Publish/Subscribe Middleware for Energy-Eff cient Mobile Crowdsensing

Ivana Podnar Žarko
University of Zagreb
Faculty of Electrical
Engineering and Computing
Zagreb, Croatia
ivana.podnar_zarko@fer.hr

Krešimir Pripužić
University of Zagreb
Faculty of Electrical
Engineering and Computing
Zagreb, Croatia
kresimir.pripuzic@fer.hr

Aleksandar Antonić
University of Zagreb
Faculty of Electrical
Engineering and Computing
Zagreb, Croatia
aleksandar.antonic@fer.hr

UbiComp'13 Adjunct, September 8–12, 2013, Zurich, Switzerland.

http://dx.doi.org/10.1145/2494091.2499577

Abstract

In this paper we focus on mobile crowdsensing applications for community sensing where sensors and mobile devices jointly collect and share data of interest to observe and measure phenomena over a larger geographic area. Such applications, e.g., environmental monitoring or crowdsourced traffic monitoring, involve numerous individuals that on the one hand continuously contribute sensed data to application servers, and on the other hand consume the information of interest to observe a phenomenon typically in their close vicinity. Energy-efficient and context-aware orchestration of the sensing process with data transmission from sensors through mobile devices into the cloud, as well as from the cloud to mobile devices such that information of interest is served to users in real-time, is essential for such applications, primarily due to battery limitations of both mobile devices and wearable sensors. In addition, the latency of data propagation represents their key quality measure from the user's perspective.

Publish/subscribe middleware offers the mechanisms to deal with those challenges: It enables selective real-time acquisition and filtering of sensor data on mobile devices, efficient continuous processing of large data volumes within the cloud, and near real-time delivery of notifications to mobile devices. This paper presents our

implementation of a publish/subscribe middleware system which is tailored to the requirements of mobile and resource-constrained environments with a goal to reduce the overall energy consumption in such environments, and proposes a general architecture for mobile crowdsensing applications. We demonstrate the usability of both the architecture and middleware through our application for air quality monitoring, and discuss the energy footprint of the proposed solution.

Author Keywords

Internet of Things, crowdsourcing, mobile sensors.

ACM Classification Keywords

C.2.4 [Distributed Systems]: Miscellaneous.

Introduction

The ubiquity of wearable sensors and proliferation of mobile devices with a number of built-in sensors creates a vast amount of data sources that continuously generate huge amounts of data. In the context of Internet of Things (IoT), these data sources are named *Mobile Internet-connected Objects* (MIO) as they generate sensor data streams to be relayed through mobile devices into a data cloud, while mobile devices can serve cloud-processed data of interest to users in a timely fashion. Such heterogeneous and complex environments comprising sensors, various mobile devices and the cloud, offer new opportunities for deploying innovative mobile crowdsensing applications, e.g., applications that fall within the domain of smart cities, real-time traffic monitoring, crowdsourced environmental monitoring, ambient assisted living, or social sensing.

Mobile crowdsensing poses a number of research challenges due to 1) resource limitations related to energy, bandwidth and computation, 2) privacy, security, and data integrity, and 3) both local and aggregate data analytics [4]. In this paper we tackle the following two challenges: energy-efficiency at the level of MIOs, and the orchestration of local and aggregate cloud-based continuous processing of sensor data. To that end, both time and space-related coordination of the sensing process with data transmission to/from the cloud is vital, as the overall energy consumption on a mobile device, including sensing and communication cost, becomes an important efficiency measure. Moreover, the latency of data propagation related to the entire life-cycle of a sensor reading, from sensing process to delivery of alerts to mobile devices, needs to be carefully optimized since it represents a vital quality measure from users' perspective.

Publish/subscribe middleware (or event-based middleware) offers the mechanisms for efficient continuous processing and filtering of sensor data in largely-distributed and heterogeneous environments. Publish/subscribe is a well-established continuous processing and communication infrastructure for push-based data dissemination from data sources (*publishers*) to data destinations (*subscribers*) such that data distribution is flexible, i.e., it takes into account user preferences expressed as *subscriptions* (or continuous queries) [9]. It can be used to filter out the data which is not of interest at publication time on MIOs, and thus to greatly reduce their energy consumption due to reduced frequency of data transmissions into the cloud. Mobile publish/subscribe solutions have been proposed many years ago [20, 11], mainly for environments with frequently disconnected clients (publishers and subscribers) when mobile phones did not posses adequate computing resources to process data objects. One could argue that these were preliminary solutions whose potential can nowadays be fully exploited in environments

comprising on one hand smartphones and tablets, and on the other huge computing resources offered by the cloud. Thus we leverage existing concepts and solutions stemming from the field of publish/subscribe and event-based systems to design an energy-efficient mobile crowdsensing solution.

We describe the properties of *Mo*bile *P*ublish/*S*ubscribe (MoPS) middleware adequate for mobile crowdsensing such that the data transmission from MIOs to the cloud, and possibly also the sensing process, is suppressed when there are no active subscriptions in the system which could match the sensed data, or sufficient data density for a certain area is achieved. For example, if a number of users traveling in the same bus are using the same crowdsourcing application, we can shut down the redundant sensing process on their mobile phones and greatly contribute to battery power-saving. This approach is complementary to the use of supplementary low-power processors for continuous sensing tasks as proposed in [13] or the problem of determining an optimal sensing frequency for a public transport network [19, 15] where sensor routes are known a priori. Note that in this paper we do not focus on the problem of identifying an optimal sampling frequency for mobile sensors with arbitrary movement as it is done in [16], but rather propose an architectural style and appropriate usage of publish/subscribe concepts with a goal to achieve energy-savings on mobile devices and sensors.

Our contributions can be summarized as follows: We describe the properties of our publish/subscribe middleware for mobile crowdsensing which orchestrates the existing publish/subscribe concepts for mobility-aware and energy-efficient data acquisition, filtering and delivery. Next, we propose a general architecture for mobile crowdsensing and present our implementation of an air quality monitoring application which follows the general guidelines of the proposed architecture. The platform for air quality monitoring integrates data from wearable gas sensors through mobile devices and MoPS middleware. It also includes i) a web application that displays sensor readings and heatmap visualization of sensed values over time, and ii) client application for the Android OS which transmits sensor readings to the platform back-end system, and displays alerts in accordance with user-defined subscriptions. Finally, we discuss potential energy savings of our approach.

The paper is structured in the following way: Section 1 defines the publish/subscribe model and architecture for mobile crowdsensing while Section 2 provides the implementation details of our air monitoring application. In Section 3 we provide the initial evaluation results related to energy savings, Section 4 provides an overview of related work, and Section 5 concludes the paper.

Mobile Publish/Subscribe and Crowdsourcing Architecture

Publish/subscribe middleware for mobile crowdsensing has to provide the means for *energy-efficient data collection and dissemination* in IoT environments where mobile devices are used as gateways for collecting and transmitting sensor data into the cloud, while at the same time mobile devices receive the data of interest from the cloud. In contrast to existing centralized database solutions which typically send all sensed data into the cloud, a publish/subscribe-based solution can enable flexible and controllable acquisition of data and its subsequent transmission into the cloud only in situations when the sensed data is indeed required by the application. In other words, the data should be produced

and transmitted to the back-end systems only if it is valuable, e.g., there is current interest by system users to be alerted about certain events, or the data is needed for future data-mining tasks.

Our solution enables *content-based filtering of sensor data on mobile devices* before its transmission into the cloud depending on context, e.g., current data needs specified by application users, sensing coverage for a specific application, or even bandwidth of the available communication link. Moreover, it can suppress the sensing process on either built-in or wearable sensors. We have designed and implemented MoPS, a Java middleware library which includes an Android package with support for publishers, subscribers and mobile brokers. Similar to [14], we support a rich predicate language with an expressive set of operators for the most common data types: relational operators, set operators, prefix and suffix operators on strings, and the SQL BETWEEN operator. Hereafter we explain the MoPS model and underlying design principles.

Model. The MoPS model comprises a set of publishers, P_i, and a set of subscribers, S_j, that interact over a hierarchical two-tier publish/subscribe network composed of *mobile brokers*, MB_k, and *cloud brokers*, B_l. An example model is shown in Figure 1. Publishers, e.g., wearable or built-in sensors, *publish* data items and send them either to mobile brokers or directly to cloud brokers. Subscribers, e.g. processes on mobile devices, can activate and dismiss subscriptions by sending messages *subscribe* and *unsubscribe* to mobile or cloud brokers, which in turn use the message *notify* for push-style delivery of matching data items, i.e., items that satisfy subscription constraints, to subscriber processes. Brokers are responsible for efficient matching of data items to active subscriptions as well as their subsequent delivery to either subscribers, mobile brokers, or other cloud brokers, i.e., components that have defined matching subscriptions. Cloud brokers share the processing load to enable scalable system performance, and exchange subscriptions as well as matching data items. Subscribers and mobile brokers may connect to any cloud broker, or the cloud-based implementation may be centralized such that the initial connection to the cloud is established through a single proxy broker.

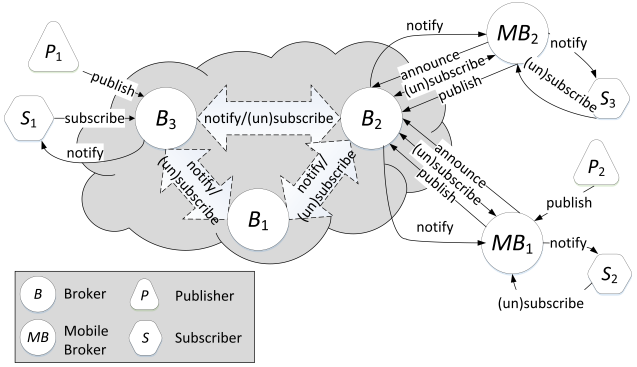


Figure 1: Mobile publish/subscribe model

The main novelty of the MoPS model compared to existing publish/subscribe solutions is the implementation of mobile brokers running on mobile devices such as smartphones and tablets. After their initial registration with cloud brokers, mobile brokers can *announce* the type of data for publishers which they represent. For example, P_2 in Figure 1 is, e.g., a wearable gas sensor detecting levels of nitrogen dioxide (NO2) and ozone (O3). After MB_1 detects P_2 because they exchange signaling information over a Bluetooth connection, MB_1 can define the type of data items to transmit to its cloud broker B_2

in the future. MB_1 sends a message *announce*(NO2, O3, x, y), where x=45.81302 and y=15.97781 represent MB_1's current geographical latitude and longitude. The reason for creating the announce message is the following: We need to activate subscriptions from cloud brokers on MB_1, but only those that can potentially match future publications created by P_2. Obviously, as it is not desirable to activate all subscriptions from the cloud on a single mobile device, the announce message is compared to existing subscriptions on B_2. For example, B_2 identifies subscription $s_i = [\texttt{NO2} > 40\mu gm^{-3}\ \texttt{AND}\ 45.81 < \texttt{lat} < 45.82\ \texttt{AND}\ 15.96 < \texttt{long} < 15.98]$ as a subscription potentially matching future publications of P_2. Thus, B_2 sends a message *subscribe* to activate s_i on MB_1. Further on, MB_1 publishes P_2's data items into the cloud, but only those that match s_i.

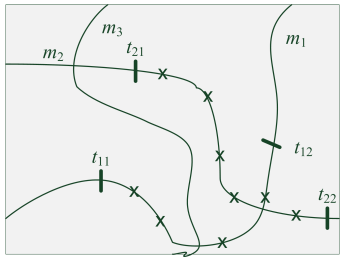


Figure 2: Movement traces and data transmissions

Energy-efficient and flexible data acquisition. By reusing the inherent features of the two-tier publish/subscribe model, we provide a flexible mechanism to control the sensing density over a predefined area covered by traces of MIOs. It requires an orchestration of the sensing process with activation of adequate subscriptions on mobile brokers, as instructed by the back-end cloud system based on the integrated crowdsensed data. If we assume the density demand is predefined for an area as required by the application logic, MIOs residing in this area during a certain time interval can be instructed either (1) to transmit the sensed data into the cloud as additional data samples are needed within this area for the particular time interval, or (2) to restrain from such transmissions since the application has already acquired sufficient data samples for the area.

This is the main mechanism for frequency reduction of data transmissions from MIOs into the cloud which has the potential to greatly reduce energy consumption on MIOs. Consider the following example in Figure 2. It depicts movement traces for three MIOs m_1, m_2 and m_3 within a certain area, and denotes time intervals $[t_{11}, t_{12}]$ and $[t_{21}, t_{22}]$ within which the two MIOs perform data transmissions into the cloud (they are marked by the symbol $\times$), while m_3 does not perform any transmissions. MIOs perform transmissions at marked places because during the two time intervals subscriptions matching the data acquired by m_1 and m_2 are active on those MIOs. This does not impose any constraints on the sensing process as it largely depends on MIO interaction with sensors in its vicinity. For example, if the sensing process is pull-based, an MIO can invoke it periodically during the subscription activity periods. If sensors are configured to perform periodic sensing, mobile brokers residing on MIOs will ignore the sensed data while it does not match any of the active subscriptions.

Let us further explain who controls the activation of subscriptions on MIOs and how the sensing density is controlled. The back-end cloud system is notified when an MIO enters the depicted geographical area since MIOs are configured to announce their available data sources when

entering the area. This requires periodic GPS positioning on MIOs which is potentially energy-greedy, but if we assume that crowdsensing applications are used for a limited period of time, this process should not represent a major obstacle for application adoption. In addition, mobile devices need to be aware of area boundaries that are important for the application logic. Since the back-end system is aware of the data samples already acquired over an observed area, it can decide whether to ship matching subscriptions to MIOs or not. In our example in Figure 2, the application logic has decided that there are sufficient measurements acquired for the depicted area from m_1 and m_2, and thus subscriptions were not forwarded to m_3.

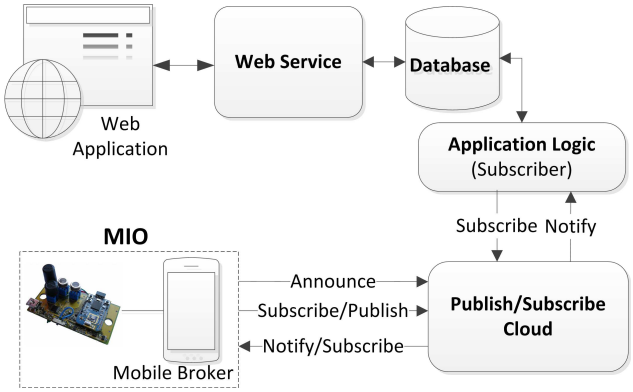


Figure 3: Generic mobile crowdsensing architecture

Architecture. Figure 3 depicts the proposed generic mobile crowdsensing architecture which uses our publish/subscribe infrastructure to integrate MIOs with back-end application-specific servers. Essential components of this architecture are the following: MIOs, publish/subscribe middleware, application back-end servers, and web service and interface.

MIOs host mobile brokers to enable filtering of data objects close to data sources. As MIOs are typically smartphones or tablets, they also run mobile applications which serve as user alert and visualization interfaces.

Application back-end servers host a vital back-end component which implements application logic and core application-specific knowledge: This component has the overall overview of a mobile crowdsensing application and its specific data needs. Thus it runs a subscriber which defines application-specific subscriptions such that the database is filled with appropriate data. For example, in our air quality monitoring application, we divide a geographical area into non-overlapping rectangles and define the maximal number of gas samples to be collected over that area during a certain time period. When an MIO enters a new area, it sends a new announce message. The application logic component defines subscriptions for each area and invalidates them when sufficient sensor readings for that area are collected. Therefore, when a MIO enters a certain area and sends an announce message since it is connected to a gas sensor in its vicinity, if there are no subscriptions for that area matching his/her gas sensor, either from the high-level application logic, or other mobile application users, this MIO will not initiate the gas sensing process. Conversely, if the application needs additional data samples from MIO's vicinity, an appropriate subscription is forwarded to it such that the sensing process can be initiated. Note that the entire publish/subscribe interaction is inefficient only in situations when there are large numbers of subscriptions generated both by end users and application logic that cover the entire sensed area: In such cases it is favorable to ship all generated data into the cloud without data filtering on mobile devices.

As crowdsensing applications typically include a web-based interface to visualize and explore a certain phenomenon, we use a web service which serves as an interface to crowdsensing data stored in the database. The web service offers application-specific methods which can be used by, e.g., web applications to offer various graphical user interfaces, or other third party applications.

Use Case: Air Quality Monitoring

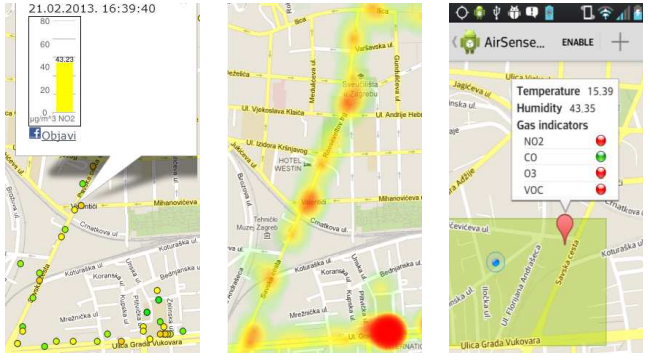
The MoPS middleware has been fully integrated into an air quality monitoring application to demonstrate the applicability of the proposed architecture in practice and its full potential for development of similar applications. The air quality monitoring application integrates the data produced by low-cost and customized mobile sensing nodes for temperature, humidity, CO, NO2 and O3 [10] which are transmitted through an Android phone running our mobile application, as shown in Figure 4.

Figure 4: Communication between our gas sensor and Nexus S phone with Android OS v4.1.2

The acquired air quality data is visualized either on a web application or on the phone, as shown in Figure 5. The web interface depicting sensor readings is shown in Figure 5(a), while the corresponding heatmap is in Figure 5(b). Figure 5(c) depicts the mobile interface which allows a user to define subscriptions (the green rectangle marks a subscribed area) and visualizes received alerts that match defined subscriptions.

Our Android client implementation has two interfaces: the sensor and cloud interface. We use a proprietary protocol for communication between a mobile device and gas sensing node which allows us to initiate and stop a periodic sensing process on the sensing node (measurements are performed once every second). This approach does not generate additional communication between the phone and sensor that would be needed in a pure pull-based sensing approach, while it provides the possibility to control and suppress the sensing process on energy-greedy sensors when measurements are not needed. Thus it enables flexible and energy-preserving sensing as well as further energy gains for the entire mobile crowdsensing system.

The cloud interface needs to support a mechanism for pushing messages from the cloud to mobile phones, and is the one which creates most problems to programmers. In addition, it can incur large energy costs since a recent study shows that periodic transfers in mobile application which account for only 1.7% of the overall traffic volume contribute to 30% of the total handset radio energy consumption [12]. Thus hereafter we briefly report three solutions that we have implemented and tested to enable delivery of notify messages in the MoPS system: 1) persistent TCP connection, 2) connection-less communication over HTTP where a REST web service is running on a mobile phone, and 3) REST web service with Google Cloud Messaging.



(a) Sensor measurements (b) Heatmap (c) Mobile application

Figure 5: Web and mobile interfaces

Persistent TCP connections are the simplest mechanism to implement, but can cause significant overhead as keep-alive messages are needed to maintain an active connection which prevents the processor from going into a sleep mode.

Connectionless REST-based communication between a mobile device and the cloud is an alternative to permanent TCP connections. Both the mobile device and server need to run a REST service: Whenever they want to communicate, they send HTTP messages to the REST service entry-point. In comparison to TCP connections, this mechanism is one step closer to push-based communication where situations of temporary connection losses and failed handover do not affect the communication mechanism. This mechanism does not allow a power save mode, but reduces the generated traffic over wireless interfaces and reduces the number of open connections. REST-based mechanism allows a mobile service to use a single entry point for all incoming messages, regardless of the sender, while the previous approach uses separate TCP connections for each sender.

For a fully implemented push-based message delivery mechanism in mobile environments we have used the Google Cloud Messaging (GCM) service. GCM is a service provided by Google running as an intermediary between application servers (cloud-based brokers in case of our prototype) and mobile devices running the Android OS. GCM uses a simple format for messages limited to 4 KB. A mobile service does not need to be in active state to receive such notifications: The Android OS will start or wake up the service upon a received message. The mechanism does not create, handle or destroy any additional connections, which makes it a true push-based communication mechanism without additional overhead. Since the support for the GCM service is an integral part of the Android operating system, GCM only requires that

a radio interface is online, and allows the processing unit to go to power save mode. The GCM mechanism is used by various Google applications on mobile devices and reuses the same connection for the delivery of all messages, thus reducing the communication overhead to a minimum. The main drawback is limited availability (only for AndroidOS) and dependency on a third party solution.

Evaluation

To gain an insight into potential energy gains due to flexible data acquisition, we have performed an analysis based on a real data set, the Mobile Data Challenge (MDC) data set collected during the Lausanne Data Collection Campaign from October 2009 until March 2011 [7]. The analysis is done such that we have randomly selected one day data traces for each of the 38 participant logs available in the MDC data set. Each such data trace represents our MIO movement over one day where we associate user locations with GSM cell identifiers. Two users are collocated if they reside with the same GSM cell during the same time interval, when they can potentially create redundant measurements. Our next assumption is that users carry wearable sensors with periodic readings generated once in a minute or once every five minutes. In addition, we assume that for our approach the required number of daily measurements within a cell equals 30 (which is comparable to the frequency of checkpoint readings, as proposed in [15]).

We have performed 10 iterations of our experiment with randomly selected daily traces from 38 different users and Table 1 depicts our results for 5 selected experiments. The second column lists the number of different cell identifiers found in all traces. It varies from 777 to 942 different cells which tells us that there is not much overlap in user movement (at most 5 to 9 users are collocated in the same cell in all our experiments). The third and the fourth column list the number of daily data measurements if sensors generate periodic readings once per minute or 5 minutes, while the fifth column lists the number of such readings with our approach. One can see that our approach generates only around 5% to 6% sensor readings and data transmissions compared to $\frac{1}{60}$ Hz measurements and 20% to 25% such readings compared to $\frac{5}{60}$ Hz measurements. Thus, based on this preliminary analysis with unfavorable movement traces with low collocation probability, one can conclude that potential energy gains are significant.

Table 1: Energy gains due to flexible data acquisition

	No. of cells	1 min	5 min	our approach
1	822	40330	9344	2013
2	870	39686	9188	2011
3	942	41153	9884	2086
4	888	42068	9977	2071
5	777	39267	9250	1807

We have also performed an initial evaluation of energy consumption on Android phones running mobile brokers. In our test scenario, a mobile broker uses either a WiFi or 3G interface for communication with the cloud. Measurements were performed on a Nexus S Android phone with PowerTutor. Table 2 shows the power and energy spent on a mobile phone to receive 1000 notifications that were sent sequentially to a mobile phone. As expected, 3G interface requires more energy for all communication modes. TCP has shown to consume the most energy, while REST is a viable solution when GCM cannot be applied, but one has to ensure that the mobile interface acquires a public IP address. In general, the GCM service shows the best results regarding energy

consumption with both 3G and WiFi interfaces because no additional network connections are needed while the processor can go to the power save mode. However, further measurements are needed to assess the messaging latency of this approach as it directly influences one of the key quality measures perceived by end users.

Table 2: Measuring the energy footprint of 1000 pushed messages

	TCP-WiFi	REST-WiFi	GCM-WiFi
Power [mW]	948	871	763
Energy [J]	1.0 K	925	797
	TCP-3G	REST-3G	GCM-3G
Power [mW]	1450	N/A	995
Energy [J]	1.6 K	N/A	1.1 K

Related Work

Publish/subscribe middleware is used mainly for either data dissemination within wireless sensor networks [2], or for data acquisition from stationary Internet-connected objects [17, 6]. There are few publish/subscribe solutions addressing MIOs as well as their interaction with the cloud. An approach most similar to ours is presented in [18] where a publish/subscribe platform for opportunistic data collection from stationary sensors is proposed. In contrast to our middleware which is general and targets environments with mobile and wearable sensors carried by users on their jackets or backpacks, Tong and Ngai design a tailored solution for a hiking trail application where mobile phones are applied as "mules" to selectively gather data from stationary sensors.

Another related body of work is continuous data stream processing on mobile devices. In comparison to publish/subscribe solutions, data stream processing supports a wider range of both stateless and statefull operators at the cost of lower efficiency. Lim et al. [8] focus on energy-efficient data acquisition from multiple sensor streams such that they choose the most favorable source with respect to continuous queries, but in contrast to our solution do not relate processing performed on one device with the cloud or other mobile devices.

Since a group of applications for mobile community sensing has been identified and labeled as mobile crowdsourcing, the authors of [1, 4] identify the need for a unified architecture and provide an outline of such architecture. There are a number of projects and real-world deployments for air quality monitoring [3, 21], however their architecture is to our knowledge not based on a publish/subscribe solution comparable to ours. Due to limited space, here we mention OpenSense which deploys environmental sensors on buses where the challenge is to define an adequate sampling frequency over bus routes for adequate sensing coverage [19]. Our approach should be used in combination with optimal sampling rates assigned to different subscriptions: For example, while Sheng et al. [16] present theoretical and simulation results to evaluate algorithms for determining energy-efficient sensing schedules of moving sensors, our approach provides the means to implement such algorithms in real systems. Finally, the work of Hasenfratz et al. [5] is comparable to ours since it uses portable and low cost sensors for gas sensing, but in contrast to our solution concentrates on sensor calibration.

Conclusion

The paper presents a generic architecture for mobile crowdsensing applications which is based on publish/subscribe middleware. We outline the underlying model and design decisions of the MoPS middleware which enables continuous data acquisition, filtering and

real-time delivery in mobile crowdsensing environments. The middleware is energy-efficient as it suppresses the transmission of sensor readings from MIOs and filters out the data which is not needed by the application logic. Furthermore, the middleware provides a mechanism to control the sensing process for energy savings on sensing nodes. To showcase the applicability of the proposed architecture, we outline the implementation of an air monitoring application. It integrates a wearable gas sensor which communicates over a Bluetooth connection with a smartphone running a mobile application that can regulate and relay sensor readings to the application back-end, as well as receive air quality alerts in real-time in accordance with user preferences and context. Our initial experiments with real user traces show great potential for energy savings of crowdsensing applications where user collocation is highly probable. Furthermore, we identify a promising technology in terms of energy footprint for pushing messages from the cloud to mobile devices.

As future work we plan to perform real-world air quality measurements to identify further optimizations of our publish/subscribe implementation and to evaluate true energy consumption and data propagation latency. Presented concepts need to be analyzed from a theoretical perspective such that optimal sampling rates and the granularity of geographic areas are identified to offer adequate sensing coverage while minimizing the overall energy consumption on MIOs.

Acknowledgements

This work has been partially carried out in the scope of the project ICT OpenIoT Project FP7-ICT-2011-7-287305 co-funded by the European Commission under FP7 program and is partly supported by the University of Zagreb Development Fund.

References

[1] Chatzimilioudis, G., Konstantinidis, A., Laoudias, C., and Zeinalipour-Yazti, D. Crowdsourcing with smartphones. *IEEE Internet Computing 16*, 5 (2012), 36–44.

[2] Costa, P., Mascolo, C., Musolesi, M., and Picco, G. P. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *IEEE J.Sel. A. Commun. 26*, 5 (June 2008), 748–760.

[3] Dutta, P., Aoki, P. M., Kumar, N., Mainwaring, A., Myers, C., Willett, W., and Woodruff, A. Common sense: participatory urban sensing using a network of handheld air quality monitors. In *SenSys '09*, ACM (New York, NY, USA, 2009), 349–350.

[4] Ganti, R. K., Ye, F., and Lei, H. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine 49*, 11 (Nov. 2011), 32–39.

[5] Hasenfratz, D., Saukh, O., Sturzenegger, S., and Thiele, L. Participatory air pollution monitoring using smartphones. In *Mobile Sensing: From Smartphones and Wearables to Big Data*, ACM (Beijing, China, Apr 2012).

[6] Jurca, O., Michel, S., Herrmann, A., and Aberer, K. Continuous query evaluation over distributed sensor networks. In *ICDE* (2010), 912–923.

[7] Laurila, J. K., Gatica-Perez, D., Aad, I., Blom, J., Bornet, O., Do, T.-M.-T., Dousse, O., Eberle, J., and Miettinen, M. The mobile data challenge: Big data for mobile computing research. In *Proc. Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf. on Pervasive Computing* (2012).

[8] Lim, L., Misra, A., and Mo, T. Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams. *Distrib. Parallel Databases 31*, 2 (June 2013), 321–351.

[9] Mühl, G., Fiege, L., and Pietzuch, P. *Distributed Event-Based Systems*. Springer, 2006.

[10] Oletic, D., and Bilas, V. Empowering smartphone users with sensor node for air quality measurement. In *IOP S&A XVII 2013 : Institute of Physics Sensors & their Applications XVII* (2013).

[11] Podnar, I., and Lovrek, I. Supporting mobility with persistent notifications in publish/subscribe systems. In *In Proc. of the 3rd Int. Workshop on Distributed EventBased Systems*, ACM Press (2004), 80–85.

[12] Qian, F., Wang, Z., Gao, Y., Huang, J., Gerber, A., Mao, Z., Sen, S., and Spatscheck, O. Periodic transfers in mobile applications: network-wide origin, impact, and optimization. In *Proc. of the 21st international conference on World Wide Web*, WWW '12, ACM (2012), 51–60.

[13] Ra, M.-R., Priyantha, B., Kansal, A., and Liu, J. Improving energy efficiency of personal sensing applications with heterogeneous multi-processors. In *UbiComp '12*, ACM (New York, NY, USA, 2012), 1–10.

[14] Sadoghi, M., and Jacobsen, H.-A. BE-tree: an index structure to efficiently match boolean expressions over high-dimensional discrete space. In *Proc. of the 2011 ACM SIGMOD*, ACM (New York, NY, USA, 2011), 637–648.

[15] Saukh, O., Hasenfratz, D., Noori, A., Ulrich, T., and Thiele, L. Route selection for mobile sensors with checkpointing constraints. In *PerCom Workshops* (2012), 266–271.

[16] Sheng, X., Tang, J., and Zhang, W. Energy-efficient collaborative sensing with mobile phones. In *INFOCOM* (2012), 1916–1924.

[17] Souto, E., Guimaraes, G., Vasconcelos, G., Vieira, M., Rosa, N., Ferraz, C., and Kelner, J. Mires: a publish/subscribe middleware for sensor networks. *Personal Ubiquitous Comput. 10*, 1 (Dec. 2005), 37–44.

[18] Tong, X., and Ngai, E. C. H. A ubiquitous publish/subscribe platform for wireless sensor networks with mobile mules. In *DCOSS* (2012), 99–108.

[19] Yan, Z., Eberle, J., and Aberer, K. Optimos: Optimal sensing for mobile sensors. In *MDM '12*, IEEE Computer Society (Washington, DC, USA, 2012), 105–114.

[20] Zeidler, A., and Fiege, L. Mobility support with rebeca. In *Proc. of the 23rd ICDCS Workshops*, ICDCSW '03, IEEE Computer Society (Washington, DC, USA, 2003), 354–.

[21] Ziftci, C., Nikzad, N., Verma, N., Zappi, P., Bales, E., Krueger, I., and Griswold, W. Citisense: mobile air quality sensing for individuals and communities. In *SPLASH '12*, ACM (New York, NY, USA, 2012), 23–24.