Vol.2013-MBL-65 No.40
Vol.2013-UBI-37 No.40
2013/3/15

A distributed system architecture for pedestrian flock detection with participatory sensing

Teemu Leppänen[1,a)] Jose Alvarez Lacasia[2,b)] Masayuki Iwai[2] Kaoru Sezaki[2]
Yoshito Tobe[3,c)] Jukka Riekki[1]

Abstract: We present distributed system architecture for smartphone-based participatory sensing applications, where the computational load is distributed between the participating devices. The system allows using of static data to reduce the computational load further. Secondly, we present a conceptual participatory sensing application for detecting pedestrian flocks moving into the same direction in certain area, using this architecture. The flock detection is based on selecting an energy efficient set of sensors in the smartphone and available location-based static data, provided by previous computations by the smartphones or by external Web services. Initially, we believe this method reduces energy consumption in the participating devices beyond the previous approaches.

1. Introduction

With the widespread introduction of smartphones into our everyday lives, new types of location-based applications and services have emerged, along with the mobile phone sensing paradigm. The smartphone users become active data contributors to achieve a common goal, subsidizing each other efforts by, for example, providing missing data to complete a high-resolution map of particular measurements.

Smartphones are suitable mobile sensing platforms, being levied on the move with the users, being programmable, integrated with sensors and with short-range communication capabilities [1], [2], [3], which also provide localization methods. Context and activity recognition from the real-time sensor-based data now becomes feasible. Because of this, crowdsensing approaches [2], including both participatory and opportunistic sensing, have become reality. These systems present unique characteristics: mobile devices are typically much more resourceful as embedded sensing platforms, allow multimodality sensing, their data types are not known a priori as in traditional sensor networks applications and millions of these devices have already been deployed, making large-scale sensing applications foreseen.

Participatory sensing with smartphones is related to interactions between humans, operating at multiple scales: individual, group and community [1], [3]. In the individual personal scale, the data collection and analysis is solely for the user personally and not for sharing. The group or social sensing adds the element of social networking or otherwise connected groups, where an element of trust exists between the participants. In community or public sensing, we have a large number of participants and there is no trust, requiring privacy protection mechanisms. The user is actively involved in the process, manually selecting the different levels of participation, where they are solely responsible for the use of their smartphones in the application [1], [2], [3]. However, this may have direct implications in the quality of the collected data, as there is no single authority to guide the process. This is known as the context problem. For example, the sensing results may differ depending on how the users carry their phones [3], [4]. The participation can also be opportunistic, where the phones passively, i.e. without explicit user involvement, contribute their resources and the decision-making is left to the application. The benefit of the opportunistic sensing is that it distributes the overall participation costs [1], [3].

In this work, we extend our previous work, where the system architecture enabling dynamic distribution of computational load was demonstrated for low-power resource-constrained wireless sensor network applications [5] and the conceptual idea of this architecture for the mobile phone sensing was presented in [6]. The distributed architecture in this paper is based on the same conceptual components, but we describe how to extend smartphones as mobile sensing and computational platforms into the system. Considering the pedestrian flock detection application, the novelty is to utilize location-based external Web services as data sources and assisting in localization indoors and outdoors, thus reducing the need for utilize physical sensors in the smartphone, eventually contributing to saving energy in the smartphone and across the whole distributed application.

The rest of the paper is organized as follows. Chapter 2 motivates this research. In chapter 3, we present the proposed system architecture and in chapter 4 we describe the method in detail.

[1] University of Oulu, FIN-90014 Oulu, Finland
[2] University of Tokyo, Komaba, Meguro, Tokyo 153-8505, Japan
[3] Aoyama Gakuin University, Chuo, Sagamihara, Kanagawa 252-5258, Japan
[a)] teemu.leppanen@ee.oulu.fi
[b)] jpalvarez@mcl.iis.u-tokyo.ac.jp
[c)] yoshito-tobe@rcl-aoyama.jp

Chapter 5 presents the conceptual example application, based on the smartphone sensing and the external data sources. Chapter 6 gives the related work and finally, in chapter 7, we discuss this approach.

2. Motivation

Our main motivation for this work is to reduce energy consumption in the participating devices across the whole application. To extend the lifetime of wireless networked devices, e.g. smartphones, it is needed to consider both communication and computational costs [5]. We believe that utilizing already available local and external data sources, considered from the application's point-of-view, can reduce the required amount of sensing, which contributing towards energy saving in the smartphones. Today, third-party Web services offer already lot of location-based pre-calculated public, i.e. static, data, accessible in the Internet. In addition, fixed or wireless sensor networks deployments already running in some locations, can provide the applications with real-time location-based additional external data.

Furthermore, we assume distributing, and this way balancing, the computational load in between mobile sensing devices and external computational platforms, such as infrastructure servers and the cloud, reduces the energy consumption even more across the whole application. We assume no centralized servers exists in the system. Uploading raw data to the cloud for data processing has been presented in previous work about participatory sensing, but it is not without drawbacks either. It consumes much energy to deliver raw or even refined data into the cloud, which has additional privacy issues and where the data processing results may not be available in real-time, depending on for example the network conditions. Therefore, some partitioning of the data processing tasks between the sensing devices, e.g. smartphones, and the backend systems, e.g. cloud, is feasible. The simplest example of this is filtering. However, here the system designers need to make sure that the smartphones are not overloaded with computational load and with participation requests, which effectively would drain the device battery and hinder the phone user experience [1].

The research challenges here include [1]: determining in runtime how much computation is done on the smartphone, how the computation is delivered into the phones, how to scale the applications to global community, how can we exploit available local data in Internet or other external data sources, how can disrupting normal phone use be avoided and finally how the users privacy is maintained.

3. System Architecture

Traditionally, mobile phone sensing system architecture consists of mobile sensing platforms to collect the data and secondly, backend server or cloud for data aggregation and processing. Also the applications utilizing the refined data are run in the backend server or in the cloud. Only a portion of the data is preprocessed, i.e. filtered, in the mobile devices to reduce the amount of data transmitted to the backend system.

We extend this work by allowing the devices to dynamically distribute the computational load, i.e. tasks as scripts, and the computed intermediate results in between themselves and the system devices [5], [6] in runtime whle the phones are in location. The smartphones execute these tasks, being initiated and delivered to them dynamically by other system devices or by requesting tasks themselves from a service requiring data input. The smartphone can also expose its intermediate computational resources into the system, as described in the next section, where it essentially becomes a service provider dynamically [5]. After the completion of a task, the final results can be exposed to the world as service content by the requestor. This way we eliminate the requirement for centralized authority to coordinate the co-operation, but we can allow in the system application-specific proxies coordinating their own task execution. This architecture allows both opportunistic and participatory sensing approaches.

This kind of distributed systems architecture requires the following components: the smartphones as mobile sensing platforms, a resource directory to host the system resource descriptions including their current location, a repository for the computational task scripts, additionally the application-specific proxies and a permanent data storage. See **Fig. 1** for illustration.

Communication between system components here is assumed to be done through an IP-based network with HTTP protocol utilizing RESTful principles. As the HTTP protocol is currently the de facto protocol in the Internet, we utilize it to provide access to the world from the system and vice versa. Moreover, HTTP has built-in privacy and security features and already supports numerous data formats providing means for content negotiation. Altought, for local short-range communication between the participating smartphones, Bluetooth or other network interfaces may be utilized.

3.1 System Resources

All the data collected by the smartphone sensors and additionally provided from the external services is considered being RESTful resources in the system. A new feature in our architecture is to consider the previous and current computations and the intermediate results also as system resources. This allows us easily extend the previousy collected data, fill in gaps in the data and address data quality issues. The external data can be both dynamic and static, where static data is not modifiable during the lifetime of the task. Following RESTful principles ensures that all the system resources can accessed uniformly in client-server manner in multi-tiered system, the resources are identifiable and locatable and the resources are accessed with stateless communication of self-describing messages.

As permanent data storage in the system, we can have a database server. Although this server is a centralized component for the historical data queries, it is not a sink node as in traditional architectures, but considered as a client to the dynamic services. It subscribes to the exposed sensor data and intermediate and final results uploaded through a proxy. Otherwise, it can be utilized indirectly through a proxy to query for the historical data.

3.2 Resource Directory

A Resource Directory (RD) is a component where resource descriptions, here as Web links, to all the system resources are

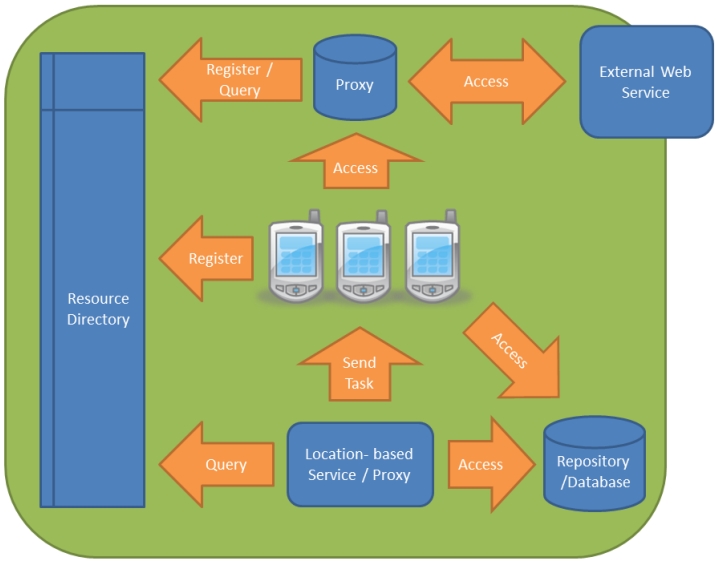

Fig. 1 The system architecture with the different components

stored. We utilize in this work the RD component developed in [5]. The RD provides interface to register, to lookup, to update and to unregister the resource descriptions. These descriptions include resource name, resource type, its address and metadata such as physical and virtual location of the resource. Using virtual locations allows semantic queries for the system resources. A unique identifier should be used for each system device, so that the resources exposed by each device can be located even if the device moves between networks. For example, the smartphone may change for more energy efficient communication network interface, available only temporarily and locally.

3.3 Proxy

The main function of the proxy here is to offer unified access and an HTTP interface to all the system resources. The proxy can expose the contents of external Web services in the system as resources, by registering them to the RD with its own address. This service content can be either static or dynamic. Additionally, the proxies can expose the system resources as services to the world through the HTTP interface. The concept of virtual sensors can be applied to the proxies as well. However, it is not necessary to use proxies for this purpose, as direct communication can be also supported. The system can have any number of proxies, for example for certain application-specific type of data, the clients can look up and locate the data and the specific proxy from the RD.

The benefit is that the proxies can offer additional application-specific functionality. Essentially, the location-based services are this kind of proxies as the computational task requestors in the system, as described in the next sections of the paper.

3.4 Mobile Sensing Platforms

In this work, we consider smartphones as the mobile sensing platforms. Currently smartphones provide multimodal sensing with at least the following types of integrated physical sensors: accelerometer, barometer, camera, compass, gyroscope, illumination, magnetometer, microphone, and proximity. Many of these sensors were originally introduced to increase the user experience of the smartphone [1]. The list of sensors can be extended with external Bluetooth-enabled sensors bridged by the smartphone. Common network interfaces in smartphones include: GSM, GPRS, 3G, Bluetooth and Wi-Fi. Additionally, RFID or NFC interfaces for short-range communications may be available. Localization can be provided with various methods by the previous sensors, through the network interfaces or commonly by integrated GPS receiver.

To be able to execute these task scripts in the smartphone, a runtime execution environment component is needed in the smartphones with a communication API. This must be run as a background service, to not to interfere with the normal use of the smartphone. The communication API needs to provide an HTTP interface to utilize the resources in the device, i.e. to actuate the physical sensor and to query data from the physical sensors. Through this interface, the sensor data types are also registered into the RD.

4. Method

The basic idea of this approach is to utilize the computational resources of the mobile sensing platforms, while the smartphones are in the given location, possibly also collecting related sensor data. We give two examples of the system functionality in the next section. The system does not provide or require predetermined workflow or graph of how to run the computational tasks, the smartphones just receive these tasks from the other system devices that are to be executed when the input data is available. This approach does not guarantee timely completion of the task as they are executed incrementally based on data availability, but it distributes the computational load dynamically. This way both opportunistic and participatory approaches are available, as smartphones are utilized when in location and the load is distributed according to the available resources.

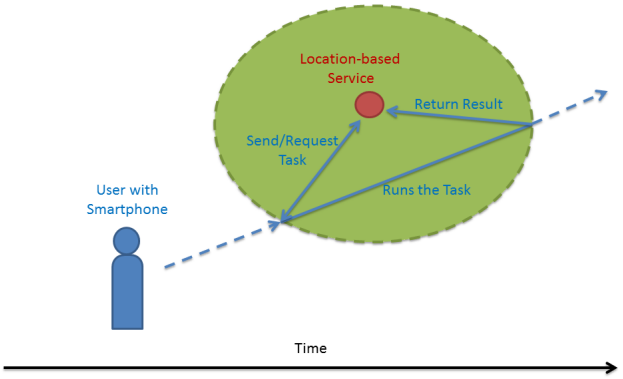

Fig. 2 Single user assisting in data collection and processing for a location-based service

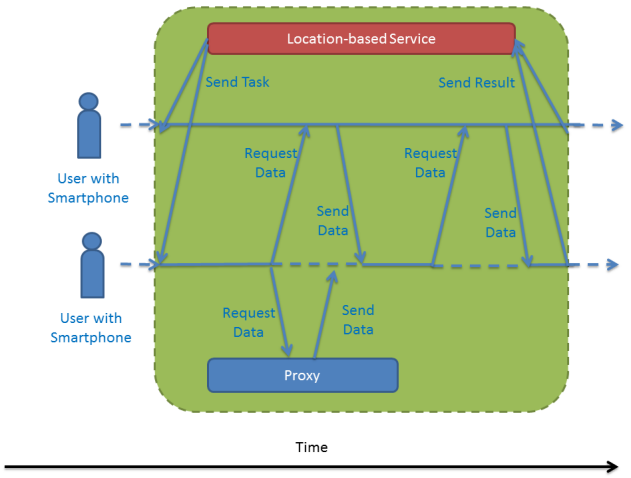

Fig. 3 Multiple users co-operating with a location-based service, where intermediate result of other and external data source are utilized in the computation

4.1 Basic Operation

Here, we illustrate two basic scenarios for utilization of location-based dynamic computational tasks in this architecture. First, we consider a single user present in the range of a location-based service, as it can be seen in **Fig. 2**. The user has previously registered his smartphone as a computational platform into the RD, with its available computational resources and it is assumed that it also updates its current location periodically. The service could be polling the RD for available computational platforms in location, and when successful, it can send the computational task into the smartphone, possible with instructions on how to continue from the previous results. While executing the computational task, the smartphone registers itself as the service producer for the given data type of the task. This way, other devices interested about the same data type can query the intermediate results from the phone. When the smartphone detects it is no longer in the desired location, it can upload the data and the intermediate results back to the service. After that, the smartphone removes the task and related data from its storage. The service can then make the final results available to the world by uploading them to the permanent storage.

In the second scenario, illustrated by **Fig. 3**, several smartphones are participating in different and separate computational tasks. The smartphones receive their tasks and after completion return final results to the service, as in the previous scenario. But, here the second smartphone uses the intermediate results of the first phone in its computations and also requests data from an external service. A coordination mechanism may be needed to synchronize the co-operation between all these system components, but for now we assume that the registration of the smartphones and the intermediate data registration during the task execution to the RD provides this. This scenario can be extended with any number of participating smartphones with the cost of perhaps increased communication and latencies in the smartphones. Short-range network interfaces should be utilized here to reduce energy consumption in the local communication and the location-based service could provide caching fo result queries, this being supported in HTTP. In the most resource-constrained case, the intermediate results are not available for other devices if the phone does not register the data type into the RD, or are not just updated in the cache in the location-based service because of low battery or even only stored into a persistent storage in the smartphone memory to be published when offline.

4.2 Requirements for Smartphones

Previous work describes a set of requirements for smartphones as research platforms [1], [2], [3], [7]. These requirements can be considered also in measuring and in implementing energy efficient participatory sensing applications. Considering communications, the network scanning capability is essential to access and switch to different network interfaces in an energy efficient manner, feature that is needed in participatory sensing applications to reduce energy consumption and to extend the length of sensing temporally. There is still the open issue of whether or not should the network scanning be user- or application-initiated. Related capability to reduce energy consumption is to dynamically change and enable / disable network interfaces separately, however this may not be always optimal in terms of access. In participatory sensing applications, short-range communication interfaces, such as Bluetooth or NFC, could be utilized in phone-to-phone communications to reduce communication costs. These capabilities also assist in localization in general, both indoors and outdoors, and in utilizing location-based services in the system, which are essential requirements in participatory sensing applications. Integrated GPS receivers provide efficient localization outdoors, at the cost of being are somewhat energy-hungry.

Energy consumption monitoring capability is also needed, because of the diversity of the smartphone platforms and physical hardware and sensors components with their individual energy consumption profiles. Persistent storage is not anymore a problem in smartphones, as integrated memory cards with gigabytes of storage space are widely available.

Importantly, the participatory sensing application must not interfere with the normal usage of the phone, thus background processing capability is needed in the operating system. This also makes possible the development for users activity recognition applications.

4.3 Energy Efficiency

We describe the energy efficiency problem with the following model, adopted and extended from [8]. The total energy consumption of this application in each phone (P_{total}) is partitioned into a set of phone subsystem energy consumptions, namely operating system (P_{os}), utilized sensors ($P_{sensors}$) and communications (P_{comm}). We propose reducing the energy consumption by replacing these subsystem components with less energy consuming components or local static data, therefore we need to add the energy consumed in fetching local static data (P_{data}) from the data sources. We formulate equation (1), where we want to minimize $P_{sensors}$ by either reducing it or by replacing components of it by adding local components to the P_{data}.

$$P_{total} = P_{sensors} + P_{comm} + P_{data} \quad (1)$$

As it is required not to interfere with the normal usage of the smartphone and it is supposed that the application is used as a background service, we leave the P_{os} in equation (1) out of this work. However, in any case to reduce P_{os}, the smartphone screen should be turned off to further save energy, unless when explicit input is required from the user [1]. Considering the energy consumption $P_{sensors}$, it is clear that continuous and even multimodal sensing consumes significant amount of energy, so we cannot assume to run the application all the time the smartphone is on and not to interfere with the phone normal use.

In this application, the energy consumption of communications is significant in between participating devices and the backend systems. Concerning the energy P_{comm}, the authors in [9] measured data transfer versus elapsed time and determined that Bluetooth consumed significantly less energy compared to 3G or Wi-Fi, where battery lasted 4 extra hours. With the same battery lifetime, Wi-Fi transfers twice as much data and Bluetooth about one third more data, compared to 3G.

Moreover, the used localization method can contribute significantly to the energy saving in the phone. To conserve energy, adaptive sampling may be utilized with the sensors, for example based on processor load or available GPS satellites. Also, it is possible to utilize a location hierarchy, starting with less accurate localization methods, i.e. consuming less energy, and switch to more accurate methods while approaching the location.

5. Pedestrian Flock Detection

A flock is defined as a group of pedestrian moving cohesively for a certain amount of time [10]. We believe that detecting pedestrian flocks is important, due to a number of reasons. Emergency related situations, such as the development of real-time evacuation guidance applications, assisting in the understanding of human activities and mobility models in densely populated urban areas and assisting in future urban developments are some of the applications we hope can be derived from this work.

5.1 Proposed Approach

Our concept application is meant for both indoor and outdoor environments, extending previous work, and uses data from sensors available in present smartphones. Specifically we can utilized data from accelerometer, magnetometer, barometer, Wi-Fi, phones microphone, GPS and proximity sensor. All the integrated sensors on the phone will only be used in the training phase of the application, in order to determine the accuracy of flock detection of each sensor, as the confidence level in the [4], and then store this information in a score list, along with the measured costs for each sensor, for example the energy consumption. The novelty of our approach resides in the usage of a extended sensor score list. Additionally, the score list includes possible energy efficient alternative data sources per sensor, which are provided into the list by the previous iterations of the same application or by the location-based services in the system. So, depending on the battery-life remaining on a phone, the application could switch to less power hungry sensors with lesser accuracy, or the opposite case. It will also be possible to define an accuracy level that the user would want to maintain, and then the application will use the least power-hungry combination of sensors found from the score list to sustain the required accuracy level of flock detection, once the application is deployed in real-life situations. It is important to mention that during the training phase this score list would not exist. The training phase will collect sensor data from all the available sensors on the phone in order to create the first version the score list.

The sensors used in the application as data sources will depend on the available sensors on the phone and the remaining battery-life. Once the sensors to be used are determined using the data from the score list, the data collection begins. Then from this raw data and possibly from the static data from other data sources, features are extracted in the same fashion that it was done in [11]. We observe that by sending features obtained from the sensor raw data, instead of sending the whole raw data itself reduces the energy used in communication [4] and we are safeguarding the users privacy to some extent. These extracted features are registered to the RD as system resources, available for use in the next step in the computation. These features give a measurement of the similarity between participating phones sensed location-based data. This way, smartphones running the application can be clustered according to the similarity of their features obtained in the previous step.

Then, depending on how many sensor modalities agree on the same clusters, these will then be considered as flocks, by majority voting. This clustering task is also a computational task in the system, and can be performed also dynamically once the data is available. It is assumed this clustering task is much more time consuming than the feature extraction [4] and requiring more data, so executing this in a parallel and distributed manner would be feasible. Once the flocks have been determined, the data related to them is uploaded to the infrastructure services along with the information regarding resource availability. This information is necessary in the future in order to determine energy consumption of these particular tasks. For example, which phone can run the clustering algorithms and will collect the features extracted from the raw sensor data of each other phone. Finally, we believe that an evaluation of the clustered flocks and the associated data, will allow the application to update the information in the score list by updating information concerning the accuracy of sensors (for example changes in the GPS sampling rate). However, the

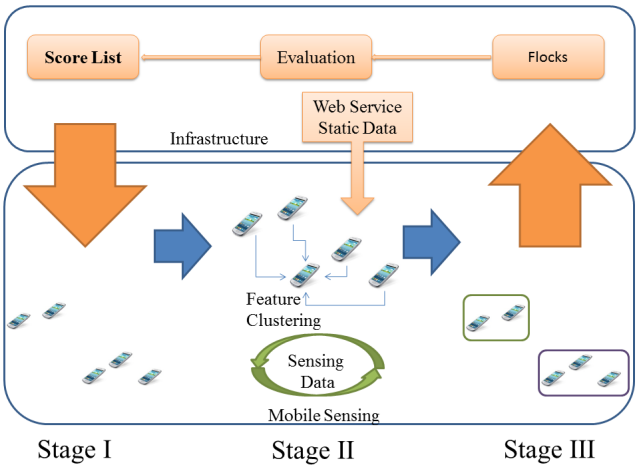

Fig. 4 The flock detection application functionality divided into 3 stages

details of the evaluation escape the scope of this paper.

See **Fig. 4** for illustration of our approach. In stage I, the smartphones consult the score list to suggest utilized sensors in the data collection. In stage II, the phones are collecting sensor data and calculating the features. In stage III, the flocks are determined according to the clustering of extracted and exposed features.

6. Related Work

The proposed distributed architecture is related and based to a diverse set of previous work. Here, we consider recent related work in frameworks enabling participatory sensing applications and in methods for distributing computational tasks in mobile sensing applications, being the focus of this paper. Additionally, we consider recent work in pedestrian flock detection.

Prism [12] addresses running precompiled binary code in a set of smartphones as tasks. The phones are centrally orchestrated to co-operatively participate to mobile sensing application. The smartphones will register their current location and available resources into the server, who determines suitable phones for the task based on given set of predicates pushing the tasks into the selected smartphones. Metering the current available resouces and forcing the phone to stop task execution make user to confirm that the tasks are draining the battery. Prism central application server to push the tasks into the smartphones, based on requests from external application server. Prism does not address co-operation between the smartphones and considers selecting the participating smartphones proactively.

The Medusa [13] is a programming framework, with its own macroprogramming language, for participatory sensing applications, based on centralized master-worker pattern. Tasks are described as stages, which are concurrently executed in the smartphones and in the cloud. The stages are stored in and reused from a repository. The intermediate results produced in stages are inputs to the following stages, where the master is only informed when the task is completed. In comparison, our approach is without any centralized components and we do not assume any specific macroprogramming language. Additionally, Medusa predetermines the task execution participating devices and execution order in the compilation stage, where we do not assume this.

In [14], the authors present a framework for tasking applications, where users can deliver simple tasks to their smartphones from a Web interface. In the framework, they provide a set hi-level abstractions to assist users in programming tasks, composing higher level abstractions from these tasks and provide a repository to publish the tasks as scripts. The tasks are precompiled in servers and heuristically partitioned to execution in either on the smartphone or in remote servers, where the tasks requiring input from other system devices are run in the servers.

In G-Sense [15], mobile sensing is integrated with fixed location wireless sensor networks, supporting location-based services. The backend system is based on servers organized in peer-to-peer manner, each providing a sensing range. Data collected by the sensors is first aggregated in a devices in the network, such as smartphones, and then transmitted into the server overlay, creating a two-tier client-server system architecture. A component in the server interconnects sensing applications, allowing distributing the sensing tasks between servers. In comparison, here the sensing platforms only consider tasks based on local and individual data, whereas the components in the servers aggregate this information.

The Micro-Blog [16] utilizes opportunistic and participatory mobile phone sensing and infrastructure Web service content to create user requested ”blog” entries, which are based on aggregating and annotating data when in location. The blog entries are uploaded to a database in centralized server and accessed thought publish / subscribe. The blog entry requests span over long time and consider both virtual and physical spaces. Each time a request is received, it is first checked whether already existing entry or set of entries respons to that particular query, before sending it to the phones in location. An energy efficient method for localization is considered in the system, based on GSM, Wi-Fi and GPS.

Darwin [4] is collaborative reasoning system is based distributing sensing tasks into the participating smartphones opportunistically. The phones collect sensor data and unsupervisedly train classifiers over time to cope with issues in fluctuating sensing environment. System allows sharing the evolved classification models for reuse, removing the need for pretrained classifiers. The smartphones then run the classifiers co-operatively in parallel to increase the selected model robustness. The Darwin system architecture is similar, however its based on the two-tier architecture, where partially the result are calculated in the phone and partially in backend servers. The goal of Darwin is unsupervised collaborative machine learning environment where the classification models are shared evolve over time. We extend this by introducing multimodality of the sensed data and we consider the available resources in the smartphone in distributing the computational tasks.

The work in [10] used only features derived from Wi-Fi signals to detect pedestrian flocks. This allowed the application to automatically determine if a group of people was in fact a flock or not. The following approach by the same authors [17] incorporates the usage of features derived not only Wi-Fi, but magnetometer and accelerometer. The values are compared and then clustered in the same manner, as described in the section 5, to define the flocks.

In comparison, both works did not consider use of external data in the flock detection. Our work also add more modalities to the data collection and to the feature extraction.

7. Discussion

In this work, we presented a distributed architecture and system components for crowdsensing applications, which supports both participatory and opportunistic approaches. Previous work demonstrated the two-tier architecture, where some data processing, such as data filtering, is run in the smartphones but the application-specific analysis is done in the backend servers. When considering the research challenges presented in the section 2, we described how currently available resource in a smartphones can dynamically utilized, without hindering the normal phone use. We presented RESTful HTTP messaging as an alternative to deliver tasks into the smartphones with Internet-connectivity. We suggested the elimination of centralized server or cloud platform running solely the computations or data analysis. Instead, we suggest distributing this computational load completely between the mobile devices in the system. However, our system does not limit the use of servers as system components in parallel, as it is only required to register those as a computational platforms. We presented how proxies are used to offer access and unified interface to the external data sources in the Internet. We did not consider security and privacy issues in this work, but the HTTP protocol used in the communication already has some security and privacy features.

We provided the system resources in RESTful manner to be used in the task scripts programming, but not discussed their use in detail, leaving the burden currently to the application or task compiler developers in the future work. We assume that the task scripts are already posted into the code repository or being the responsibility of an application-specific proxy components, where the task description is also out of the scope of this paper. However, an example task description was introduced in our previous work for the wireless sensor network applications [5]. Furthermore, task granularity is not considered here, which influences the tradeoff between communication and computation costs in resource-constrained wireless networked devices. This architecture does not guarantee task completion in a timely manner, or generate a workflow or task execution graph as instructions to run the task, but allows greater flexibility and modularity in the applications at the cost of needing to waiting for the input data to be ready. Reducing the energy consumption of communication is complex task with a number of available network interfaces and their static and dynamic characteristics.

Additionally, the architecture allows integration of external third party data sources, e.g. Web services, into the system. The external data sources are utilized to further reduce the need for energy consuming sensing in the smartphones, thus reducing the energy consumption in the participating devices. So far, nothing can be said about the quality of the data provided by the external services, but it is assumed to be static during the lifetime of the particular task where it is utilized. Implementation details of a communication and messaging remains as future work, although somewhat presented in our previous work [5].

This paper considers the distributed features of the proposed architecture, omitting a lot of technical details. The remaining issues will be addressed in the future work by implementing the first real-world applications with this architecture.

Acknowledgments Teemu Leppänen wishes to thank the Academy of Finland - Researcher Mobility for funding this work.

References

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

[2] R. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *Communications Magazine, IEEE*, vol. 49, pp. 32 –39, november 2011.

[3] W. Khan, Y. Xiang, M. Aalsalem, and Q. Arshad, "Mobile phone sensing systems: A survey," 2012.

[4] E. Miluzzo, C. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. Campbell, "Darwin phones: the evolution of sensing and inference on mobile phones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 5–20, ACM, 2010.

[5] T. Leppanen, P. Narhi, J. Ylioja, J. Riekki, Y. Tobe, and T. Ojala, "On using continuations in wireless sensor networks," in *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, pp. 1–2, IEEE, 2012.

[6] T. Leppänen, J. Riekki, and Y. Tobe, "A method for dynamic service deployment in sensor networks," p. 41, 2011.

[7] E. Oliver, "A survey of platforms for mobile networks research," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 12, no. 4, pp. 56–63, 2009.

[8] R. Murmuria, J. Medsger, A. Stavrou, and J. Voas, "Mobile application and device power usage measurements," in *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on*, pp. 147–156, IEEE, 2012.

[9] G. Kalic, I. Bojic, and M. Kusek, "Energy consumption in android phones when using wireless communication technologies," in *MIPRO, 2012 Proceedings of the 35th International Convention*, pp. 754–759, IEEE, 2012.

[10] M. Kjærgaard, M. Wirz, D. Roggen, and G. Troster, "Mobile sensing of pedestrian flocks in indoor environments using wifi signals," in *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, pp. 95–102, IEEE, 2012.

[11] H. Lu, W. Pan, N. Lane, T. Choudhury, and A. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pp. 165–178, New York, NY, USA, 2009.

[12] T. Das, P. Mohan, V. 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*, pp. 63–76, ACM, 2010.

[13] M. Ra, B. Liu, T. 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*, pp. 337–350, ACM, 2012.

[14] L. Ravindranath, A. Thiagarajan, H. Balakrishnan, and S. Madden, "Code in the air: simplifying sensing and coordination tasks on smartphones," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, p. 4, ACM, 2012.

[15] A. Perez, M. Labrador, and S. Barbeau, "G-sense: a scalable architecture for global sensing and monitoring," *Network, IEEE*, vol. 24, no. 4, pp. 57–64, 2010.

[16] S. Gaonkar, J. Li, R. Choudhury, L. Cox, and A. Schmidt, "Microblog: sharing and querying content through mobile phones and social participation," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pp. 174–186, ACM, 2008.

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

Errata

The following editorial correction has been found in the paper 'A distributed system architecture for pedestrian flock detection with participatory sensing', and should be corrected as follows.

Page 1, column 1, line 55:
We extend this work by allowing the devices to dynamically distribute the computational load, i.e. tasks as scripts, and the computed intermediate results in between themselves and the system devices [5], [6] in runtime while the phones are in location.

Page 2, column 2, line 15:
This kind of distributed systems architecture requires the following logical components: the smartphones as mobile sensing platforms, a resource directory to host the system resource descriptions including their current location, a repository for the computational task scripts, additionally the application-specific proxies and a permanent data storage.

Page 2, column 2, line 27:
Although, for local short-range communication between the participating smartphones, Bluetooth or other network interfaces may be utilized.

Page 2, column 2, line 36:
This allows us easily extend the previously collected data, fill in gaps in the data and address data quality issues.

Page 4, column 1, line 34:
Short-range network interfaces should be utilized here to reduce energy consumption in the local communication and the location-based service could provide caching for result queries, this being supported in HTTP.

Page 6, column 1, line 3:
In stage I, the smartphones consult the score list to suggest sensors to be utilized in the data collection.

Page 6, column 1, line 22:
Prism central application server is used to push the tasks into the smartphones, based on requests from external application server.

Page 6, column 2, line 28:
Each time a request is received, it is first checked whether already existing entry or set of entries respons to that particular query, before sending it to the phones in location.

Page 7, column 2, line 15:
[3] W. Khan, Y. Xiang, M. Aalsalem, and Q. Arshad, "Mobile Phone Sensing Systems: A Survey," *Communications Surveys & Tutorials, IEEE*, vol.15, no.1, pp.402-427, First Quarter 2013.

Page 7, column 2, line 21:
[5] T. Leppänen, P. Närhi, J. Ylioja, J. Riekki, Y. Tobe, and T. Ojala, "On using continuations in wireless sensor networks," in *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, pp. 1-2, IEEE, 2012.

Page 7, column 2, line 25:
[6] T. Leppänen, J. Riekki, and Y. Tobe, "A method for dynamic service deployment in sensor networks," in *International Conference on Human Probes and Smartphone Sensing*, Dec 15-17, Chiang mai, Thailand, IEICE, 2011.