Framework for Smart City Applications Based on Participatory Sensing

R. Szabó*‡, K. Farkas*‡, M. Ispány§, A.A. Benczúr§‖, N. Bátfai§, P. Jeszenszky§, S. Laki*¶, A. Vágner§, L. Kollár§, Cs. Sidló§‖, R. Besenczi§, M. Smajda§, G. Kövér§, T. Szincsák§, T. Kádek§, M. Kósa§, A. Adamkó§, I. Lendák*
B. Wiandt‡, T. Tomás*‡, A. Zs. Nagy*‡, G. Fehér*‡
*Inter-University Centre for Telecommunications and Informatics, Debrecen, Hungary
‡Budapest University of Technology and Economics, Budapest, Hungary
§ University of Debrecen, Hungary
¶Eötvös Loránd University, Budapest, Hungary
‖Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
Corresponding author: farkask@hit.bme.hu

Smart cities offer services to their inhabitants which make everyday life easier beyond providing a feedback channel to the city administration. For instance, a live timetable service for public transportation or real-time traffic jam notification can increase the efficiency of travel planning substantially. Traditionally, the implementation of these smart city services require the deployment of some costly sensing and tracking infrastructure. As an alternative, the crowd of inhabitants can be involved in data collection via their mobile devices. This emerging paradigm is called mobile crowd-sensing or participatory sensing. In this paper, we present our generic framework built upon XMPP (Extensible Messaging and Presence Protocol) for mobile participatory sensing based smart city applications. After giving a short description of this framework we show three use-case smart city application scenarios, namely a live transit feed service, a soccer intelligence agency service and a smart campus application, which are currently under development on top of our framework.

Keywords—Smart City, Participatory Sensing, XMPP, Public Transport, Soccer, Smart Campus

I. Introduction

The development of cites are not any more solely depend on the city's basic (physical) infrastructure but more and more correlated to the availability of information and communication technologies (ICT) supporting knowledge sharing about cities. More formally, Gartner defined smart cities as "multiple sectors cooperating to achieve sustainable outcomes through the analysis of contextual real-time information shared among sector-specific information and operational technology systems". In this context, the real-time information is *big data* and the contextual sharing system is generally believed to be realized by the Internet of Things. Internet of Things is visioned to become true with over 50 billion connected devices around 2020.

In the meantime, with the proliferation of smart-phones more and more computing and sensing power becomes available at the hands of urbanites. If the community finds incentives (good services) for urbanites to participate in context sharing, or often called crowd-sourcing[1], then combined with big data analytics we can realize a vision similar to the Internet of Things based smart cities (see Fig. 1).

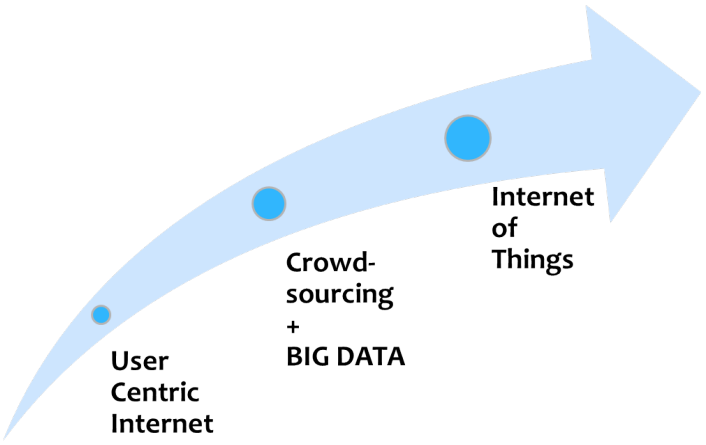

Fig. 1. Internet evolution: from user centric Internet to Internet of Things

However, a typical crowd-sourcing application today has two application specific components: i) one at the user's device and ii) another one in the cloud [1]. This results in many parallelism, unnecessary developments and slow application innovation cycle. We envisioned a framework that separates the application logics from the core *communication* and *big data analytic* functions and thus results in an architecture where innovation can be done at the end systems focusing on the application and presentation layers. Introducing such a framework could boost the developments similarly to the innovation at the application front enabled by IP.

To allow independent application innovation the framework must transparently pass extra information, by which the basic services can be extended. This yields to a generic information modeling approach based on some extensible messaging service. Moreover, by the means of participatory sensing we have

[1]We use the terms *crowd-sourcing*, *crowd-sensing*, *urban sensing* and *participatory sensing* interchangeably in this paper.

978-1-4799-1546-0/13/$31.00 ©2013 IEEE

to aim at decoupling information producers and information consumers in space, time and synchronization.

Due to the lack of such a unifying, open, extensible and producer/consumer decoupled framework as an enabler for service innovation, and the emerging popularity of considering crowd-sensing for data collection we started to work on a system for participatory sensing based smart city applications. Our system, in line with the value chain of crowd-sourcing (see Fig. 2), comprises

- a communication and extensible information modeling and messaging framework;
- local and cloud based analytics;
- and pilot applications.

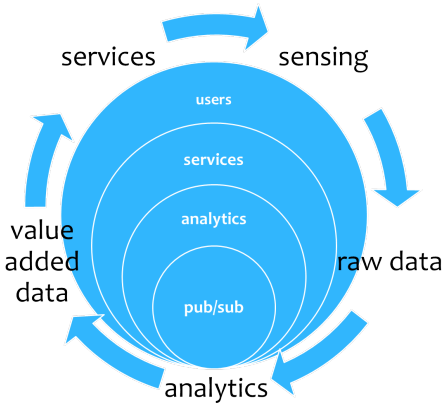

Fig. 2. Value chain: from sensing to value added services

In order to exploit available best practices, we investigated the reuse of the basic Extensible Messaging and Presence Protocol (XMPP) [2] and its publish-subscribe (pubsub) service [3], and built our system upon them. This approach provides an absolutely generic information and communication framework, which extended by analytics can offer a unifying solution for developing participatory sensing based applications.

The rest of the paper is structured as follows. In Sec. II, we describe our framework. In Sec. III, we show some application examples, such as a live transit feed service, a soccer intelligence agency service and a smart campus application, we have been developing as use-cases and proof of our concept. Finally, in Sec. IV we summarize our work with a short insight to our future plans.

II. Framework for Participatory Sensing Based Smart City Applications

In this section, we describe shortly our XMPP-based publish-subscribe architecture to aid the development of participatory sensing based smart city applications. Moreover, we discuss the analitics related issues of such applications.

A. XMPP-based Publish-Subscribe Architecture

The basic communication principle of the most crowd-sensing based applications fits well with the publish-subscribe communication scheme, as users participate in data collection (publish) and consume the services updated on the basis of the collected data (subscribe). Thus, in our architecture we use a generic publish-subscribe communication model for implementing interactions. In this model, we define three roles, like *Producers*, *Service Providers* and *Consumers* (see Fig. 3). These entities interact with each other via the core service, which consists of event based pubsub nodes.

Producers: In our model, the Producers act as the original information sources and play a central role in data collection. They are users who contribute their mobiles' sensor data, thus producing raw data streams.

Consumers: The Consumers are the beneficiaries of the provided services. They enjoy the value of the collected, analized, extended and disseminated information in the service. Sometimes the users participating in the service can also act as Producers. In this case, we call them as *Prosumers*.

Service Providers: The Service Providers introduce added value to the raw data collected by the crowd. Hence, they intercept and extend the information flow between Producers and Consumers. Service Providers can play several roles at the same time, as they collect (Consumer role), store and analyze Producers' data to offer (Service Provider role) value added service.

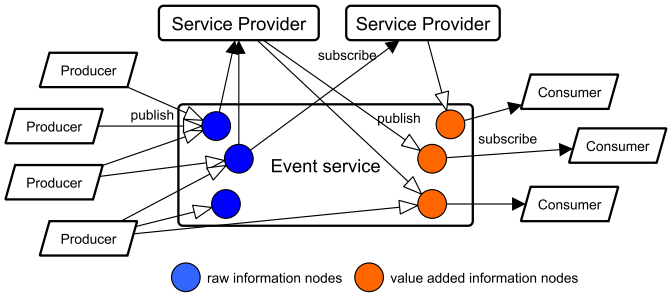

Fig. 3. A publish-subscribe scheme based crowd-sourcing model

In our model, depicted in Fig. 3, Producers are the source of original data by sensing their environment. They publish (marked with empty arrowheads) the collected information to event nodes (raw information nodes marked with blue). On the other hand, Service Providers intercept the collected data by subscribing (black arrowheads) to raw event nodes and receiving information in an asynchronous manner. They extend the crowd sensed data with their own information or extract cleaned-up information from the raw data to introduce added value to Consumers. Moreover, they publish their service to different content nodes. Consumers who are interested in the reception of the added value/service just subscribe to the appropriate content node(s) and collect the published information in an asynchronous manner.

We can directly map this model to the XMPP publish-subscribe service according to the following (see Fig. 4):

- To gather Producers' data Service Providers establish raw pubsub data nodes for their offered services.
- Also Consumers, with appropriate node access rights, can freely publish their collected data to the corre-

sponding nodes, but only the owner or other affiliated Consumers can retrieve this information.

- Producers can publish the collected data or their annotations to the raw data nodes at the XMPP server if they have appropriate access rights.
- Using the pubsub subscription service, Service Providers collect the published data and introduce such a service structure for their added value which makes appropriate content filtering possible for their Consumers.
- Prosumers publish their sensor data or annotations into and retrieve events from XMPP pubsub nodes.
- Service Providers subscribed to raw pubsub nodes collect, store, clean-up and analyze data and extract/derive new information introducing added value. This new information is published into pubsub nodes following a suitable structure.

The pubsub service node structure can benefit from the XMPP's aggregation feature via using collection nodes, where a collection node will see all the information received by its child nodes. Note however, that the aggregation mechanism of the XMPP's collection node is not appropriate to filter events. Hence, the Service Provider role has to be applied to implement scalable content aggregation. Fig. 4 shows the XMPP's aggregations as dark circles at the container node while empty circles represent only logical containment where intelligent aggregation is implemented through the service logic.

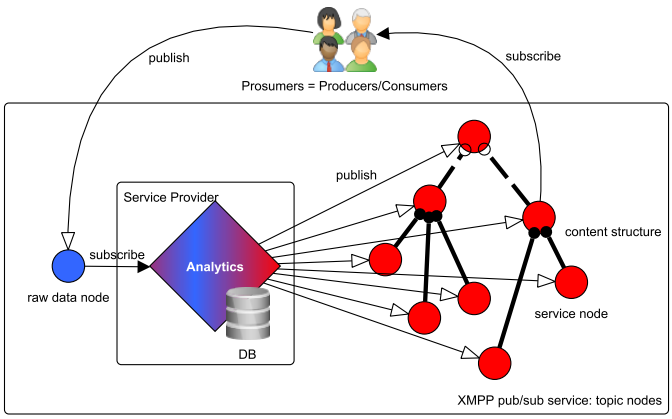

Fig. 4. Mobile participatory sensing: the publish-subscribe value chain

B. Analytics Component

The goal of the analytics component of our framework (Fig. 4) is to manage (store, clean, organize and analyze) the raw data coming from the intelligent sensors and provide valuable information for the services that will allow the active users to deal with a vast variety of situations in the city on the area, e.g., traffic, scheduling, criminality and environmental sustainability. There are generic data mining tasks in the great majority of smart systems planned for urban computing. Such tasks are among others: predicting the people's activities and movements in space and time; discovering region of different functions in a city; detecting anomalous rare events in a city.

The data sets arising in smart city applications usually fall into the category of "Large-Scale Data" or "Big Data", referring to datasets whose size, velocity or variety is well beyond the ability of typical software tools to capture, store, manage, and analyze. This situation pushes towards new algorithms which are typically approximated and/or distributed improvement of the standard machine learning and data mining algorithms.

1) The Real-time Distributed Smart City Analytics Layer: Real time analytics for traffic and mobility, as opposed to offline tasks such as city planning, requires processing the incoming data stream without first storing, cleaning and organizing it in any sense. Scalability and low latency are crucial factors that require new algorithms (typically, approximated or distributed) and new computational frameworks (e.g., MapReduce [4], NoSQL and streaming data).

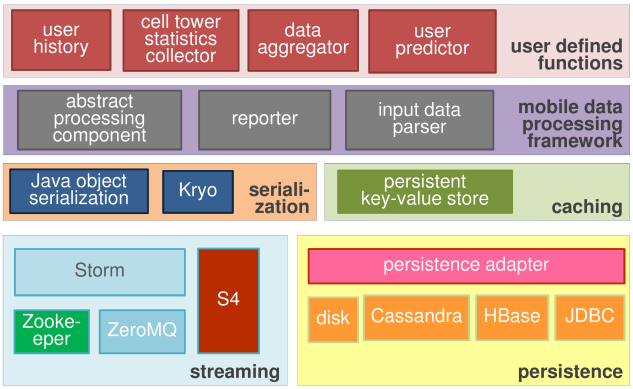

Fig. 5. Layers of our mobility prediction architecture: the streaming framework (bottom left), persistence components (bottom right), and the custom analytics (top)

We implemented a distributed streaming smart city analytics environment that reaches remarkably high throughput with low latency using a properly designed streaming architecture. Fig. 5 depicts the layered architecture that enables easy model implementation while relying on the scalability, low latency and fault tolerance of the underlying distributed data processing software. In this architecture, we are free to choose from existing frameworks such as Storm [5] or S4 [6]. Since these frameworks may loose history information when their processing modules restart after failures, we built a generic persistence module (bottom right side of Fig. 5). We deploy Cassandra [7] due to its high throughput writing capabilities and memcached [8] for its efficiency.

When a node fails, a new node is initialized with the stored states of the affected processing components. According to the guarantees of Storm, the lost packets are processed again. Fig. 6 shows how node failures affect the overall performance. We can observe rapid recoveries, despite of the large number of failing nodes, the overall performance remains predictable.

2) Anomaly Detection for Smart City Applications: Anomaly detection refers to detecting patterns in a given data set that do not conform to an established normal behavior. The detected patterns are called anomalies (outlier, surprise

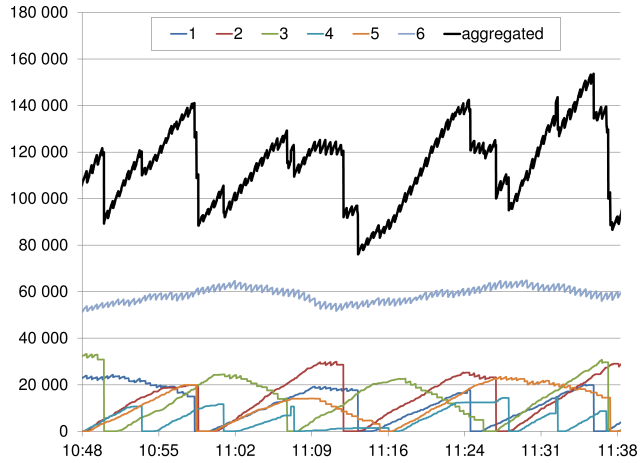

Fig. 6. Throughput (number of records per second) as the function of the time passed (absolute times), with six nodes, each with a spout. One node works continuously, while the others occasionally stop

deviation etc.) and translate to actionable information in smart city applications. For example, detecting a possible traffic jam, finding forthcoming fault in the pattern of players in soccer that results in penalty, or recognition of rare significant departure of class scheduling in daily routine of students at the campus are all subject to anomaly detection. If we could predict such events in advance it would also be possible for people to avoid them. We investigated the capabilities of various machine learning algorithms to predict anomalous events in Smart City. These algorithms, namely Recursive Partitioning (Decision Tree), Naive Bayes (NB), Support Vector Machines Classifiers (SVM-C), k-Nearest Neighbors (kNN), Random Forest (RF) and LDA/QDA, have been succesfully applied for predicting anomalies in complex softwares (see [9]) which are comparable to our framework. These techniques have also been studied in the analyses of traffic streams using GPS traces (see [10] and [11]).

3) Social Networks for Smart City Applications: Various social media sites (Twitter, Facebook, Google+, etc.) have emerged in the past decade, bringing a radically new way of communication. These platforms enable users to share and exchange information on real-world events from local to world scale. These events can vary on a wide range: popular concerts, festivals, demonstrations, accidents, traffic jams, disasters, etc. They differ from each other in many aspects, but they have in common that they affect the life of a great mass of people.

These information sources could be exploited by smart city applications, providing their user with useful information on popular events, festivals, pop concerts, etc. Furthermore, the posted messages (e.g. on Twitter) can be used to detect unpredicted events [12] like earthquakes, fires, or other disasters in real-time. In this model, the users of social networks are considered as sensors. However, real-time processing of the unstructured data available in social networks poses many challenges which have to be addressed during the design of smart city applications.

III. Use-case Smart City Application Examples

In this section, we shortly present three use-case smart city application examples, we have been developing currently, to demonstrate the usage of our framework.

A. Extended Transit Feed Service

Public transportation is an application domain which has great potentials to introduce crowd-sourcing based services. As an example, in many cities arrival information of public transportation lines is expected to be provided and updated in real-time. However, if an appropriate infrastructure, which is usually expensive, is not in place participatory sensing can come into remedy.

A crowd-sourcing based smart public transport application can offer a service which combines static transit information with live data collected by the crowd. For example, a smart transit application can provide, beyond the static schedule information based on General Transit Feed Specification (GTFS) [13] data, an enhanced timetable service with real-time refinements as people start using it.

Such crowd-sourcing based transport application can also be applied by its users to annotate trips or to outsource some of the data collection tasks of the transit operator to support its operations. For instance, crowdedness of the public transport lines could be measured by participatory sensing for different sections of the route and times of the day, week, month and season.

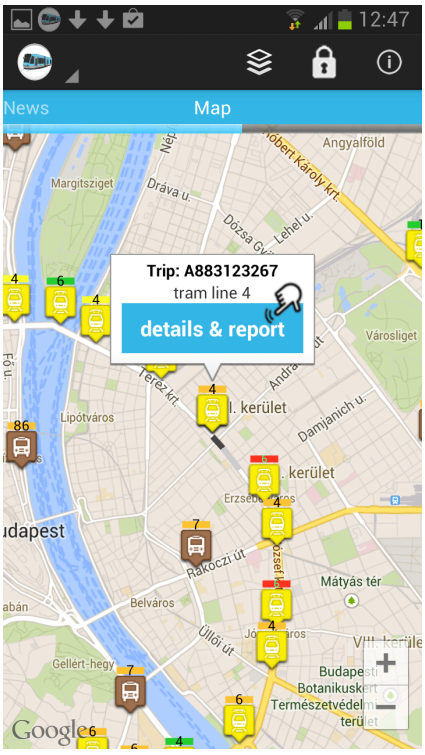

Fig. 7. User Interface of our extended transit feed service

For validation purposes and get more insight into crowd-sourced transport services we have been developing, on top of our framework, a participatory sensing based smart public transport application for Android platform using the aSmack API [14]. The Android client can receive the transit feeds, GTFS data and its refinements, from the XMPP server and visualize them on a map as shown in Fig. 7. Additionally, the

user is able to report incidents (e.g., delayed arrival/departure) or annotate the crowdedness status of any vehicle, which is sent in real-time to the appropriate pubsub node at the XMPP server. Clients subscribed to these nodes receive real-time crowdedness data besides arrivals information. In Fig. 7, vehicles are colored in red, orange and green according to high, medium and low crowdedness, while no color indicates that no participatory data is available. In this scenario, the users' quality of experience increases as more and more people start contributing to sensing the crowdedness status.

B. Soccer Intelligence Agency

The project, called SIA (Soccer Intelligence Agency, or its former name, Distributed Supporter Avatar), is designed to support distributed data collection in sport events, particularly in the context of soccer. The major novelty of this initiative is that data is gathered by supporters and fans of football. The SIA System is available in two editions: SIA Mobile and SIA Desktop Player. SIA Mobile focuses on the online (real-time) supporters watching the match in the stadium. SIA Desktop Player is a media player program that enables offline users to annotate matches more specifically than a normal media player can do. The functional icons and the splash screen of the SIA Desktop Player can be seen in Fig 8.

Fig. 8. SIA Desktop Player: the functional icons and the splash screen

The main design feature of the SIA applications was that the annotation process does not require too much effort from the users, because they would like to principally enjoy watching football matches. SIA users can collect data about actions of players, coaches and referees. For example, in the "favourite player's passes" part of the SIA Mobile, only the fire soft-button must be pressed if the ball is forwarded or received by the observed player.

Both of the SIA applications use the Smack API [15] (aSmack [14] in case of the mobile client) for XMPP based communication. Every user connects to our XMPP server, and by firing one of the softkeys an XML (Extensible Markup Language) message is sent to our Master Client (this client stores every message). On the other hand, if a user connects to our server, the Master Client sends information about matches, also in XML format. These XML messages are called SIA Messages.

SIA is under active development by a team of teachers and students at University of Debrecen. Developers who contributed to the SIA but are not authors are named in the Acknowledgement section. The implementation details of SIA can be found in the manuscript [16].

C. Smart Campus

A university campus is a good candidate to apply participatory sensing since its thousands of students are very active on social networks. Our aim was to build services that can be used to draw conclusions regarding the operation of the community, and the derived information can appear as a new service in the community (by using appropriate analytics), thus having a beneficial feedback to its operation.

An application that aggregates useful information from several sources has been developed and published as a Web service. Such information include timetable, various deadlines defined in the academic calendar, open hours of the faculty administration and staff, etc. Based on the provided data of the aggregator service, users can subscribe to events they are interested in. These interests are recorded by an Android based application and are subject to various data mining operations that can result in new services (like suggesting a practical order or activities to be done) which can be offered to the crowd. A prototype Web application working as a consumer of the Web service has also been developed (available only in Hungarian at the moment, see Fig. 9) that can be used to find the current courses held at the Campus.

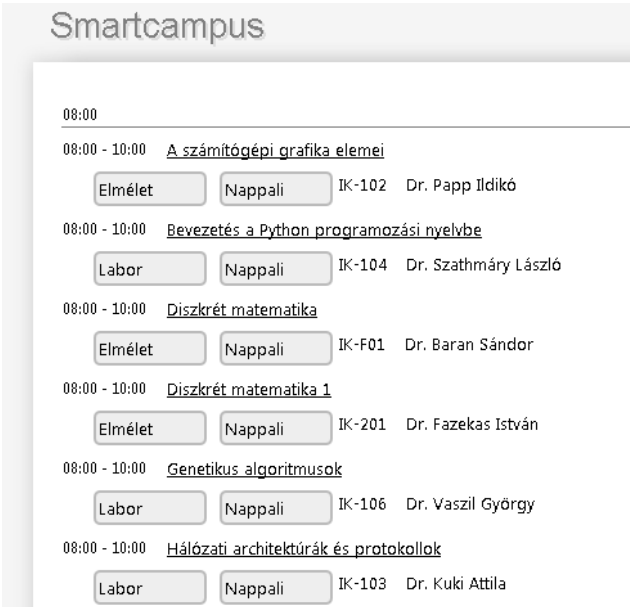

Fig. 9. Web application showing the current courses at the Campus

Another mobile application, suitable for recording data provided by the various sensors of a mobile device, have also been developed. Users get full control over data recording: they can switch it on or off whenever they want. When switched on, the application running as a background process collects data (e.g., fine-grained or coarse-grained position, temperature, gyroscope and other built-in sensors) in XML format which are then automatically sent to an XMPP server where the collected data can be processed using data mining techniques. The XML document contains the recorded data with a timestamp along with a header used for identifying the mobile device. Data sampling is 1 ms by default.

This application is working on Android 2.2+ which means that 99% of Android users can use it. Permanent Internet connection is not required, data gathered off-line will be sent to the XMPP server upon connection. The application consists of two services and an activity. One service is responsible for the recording (see Fig. 10), the other is for the uploading, and the activity handles the interaction with the user. The analytics module deployed at the server-side can be used for analyzing the traffic in the campus (or a city), finding frequent routes, suggesting faster routes, etc.

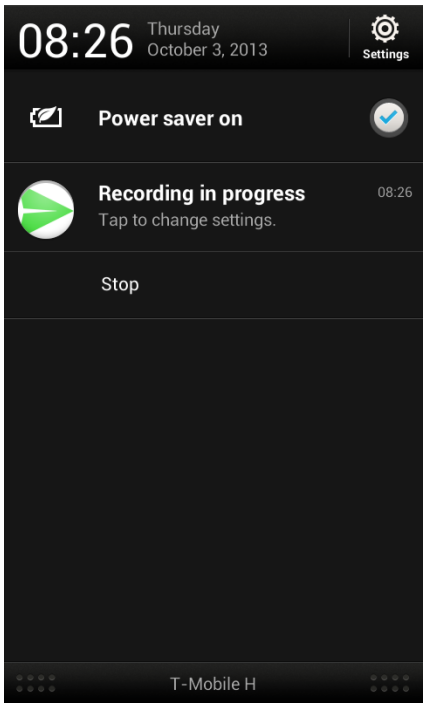

Fig. 10. Data collector application for Android

IV. Summary

In this paper, we presented our framework, based on the publish-subscribe communication model and the use of XMPP, as the core of a unifying open architecture for crowd-sourcing based smart city applications. XMPP is already established, standardized, freely available, extendible, supports the publish-subscribe communication model. Moreover, we discussed the analitics related issues of such smart city applications and showed three use-case smart city application scenarios which are currently under development on top of our framework.

As future work, we plan to evaluate the performance and scalability properties of our framework and stress test our XMPP based architecture. Moreover, we intend to implement some further application scenarios, such as smart parking, pothole detection or indoor/outdoor navigation and collect real crowd data via our pilot applications.

Acknowledgment

The authors would like to thank the former and actual members of the working group “Intelligent supporter” especially Balázs Kóti, József Zákány, Roland Dóczi and the students of the course of “Java casestudies” at the University of Debrecen for participation in the development of SIA. Special thanks to Balázs Kóti for creating the SIA logo and icons.

The publication was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund. This work has been partially supported by the KIC ICTLabs under the activity 13064 CityCrowdSource of the action line Digital Cities. The publication was supported by the KTIA_AIK_12-1-2013-0037 project. The project is supported by Hungarian Government, managed by the National Development Agency, and financed by the Research and Technology Innovation Fund. Károly Farkas has been partially supported by the Hungarian Academy of Sciences through the Bolyai János Research Fellowship.

References

[1] R. Ganti, F. Ye, and H. Lei, “Mobile Crowdsensing: Current State and Future Challenges,” *IEEE Communications Magazine*, pp. 32–39, Nov. 2011.

[2] P. Saint-Andre, “Extensible Messaging and Presence Protocol (XMPP): Core,” RFC 6120 (Proposed Standard), Internet Engineering Task Force, Mar. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6120.txt

[3] P. Millard, P. Saint-Andre, and R. Meijer, “XEP-0060: Publish-subscribe,” XMPP Standards Foundation, Draft Standard XEP-0060, Jul. 2010. [Online]. Available: http://xmpp.org/extensions/xep-0060.html

[4] T. White, *Hadoop: The Definitive Guide*. Yahoo Press, 2010.

[5] J. Leibiusky, G. Eisbruch, and D. Simonassi, *Getting Started With Storm*. Oreilly & Associates Incorporated, 2012.

[6] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, “S4: Distributed stream computing platform,” in *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*. IEEE, 2010, pp. 170–177.

[7] A. Lakshman and P. Malik, “Cassandra: A structured storage system on a p2p network,” in *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*. ACM, 2009, pp. 47–47.

[8] B. Fitzpatrick, “Distributed caching with memcached,” *Linux journal*, vol. 2004, no. 124, p. 5, 2004.

[9] J. Alonso, L. Belanche, and D. Avresky, “Predicting Software Anomalies Using Machine Learning Techniques,” in *10th IEEE International Symposium on Network Computing and Applications*, 2011, pp. 163–170.

[10] L. Pang, S. Chawla, W. Liu, and Y. Zheng, “On Mining Anomalous Patterns in Road Traffic Streams,” in *7th International Conference on Advanced Data Mining and Applications*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 237–251.

[11] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, and S. L. L. Sun, “iBAT: Detecting Anomalous Taxi Trajectories from GPS Traces,” in *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011, pp. 99–108.

[12] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes twitter users: real-time event detection by social sensors,” in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 851–860.

[13] Google Inc., “General Transit Feed Specification Reference.” [Online]. Available: https://developers.google.com/transit/gtfs/reference/

[14] “aSmack API.” [Online]. Available: https://github.com/Flowdalic/asmack/

[15] “Smack library.” [Online]. Available: http://www.igniterealtime.org/projects/smack/

[16] N. Bátfai, R. Szabó, P. Jeszenszky, J. Komzsik, A. Mamenyák, R. Besenczi, J. Zákány, R. Dóczi, C. Székelyhídi, M. Smajda, B. Kóti, G. Kövér, E. Bátfai, K. Farkas, M. Ispány, and G. Terdik, “Crowd, Cloud and Public Resource Computing Models in Soccer with Case Studies (work in progress),” 2013.