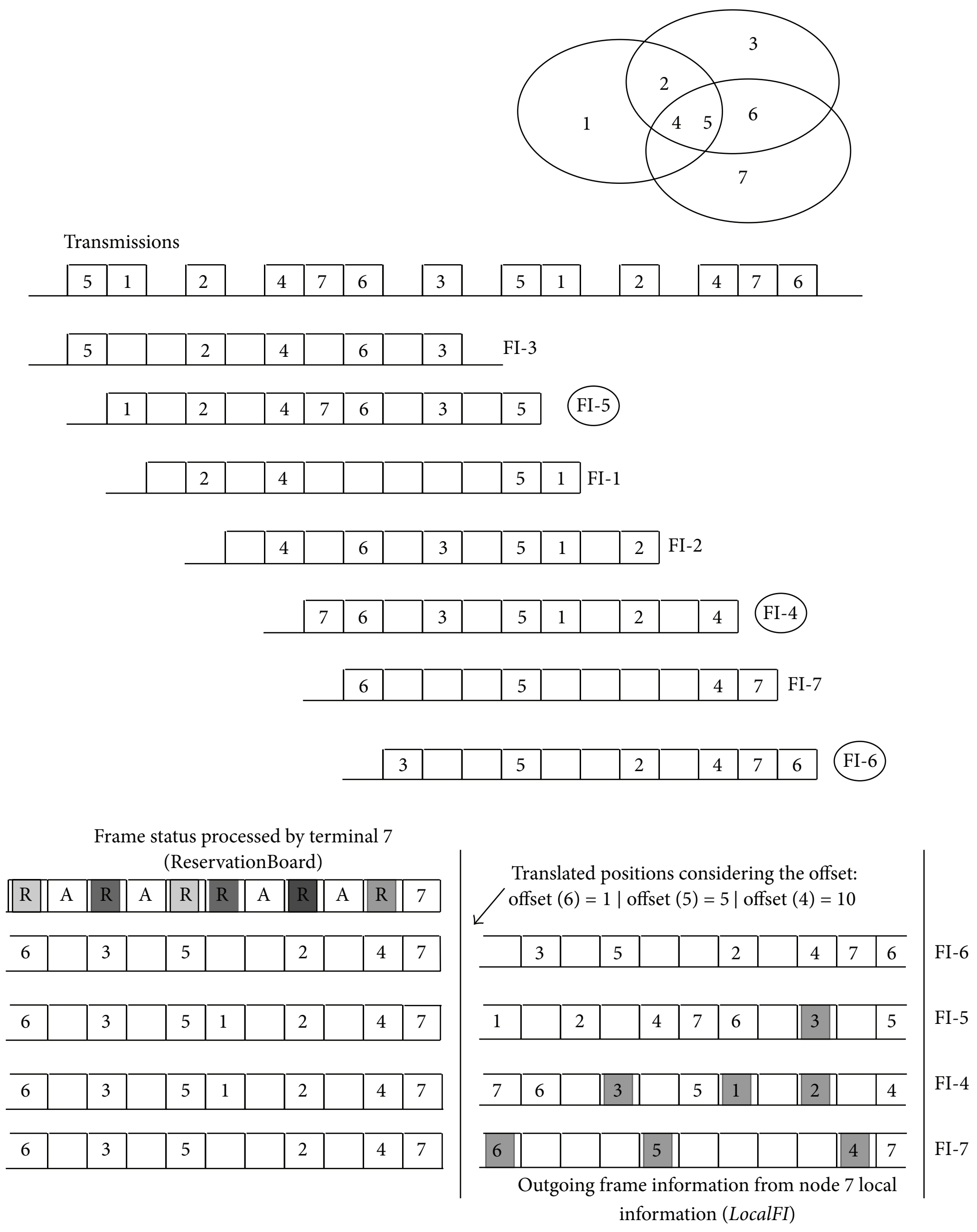

Transmissions
FI-3
FI-5
FI-1
FI-2
FI-4
FI-7
FI-6
Frame status processed by terminal 7 (ReservationBoard)
Translated positions considering the offset: offset (6) = 1 | offset (5) = 5 | offset (4) = 10
Outgoing frame information from node 7 local information (LocalFI)

Figure 2: Algorithm employed for slot status computation. In the upper part the FI composition transmitted by each node is shown while the bottom part exemplifies the steps involved in the computation of the ReservationBoard and of the outgoing FI from node 7 perspective. Circled nodes are part of node's 7 cluster.

This means that a bidirectional connection exists between the two. After this filtering process, the algorithm starts to partition the set of remaining nodes into disjoint clusters by exploiting the cluster property defined above (line 8). To avoid ambiguities, we point out that the computed clusters are not necessarily complete or entirely disjoint. The algorithm intentionally does not account for common elements (line 11) and incomplete information comes from the fact that we use only local information rather than a global view of the network. The purpose for this is related to the replication policy enforced by nodes found in cluster intersections which we detail below.

After nodes have inferred their respective positions in the neighborhood according to Algorithm 2, different behaviors are assumed depending on their role. Nodes present in at most one cluster react as explained through Section 4.1, while nodes found at cluster intersections, that is, nodes that participate in more than one connected cluster, have the responsibility to enforce the replication policy on a per cluster basis. This is achieved by making use of a special data packet which we call a *pull* data packet. This packet is sent in broadcast whenever the replica level inside a cluster is under the threshold and the cluster node has not a cached replica of the data itself. The replication criteria also take

```
input: FrameInformation (LocalFI), ReservationBoard (ReservationBoard), Received
FrameInformation(s) (ReceivedFIs), Local Node Identifier (NodeId)
output: Updated ReservationBoard and FrameInformation
(1)  bool Collision ← False, Owned ← False;
(2)  int Position ← −1; BCH Channel ← ∅;
(3)  for i ← 1 to Length(LocalFI) do
(4)      BCH Heard ← BasicChannel(LocalFI, i);
(5)      foreach fi in receivedFIs do
(6)          Position ← Shift(fi, i);
(7)          Channel ← BasicChannel(fi, position);
(8)          Collision ← False;
(9)          Owned ← Owner(Channel);
(10)         if Owned and NotSameId(Channel, Heard) then
(11)             Collision ← True;
(12)             break;
(13)         end
(14)     end
(15)     if Collision then
(16)         UpdateBoard(Reservationboard, i, Free);
(17)         UpdateFI(LocalFI, i, Free);
(18)         if NodeId = Id(Heard) then
(19)             Collision(True);
(20)             EnterContention();
(21)         end
(22)     else
(23)         if (AdditionalAndExpired(Channel)) then
(24)             UpdateBoard(board, i, Free);
(25)         else
(26)             UpdateBoard(board, i, Reserved);
(27)         end
(28)     end
(29) end
```

Algorithm 1: Slot status computation.

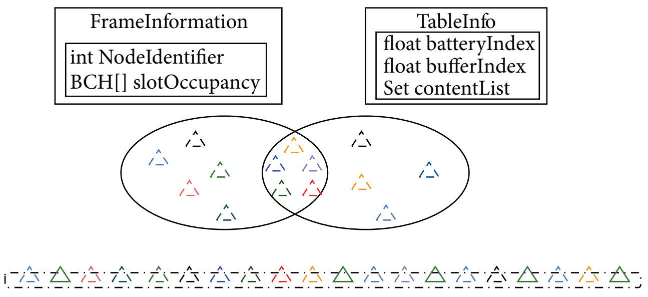


Figure 3: Devised data structures and respective data packets transmitted during the BCH. Green triangles denote free slots.

into account the number of nodes available in that particular cluster; that is, the minimum threshold is computed as the $\text{Min}(\text{Min}_{\text{thresh}}, \text{card}(C_i))$.

4.4. Discussion and Current Limitations. There are different design criteria and tradeoffs which need to be taken into careful consideration before deploying an AC. As discussed above, the FI has a fixed number of slots which needs to be decided *a priori*. Yet, a high number of slots means more nodes can participate in the system but at the same time it slows system reactivity to data replication. Symmetrically, a low number of slots could mean nodes being parked while having a reactive system which is paid in terms of battery consumption.

To tackle this issue, improvements have been proposed to the original RR-Aloha protocol. These include the possibility to dynamically adapt the Frame Information size by increasing or shortening its size depending on its level of fragmentation [27]. The fragmentation is measured by exploiting slots status information after a FIT. In our current implementation we do not contemplate this feature and leave it as a necessary future work.

A practical implication to the protocol's *modus operandi* is energy consumption due to the need to periodically reserve the communication slot. When considering an AC sustained by handheld portable devices this is indeed an issue which needs to be addressed more thoroughly. A way forward in addressing this issue might be to alternate nodes from active to parked mode and vice versa when certain criteria are met (e.g., the replica level is within the predefined thresholds).

```
     input: FrameInformation (LocalFI), Set (ReceivedFIs)
            Set (ConnectedComponents)
     output: Set of connected components ConnectedComponents
(1)  Set Connected ← Bidirectional(LocalFI, ReceivedFIs);
(2)  Cluster P;
(3)  P ← P ∪ FirstElement(Connected);
(4)  ConnectedComponents ← ConnectedComponents ∪ P;
(5)  foreach Slot in Connected do
(6)      bool InsideAnyCluster ← False;
(7)      foreach C in ConnectedComponents do
(8)          if IsMember(NodeId(Slot), C, ReceivedFIs) then
(9)              P ← P ∪ NodeId(Slot);
(10)             InsideAnyCluster ← True;
(11)             break;
(12)         end
(13)     end
(14)     if not InsideAnyCluster then
(15)         Cluster P_New;
(16)         P_New ← P_New ∪ NodeId(Slot);
(17)         ConnectedComponents ← ConnectedComponents ∪ P_New;
(18)     end
(19) end
```

Algorithm 2: Cluster set computation.

Yet, another issue is the AC interference phenomenon which occurs when different ACs come within communication range of each other. This interference can come as a result of a potentially different time synchronization of nodes participating in the different ACs. The phenomenon occurs even when a node caching data from an AC-1 moves outside the reach area of AC-1 but is still inside the AoI and has valid data that need anchoring. As per current design, this will trigger the node to create an AC-2 opened to other nodes to join in. At some point in time, the two ACs might end up interfering with each other.

The interference phenomenon undermines a factual deployment of the AC; hence we provide some insight as to how it might be addressed. The solution to this phenomenon might be to add an identifier (e.g., a timestamp of creation) to each autonomous AC which is propagated within the FI. This way nodes hearing transmissions originating from a different AC become temporarily parked. Interference between nearby communications is solved by nodes entering a recontention period for slots in their respective ACs. In this way, after less than a FIT, respective nodes can identify the presence of the other AC. Once interfering nodes are parked, an AC merger procedure could be put in place to handle the gradual merger of conflicting ACs with one another. Alternatively, nodes might stay parked until a single AC is present in their surroundings.

We deem this issue an important one, and how this interacts with a dynamic frame adaptation algorithm deserves further investigation. The experimentation part of this scenario is orchestrated in such a way so as to avoid this problem from occurring (refer to Section 5).

5. Simulation Environment and Evaluation Strategy

We chose to implement and evaluate our proposal in the Network Simulator 3 (NS3 [28]), a state-of-the-art simulation environment equipped with the 802.11 Phy/MAC stack. Before we delve into the details of the experimentation scenario, we briefly discuss some components of the simulation environment under scrutiny relevant to our work. In Figure 4 the *Node* component is shown which is an abstraction for the basic computing device. Nodes have certain capabilities which can be configured and acted upon at run-time. In our experimentation each node is equipped with a wireless 802.11 g interface and we adopt a flat addressing scheme, assigning each node a unique layer-3 address. This is of course a simplification as in a real deployment scenario this assumption is not valid anymore. However, the dynamic address assignment problem in decentralized environments is still an open research issue [29]; Level-2, MAC addresses are used in practice.

(1) *Channel and NetDevice.* They represent an abstraction for the transmission medium (either wireless or wired) and the network device respectively. In our simulation we adopt the wireless channel specification proposed by [30] which is featured in NS3.

(2) *Mobility Model.* To model the targeted scenario each node is configured with a proper mobility model. Indeed, in order to demonstrate the feasibility of our solution, we adopt the Random Way Point (RWP) mobility model which despite its simplicity provides some insights into the AC's behavior. The mobility

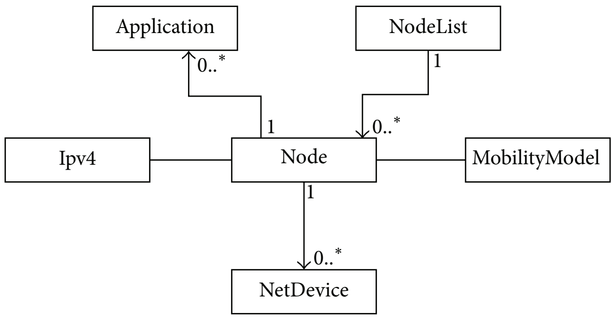


Figure 4: Node class diagram.

traces have been generated through the BonnMotion tool [31]. More details on this matter are given later on (refer to Section 6).

(3) *Application.* It represents the application layer where all the logic is implemented. The algorithms discussed throughout the paper have been implemented at this layer.

(4) *Energy Model.* It is yet another feature which can be attached to a node, accounting for the energy consumption due to communication. Our solution exploits this information when electing the sender/receiver nodes in the replication process.

The main objective of our proposal was to provide some guarantees of data survivability in the AoI. Therefore, we focus on the *TTL* metric, that is, the elapsed time interval from data being brought into the AC and the time they cease to exist. Yet another mandatory requisite was the control of spatial data distribution within the AC. Intuitively, a more homogeneous distribution of the data could ease data availability in the AoI. To this end, demonstrating the feasibility of our approach, we perform different simulations with varying node density and mobility characteristics which are discussed through Section 6.

Recalling the argument stated in Section 4.4, our protocol in its current implementation does not contemplate for the scenario of conflicting ACs. In the simulations this situation is avoided by considering a single AoI, the same for all data contents present in our simulation environment. Nodes in the simulation scenario can only join and not create the AC, while the node creating (bootstrapping) the AC is chosen randomly at the beginning of the simulation time from the entire node population.

6. Results

Through this section we discuss the simulation strategy and outcome. The evaluation metrics we employ are the time interval the data survives within the AoI and a distance metric formulated as shown below:

$$\text{Distance}_{\text{data}} = \frac{1}{N} \times \sum_{n \in \text{AoI}} \text{dist}_{\min}(n, \text{datum}). \quad (1)$$

Table 1: Parameters for the RWP mobility model used for the evaluation.

Simulation area	400×400
Mobility	$[0.5, 1.5]$ m/s
Pause time	30 s
Trace duration	60 min
Attraction point	200×200
AoI radius	150 m

The metric measures the minimum number of hops required to reach the data, averaged over the population of nodes sustaining the AC. The set $n \in$ AoI includes all nodes inside the AoI even if they have not yet joined the AC. Nodes caching a replica of the data are not considered. As anticipated, the mobility model used to evaluate our proposal is the RWP mobility model generated with the parameters shown in Table 1 and a cut-off time of 3600 s.

To activate relevant features of our protocol, each node in the population is attributed an initial battery level taken uniformly at random from the range [500, 1750] mAh. We do not account for the buffer occupancy level and the sender/receiver of data is chosen by exploiting the battery level only. Each studied scenario we perform 20 runs per configuration employing different mobility seeds. As discussed in Section 5, we avoid the AC interference phenomenon by delegating the responsibility of AC creation to a single node chosen at random node within the AoI and restrict the others to only join it. Nodes might still move away from the original AC, giving rise to different ACs which are time-synchronized with one another. The AC merger process in this scenario is left to the protocol without additional intervention; conflicting nodes (if any) enter a recontention period for new available slots.

6.1. Data Survivability in the AoI.

6.1. Data Survivability in the AoI. Our first metric of relevance is data survivability within the AoI. To this end, we generate different contact traces with different seeds employing the parameters evidenced in Table 1. The AoI is a circle within the simulation world and nodes possessing a replica of the data once outside the AoI discard the data along with the AC related information and are marked as inactive. We study the system behavior by imposing some control over the in/outflow of nodes leaving and entering the AoI.

To this end, the nodes that fall outside the AoI before simulation starts are marked as inactive. Inactive nodes become active when entering the AoI and the number of activated ones depends on the flow of nodes that have left the AC. In order to enforce this in/out policy within the AoI, we take a snapshot of the simulation world every 5 s and control the inflow depending on the outflow of the last snapshot.

We study the data survivability with varying population size and different in/out policies. Each scenario is simulated 20 times employing different mobility seeds in order to increase the confidence of the results. The considered data size along with the control data fits inside a single BCH; hence there is no need for extra slot reservation. The results are shown in Table 2.

Table 2: Average AC survival times in minutes with varying population size and different in/outflow policy.

	Trace	1 : 1	1 : 2	1 : 3	1 : 4	1 : 5	Departure
				Population size 10			
Average	48.53	39.05	22.11	20.11	10.1	7.24	6.2
Median	42.12	30.55	18.42	17.23	6.56	4.54	4.3
				Population size 15			
Average	52.12	49.11	40.2	30.23	25.23	20.53	6.3
Median	43.5	40.57	33.47	23.47	18.34	15.45	4.5
				Population size 20			
Average	55.21	51.56	48.45	33.45	27.57	25.16	6.55
Median	53.32	49.06	46.45	32.29	25.54	23.46	5.5
				Population size 25			
Average	60	59.51	54.11	50.01	45.55	38.51	7.02
Median	60	56.53	50.37	44.4	39.5	30.36	4.5
				Population size 30			
Average	60	60	58.11	55.12	51.34	48.54	6.76
Median	60	60	55.45	50.7	49.23	40.32	7.3
				Population size 35			
Average	60	60	60	60	58.12	49.11	7.54
Median	60	60	60	60	52.18	44.5	8.12

The scenario denoted with *Trace* serves as the benchmark solution where no in/outflow policy is applied; that is, the mobility dynamics inside the AoI are those exhibited by the original trace. The last column (*Departure*) denotes the average time interval that the producer nodes left the AoI for each configuration run.

As it is shown by the outcome, the AC is able to provide a margin of profit which decreases when the control flow policy becomes more aggressive. Overall the obtained results evidence the importance of a mechanism guaranteeing data availability in the AoI and at the same time show that AC serves its purpose. An increase in the node population size corresponds to an increase of data lifetime. Similarly, when a 1 : 6 policy is enforced the protocol is not as capable at counteracting the effects of nodes departing the AoI. This trend is exhibited in all the mobility traces. However, even in this case we are able to provide a margin of profit when compared to a scenario where no cooperation is involved.

6.2. Controlled Data Distribution. A homogeneous data distribution eases data accessibility in the AoI. As discussed, this control is enforced by cluster intersection nodes whenever conditions are met. To measure the benefits of this feature we contrast the approach with the one where no distribution control is enforced. To this end, we have orchestrated a set of simulations, 10 for each configuration run, with the parameters evidenced in Table 3.

We point out that, in the scenario where no distribution control applies, the data does not survive the end of the trace in 3 different runs with a data lifetime of 2.45, 5.45, and 6.34 min. We do not consider this particular runs when averaging the distance metric reported in Figure 5. An important effect of applying distribution control is resiliency

Table 3: Settings of the controlled distribution scenario.

Simulation area	500 × 500
Mobility	[0.5, 1.5] m/s
Pause time	30 s
Trace duration	10 min
Population size	35
Attraction point	250 × 250
AoI radius	200 m

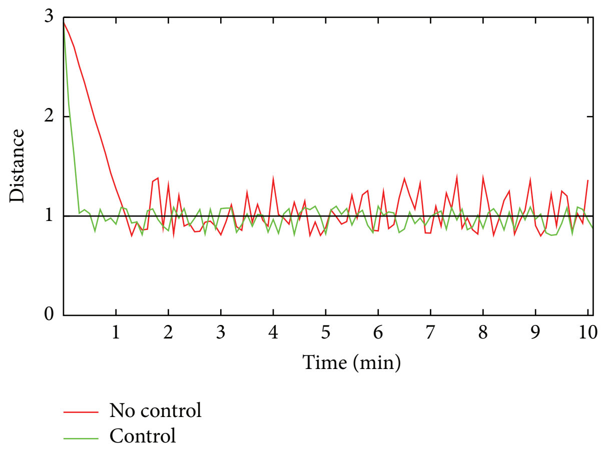


Figure 5: Comparison of the strategy employing a controlled distribution approach versus the default one. A Min/Max policy of 1 : 2 is employed.

to mobility when compared to the strategy where no control is applied. In the controlled approach the data are replicated more rapidly, creating a distributed number of replicas over the AoI rendering the AC data less subject to mobility of nodes in a certain areas. The distance metric for the distribution control scenario as in the graph converges to 1 more rapidly. In the case where no control is applied, converging to 1 requires more time.

The process of data replication in the no control scenario is incidental rather than intentional. Indeed, the scenarios where data is replicated in other places that differ from the initial point of entry depend on the mobility dynamics and/or the chance of cluster nodes caching the data. Also, when the data has reached a stable replica level within the AC, the no control scenario is subject to more fluctuations while the controlled scenario exhibits a more stable trend. This fluctuation difference between the two schemes stands in their *modus operandi* where in the control scenario data the replication policy is enforced in a more timely fashion while in the no control policy it depends upon node mobility physically carrying the data to newly arrived nodes.

7. Conclusion

In this paper we have presented AirCache, a solution designed to enable a floating data network in a distributed and collision-free way. In the considered scenario, users voluntarily provide data, which is then maintained in an AoI through replication among passing-by or stationary users. In our experiments, we have considered an AoI composed by mobile wireless nodes and study the system behavior under different scenarios.

Our tests have shown that AirCache can be effectively deployed to guarantee data availability in the AoI. In other words, we can push data storage and consumption close to the network edges instead of congesting Internet links even in presence of a *local scopus*. Clearly, node density in the considered AoI is a crucial factor to ensure the survivability of the system, thus requiring AirCache to be employed in areas where people tend to gather.

AirCache also includes a node election strategy to improve the global energy/memory consumption needed to maintain the data floating in the AoI. However, this strategy can be improved, for instance, by dynamically tuning the slot length depending on the mobility dynamics. We plan to add this feature as future work. Furthermore, we also plan to implement some real application on top of our system to test how it could support it; to this aim, an interesting case study would be represented by mobile games [32, 33].

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

Acknowledgments

This work was partially supported by the University of Padua through Projects CPDA151221 and CPDA137314.

References

[1] V. Cerf and M. Senges, “Taking the internet to the next physical level,” *IEEE Transactions on Mobile Computing*, vol. 49, no. 2, pp. 80–86, 2016.

[2] Cisco, *The Zettabyte Era—Trends and Analysis*, 2015.

[3] M. S. Desta, E. Hyytia, J. Ott, and J. Kangasharju, “Characterizing content sharing properties for mobile users in open city squares,” in *Proceedings of the 10th Annual Conference on Wireless On-Demand Network Systems and Services (WONS ’13)*, pp. 147–154, Banff, Canada, March 2013.

[4] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson, “People-centric urban sensing,” in *Proceedings of the ACM Annual International Workshop on Wireless Internet (WICON ’06)*, New York, NY, USA, 2006.

[5] G. Smith, R. Wieser, J. Goulding, and D. Barrack, “A refined limit on the predictability of human mobility,” in *Proceedings of the 12th IEEE International Conference on Pervasive Computing and Communications (PerCom ’14)*, pp. 88–94, Budapest, Hungary, March 2014.

[6] A. Bujari, *Opportunistic data gathering and dissemination in urban scenarios [Ph.D. dissertation]*, Alma Mater Studiorum Univesity of Bologna, Bologna, Italy, 2014.

[7] C. E. Palazzi, A. Bujari, G. Marfia, and M. Roccetti, “An overview of opportunistic ad hoc communication in urban scenarios,” in *Proceedings of the 13th Annual Mediterranean Ad Hoc Networking Workshop (MedHocNet ’14)*, pp. 146–149, Piran, Slovenia, June 2014.

[8] P. Ruiz and P. Bouvry, “Survey on broadcast algorithms for Mobile Ad Hoc Networks,” *ACM Computing Surveys*, vol. 48, no. 1, article no. 8, 2015.

[9] C. E. Palazzi, A. Bujari, and E. Cervi, “P2P file sharing on mobile phones: design and implementation of a prototype,” in *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT ’09)*, pp. 136–140, Beijing, China, August 2009.

[10] S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni, “DTN protocols for vehicular networks: an application oriented overview,” *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 868–887, 2015.

[11] L. Pelusi, A. Passarella, and M. Conti, “Opportunistic networking: data forwarding in disconnected mobile ad hoc networks,” *IEEE Communications Magazine*, vol. 44, no. 11, pp. 134–141, 2006.

[12] A. Bujari, C. E. Palazzi, D. Maggiorini, C. Quadri, and G. P. Rossi, “A solution for mobile DTN in a real urban scenario,” in *Proceedings of the IEEE Wireless Communications and Networking Conference Workshops (WCNCW ’12)*, pp. 344–349, Paris, France, April 2012.

[13] A. Bujari, S. Gaito, D. Maggiorini, C. E. Palazzi, C. Quadri, and G. P. Rossi, “Delay tolerant networking over the metropolitan public transportation,” *Mobile Information System*, In press.

[14] R. K. Ganti, F. Ye, and H. Lei, “Mobile crowdsensing: current state and future challenges,” *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[15] J. Scott, P. Hui, J. Crowcroft, and C. Diot, “Haggle: a networking architecture designed around mobile users,” in *Proceedings of the 3rd International Conference on Wireless on Demand Network Systems and Services (WONS ’06)*, pp. 78–86, Les Menuire, France, January 2006.

[16] S. Wood, J. Mathewson, J. Joy et al., “Iceman: a practical architecture for situational awareness at the network edge,” Tech. Rep., 2013.

[17] I. Baev, R. Rajaraman, and C. Swamy, “Approximation algorithms for data placement problems,” *SIAM Journal on Computing*, vol. 38, no. 4, pp. 1411–1429, 2008.

[18] L. Yin and G. Cao, “Supporting cooperative caching in ad hoc networks,” *IEEE Transactions on Mobile Computing*, vol. 5, no. 1, pp. 77–89, 2006.

[19] M. Fiore, F. Mininni, C. Casetti, and C.-F. Chiasserini, “To cache or not to cache?” in *Proceedings of the IEEE INFOCOM*, pp. 235–243, IEEE, Rio de Janeiro, Brazil, April 2009.

[20] S.-B. Lee, S. H. Y. Wong, K.-W. Lee, and S. Lu, “Content management in a mobile ad hoc network: beyond opportunistic strategy,” in *Proceedings of the IEEE INFOCOM*, pp. 266–270, Shanghai, China, April 2011.

[21] E. Hyytiä, J. Virtamo, P. Lassila, J. Kangasharju, and J. Ott, “When does content float? Characterizing availability of anchored information in opportunistic content sharing,” in *Proceedings of the (INFOCOM ’11)*, pp. 3137–3145, Shanghai, China, April 2011.

[22] J. Virtamo, E. Hyytiä, and P. Lassila, “Criticality condition for information floating with random walk of nodes,” *Performance Evaluation*, vol. 70, no. 2, pp. 114–123, 2013.

[23] S. Ali, G. Rizzo, V. Mancuso, V. Cozzolino, and M. A. Marsan, “Experimenting with floating content in an office setting,” *IEEE Communications Magazine*, vol. 52, no. 6, pp. 49–54, 2014.

[24] F. Borgonovo, A. Capone, M. Cesana, and L. Fratta, “Adhoc: a new, flexible and reliable MAC architecture for ad-hoc

networks," in *Proceedings of the IEEE Wireless Communications and Networking*, pp. 965–970, New Orleans, La, USA, 2003.

[25] G. Marfia, A. Amoroso, M. Roccetti, and G. Pau, "Safe driving in LA: report from the greatest intervehicular accident detection test ever," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 2, pp. 522–535, 2013.

[26] A. Amoroso, G. Marfia, and M. Roccetti, "Going realistic and optimal: a distributed multi-hop broadcast algorithm for vehicular safety," *Computer Networks*, vol. 55, no. 10, pp. 965–970, 2011.

[27] J. Liu, F. Ren, L. Miao, and C. Lin, "A-adhoc: an adaptive real-time distributed MAC protocol for vehicular Ad Hoc networks," *Mobile Networks and Applications*, vol. 16, no. 5, pp. 576–585, 2011.

[28] https://www.nsnam.org/documentation/.

[29] M. Fazio, C. E. Palazzi, S. Das, and M. Gerla, "Facilitating real-time applications in VANETs through fast address auto-configuration," in *Proceedings of the 4th Annual IEEE Consumer Communications and Networking Conference (CCNC '07)*, pp. 981–985, Las Vegas, Nev, USA, January 2007.

[30] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proceedings of the ACM Workshop on ns-2: the IP Network Simulator (WNS2 '06)*, New York, NY, USA, 2006.

[31] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "Bonnmotion: a mobility scenario generation and analysis tool," in *Proceedings of the Annual International Conference on Simulation Tools and Techniques (SIMUTools '10)*, pp. 1–10, Malaga, Spain, March 2010.

[32] M. Furini, "Mobile games: what to expect in the near future," in *Proceedings of the GAMEON Conference on Simulation and AI in Computer Games*, Bologna, Italy, 2007.

[33] M. Gerla, D. Maggiorini, C. E. Palazzi, and A. Bujari, "A survey on interactive games over mobile networks," *Wireless Communications and Mobile Computing*, vol. 13, no. 3, pp. 212–229, 2013.