Mobile Data Collection Frameworks: A Survey

Paul Y. Cao[†], Gang Li[‡], Guoxing Chen[‡] and Biao Chen[*]

[†]Department of Mathematics and Computer Science
Ashland University
Ashland, Ohio, USA, 44805
pcao@ashland.edu

[‡]Department of Computer Science and Engineering
The Ohio State University
Columbus, Ohio, USA, 43210
{lgang,chenguo}@cse.ohio-state.edu

[*]Department of Computer and Information Science
University of Macau
Macau, P.R. China
bchen@umac.mo

ABSTRACT

Mobile phones equipped with powerful sensors have become ubiquitous in recent years. Mobile sensing applications present an unprecedented opportunity to collect and analyze information from mobile devices. Much of the work in mobile sensing has been done on designing monolithic applications but inadequate attention has been paid to general mobile data collection frameworks. In this paper, we provide a survey on how to build a general purpose mobile data collection framework. We identify the basic requirements and present an architecture for such a framework. We survey existing works to summarize existing approaches to address the basic requirements. Eight major mobile data collection frameworks are compared with respect to the requirements as well as additional issues on privacy, energy and incentives.

Categories and Subject Descriptors

A.1 [**General Literature**]: Introductory and Survey

Keywords

mobile data; data collection framework

1. INTRODUCTION

In recent years, mobile devices, especially programmable smartphones, have become ubiquitous. They are usually equipped with powerful sensors such as accelerometer, gyroscope, gravity, GPS, proximity sensor, and general sensors such as microphone and camera [9, 7, 6, 20, 2, 25]. Thus researchers are presented with an unprecedented opportunity to track and analyze dynamic information sensed by smartphones. The applications of mobile sensing are wide ranging such as traffic and road monitoring, health monitoring, human behavior studies, and environment monitoring [16]. However, many research focused on the developments of monolithic programs for specific applications instead of general purpose frameworks. As one of the challenges in mobile sensing, the development of scalable and platform independent data collection frameworks has not been adequately addressed. Supported by a general purpose framework to collect data, researchers from many fields will be able to utilize mobile sensing more efficiently and effectively [19]. We provide a survey on how to build a general purpose mobile data collection framework. We identify four basic requirements and present a schematic architecture whose components provide all necessary functionalities. Technical approaches to address the basic requirements are also summarized based on a comprehensive survey. Complementary to basic requirements, we also discuss additional issues on privacy, energy and incentives. Eight major frameworks are thoroughly compared against the aforementioned requirements and additional issues.

In this paper, we claim the following contributions:

- We identify the basic requirements and additional issues for building a general mobile data collection framework. A schematic architecture for the framework is presented to address the requirements and issues.
- We summarize existing approaches to address the requirements and issues. A comprehensive comparison of eight major frameworks is also presented. To the best of our knowledge, this is the first survey focusing on mobile data collection frameworks.

The rest of the paper is organized as follows. Section 2 describes the basic requirements and architectural components for a general mobile data collection framework. Section 3 reviews existing works and discusses existing approaches to address the requirements. Section 4 discusses additional issues. Section 5 concludes the paper.

2. PRELIMINARIES

In this section, we first use an exemplary mobile data collection scenario to pinpoint the basic requirements for designing a general purpose mobile data collection framework. Then we present the architecture and describe its components in detail.

2.1 Basic Requirements

Let us consider the following scenario. Two researchers at a university want to study student behaviors. A professor specialized in data analysis, Bob, wants to investigate eating habits and study locations of computer science students during the finals week. Another social science researcher Alice wants to study social patterns of the general

Mobidata'15, June 21, 2015, Hangzhou, China.
Copyright © 2015 ACM 978-1-4503-3524-9/15/06 ...$15.00.
DOI: http://dx.doi.org/10.1145/2757384.2757396.

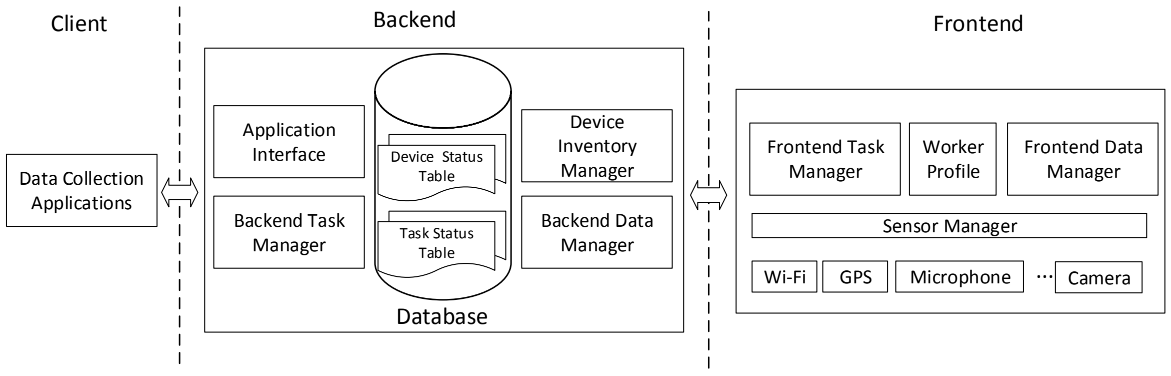

Figure 1: Schematic Architecture of General Purpose Mobile Data Collection Frameworks

student body, such as association patterns across majors, throughout an entire semester. Neither researcher is able to physically monitor their desired targets nor can they obtain accurate data through conventional surveys. Thus they decide to use the mobile devices owned by students to collect data. In this paper, we define workers as mobile phone owners who collect data (e.g. students in our scenario), and clients as researchers or companies who need the collected data to achieve a specific goal (e.g. Bob and Alice). We also use clients and (their) applications interchangeably in this paper.

To collect data efficiently, Bob and Alice may resort to a general purpose mobile data collection framework instead of collecting their own data separately. The framework must satisfy the following basic requirements.

– *Task Specification Requirement:* The framework must provide clear and precise means for clients to express their data collection needs which are called task specifications. Such specifications might include the type of data to be collected, the amount of data, and the criteria to select workers. For example, the framework should allow Bob to specify image and location as the data type and data collection should last for 7 days. For Alice, she can have 200 randomly selected students at the university and monitor the mutual proximity and locations of those students for an entire semester.

– *Task Managing Requirement:* The framework must be able to automatically distribute data collection tasks to appropriate workers based on task specifications, and keep track of the progress for each task. In the context of our example, the framework should coordinate the distribution of data collection tasks to many students' phones based on their eligibility for either Bob or Alice's research, and monitor the execution status for the two tasks.

– *Sensing Requirement:* The physical sensing activity should be carefully controlled by the framework since it is the data source. The framework should be able to initiate sensors for data collection, manage sensors, and control sensing frequencies. The framework should also guarantee data accuracy once the collection process has started. In our example, GPS, Wi-Fi, camera and bluetooth should be controlled by the framework during sensing and the location data should be accurate.

– *Data Managing Requirement:* After data are collected, they must be properly handled by the framework on three aspects. Firstly, the framework should be able to process data based on application requirements. Some applications may require data pre-processing such as noise, error, or volume reduction. Secondly, the framework should facilitate data sharing among different applications or devices for efficiency yet not at the expense of worker privacy. Thirdly, the framework should properly conduct data transmission according to clients' requirements and network availability. In our example, the framework can aggregate GPS and Wi-Fi locations before sending them to Bob and Alice. And some students' location data can be shared for both researches. Both researchers may want mobile devices to upload data when they have Wi-Fi connections.

2.2 Basic Components

Based on literature reviews, we identify basic components for a general purpose mobile data collection framework. It is shown in Fig. 1. The framework consists of two parts, the backend and the frontend. The backend operates on a local server or the cloud. It is responsible for receiving data collection requests from clients' applications, deploying collection tasks to individual mobile devices, and transmitting the collected data to applications. The frontend is usually on mobile devices. It receives collection tasks from the backend, collects sensor data, and sends the data back to the backend. Detailed functionalities of each component are described as follows.

2.2.1 Backend Components

The backend has five major components: application interface, device inventory manager, backend task manager, backend data manager and database.

– *Application Interface*: This component interacts with data collection applications and registers data collection requests. It also interprets the requests and instructs the backend task manager to deploy tasks to appropriate workers.

– *Device Inventory Manager*: It registers mobile devices which are eligible and willing to participate in a specific data collection task. It also updates devices' information in the device status table in the database.

– *Backend Task Manager*: It selects appropriate registered mobile devices from the device status table based on data collection task specifications. And it distributes collection tasks to selected devices and maintains the task status in the database.

– *Backend Data Manager*: It receives data from mobile devices, aggregates the data and sends aggregated data to different applications. This component is also responsible for possible data pre-processing before directing the data to appropriate applications.

– *Database*: The database contains at least two tables, the device status table and the task status table. The former records information for each registered mobile device, such as types of sensors, energy level, and location information. The task status table records the execution status for each distributed data collection task and it is updated periodically by the backend task manager.

2.2.2 *Frontend Components*

The frontend has four components: worker profile, frontend task manager, sensor manager, and frontend data manager.

– *Worker Profile*: It stores device owners' preferences for data collection activities. For example, a worker may decide to pause data collection if the phone's battery is under 30%. The worker profile may also be used for privacy reasons as explained in Section 4.

– *Frontend Task Manager*: It receives data collection tasks from the backend, and initializes data collection on mobile devices according to the worker profile and the sensing task. It also tracks status for different data collection tasks on the same device.

– *Sensor Manager*: It serves as an interface for the underlying sensors and the data source. It controls physical sensing activities including sampling frequencies, and provides abstraction of sensors for different sensing tasks.

– *Frontend Data Manager*: It receives collected data from the sensor manager, and transmits data to the backend. It may pre-process data locally before transmission due to energy or bandwidth concerns.

Components in the frontend and backend are designed to satisfy the requirements in Section 2.1. The application interface addresses the task specification requirement, and the two task managers as well as the database are responsible to manage, tasks at different levels. The sensor manager is in charge of sensing, and the data managers manage data for the framework. The worker profile module provides privacy and efficiency for the framework.

3. APPROACHES

In this section, we review existing works in terms of the four basic requirements given in Section 2.1. Existing approaches and strategies are discussed for most requirements.

3.1 Task Specification

Data collection needs are described in task specifications. Due to its simplicity and extensibility, XML is adopted by some works for clients to write their task specifications. There are also works that designed special languages to describe sensing tasks. Task specifications are interpreted by the application interface.

– *XML:* MEDUSA [24] provides the MedScript programming language which is an XML-based domain-specific language for clients to describe data collection needs. gPS [28] and USense [1] both use XML language to identify data needs and basic functionalities for applications.

– *Non-XML:* PRISM [8] uses an API with two-level predicates to specify where tasks are deployed and when tasks begin execution. SeeMon [14] provides a context monitoring query (CMQ) language for applications to express required context monitoring semantics. MECA [30] provides phenomenon collection specifications for clients to express their data needs. A phenomenon is defined as the occurrence of a certain kind of event at a certain location and time.

3.2 Task Management

There are two main issues for task management: task distribution and task tracking.

3.2.1 *Task Distribution*

The backend task manager is responsible for distributing tasks to mobile devices. There are two approaches for task distribution: pull approach and push approach.

– *Pull Approach:* The pull approach requires all mobile devices to independently download tasks from the backend. For example, in AnonySense [5], server distributes tasks to mobile nodes when they are ready to download new tasks. Then mobile devices decide whether to run the task(s) locally. This approach has the advantage of protecting workers' privacy since participating devices do not need to reveal their identities or other contexts such as their locations. But it brings a high overhead to the backend server due to potentially large download demands.

– *Push Approach:* This approach requires the backend server pushes tasks to a selected set of mobile devices. It requires the backend server to track registered mobile devices. The server can then select qualified devices to push data collection tasks. This approach is adopted by most frameworks for task distribution because it avoids the bottleneck in the backend server, and also deploys tasks timely. However, the privacy of the workers may be compromised since the backend server can track them.

3.2.2 *Task Tracking*

In order to track task execution status, the frontend and the backend need to cooperate on task monitoring at different levels. Therefore, task tracking includes both frontend and backend tracking.

– *Frontend Task Tracking:* Tracking in the frontend is handled by the frontend task manager in our proposed framework. It tracks tasks at a device level and monitors execution status for assigned tasks. Task execution status, such as which stage a task is at and how much data has been collected, is reported to the backend.

– *Backend Task Tracking:* For tracking in the backend, the backend task manager monitors all the tasks at a global level. For each deployed data collection task, the backend task manager maintains an entry in the task status table and monitors its progress on the assigned mobile devices. The status entry is updated when the backend receives task execution status reports from the frontend. The backend aggregates status reports from different mobile devices for each task and reports overall progress to clients. When a task expires, the backend task manager instructs frontend task managers to terminate the execution of that task on all corresponding mobile devices.

Table 1: Comparison of Frameworks on Basic Requirements

Requirements		Frameworks							
		SeeMon[14]	**DEAMON[26]**	**PRISM[8]**	**Medusa[24]**	**USense[1]**	**MECA[30]**	**MOSDEN[13]**	**gPS[28]**
Task Specification	XML				x	x			x
	Non-XML	x		x			x		
Task Distribution	Pull								
	Push			x			x		
Task Tracking	Frontend Task Tracking			x	x		x		
	Backend Task Tracking			x	x		x		
Sensor Management		x		x		x		x	x
Sensing Accuracy	Sensor Heterogeneity							x	x
	Device Placement								
	Energy Constraint	x	x	x					
Data Processing	Frontend Data Processing						x	x	x
	Backend Data Processing	x					x		x
Data Sharing	Among Tasks					x	x		
	Among Mobile Devices		x					x	
Data Transmission	Network Type		x			x			
	Sychronicity		x	x		x	x		
	Delay					x		x	

3.3 Sensing

Sensing requirement needs to address two issues: how to manage physical sensors and how to maintain sensing accuracy.

3.3.1 Sensor Management

Most existing frameworks use high-level abstractions to manage the underlying physical sensors. MOSDEN [13] introduces the concept of Plugin which is an independent application that defines how a sensor communicates with the framework. MOSDEN also provides an abstraction called virtual sensor for each underlying data source to help sensor instantiation, update and removal. gPS [28] incorporates the concept of human-as-a-sensor into the data collection process. Mobile device owners can submit their observations, opinions as well as other sensor data. USense [1] provides a sensor control layer to schedule sensor duty cycle and sampling frequency. SeeMon [14] provides a component called sensor controller which configures sensors when query changes. PRISM [8] regulates access to sensors by using three types of sensor access control policies: no sensor, location only and all sensors. The sensor manager in SeeMon [14] dynamically controls sensors to avoid unnecessary data transmissions.

3.3.2 Sensing Accuracy

The collected data may not have the expected accuracy due to sensor heterogeneity, device placement and energy constraint.

– *Sensor Heterogeneity:* Data from the same type of sensors such as microphones are not uniform because of the heterogeneity of sensors on different mobile devices. Even sensors of the same model but with different version of operating systems might yield different measurements under the same condition. There are mediator based solutions and field wrappers to normalize data; but we believe a calibration program executed prior to data collection should suffice for most applications.

– *Device Placement:* Some sensors, such as accelerometer and microphone are more susceptible to how the phone is placed during the sensing process. So appropriate adjustment during data collection might be necessary to compensate the error [17].

– *Energy Constraint:* Sometimes sensing accuracy may need to compromise with energy constraints. A typical example is the collection of location data. GSM offers coarse grained location information and is energy efficient, while GPS provides better location information but lacks power efficiency. GSM and Wi-Fi information are usually accurate enough to mine trajectory patterns. A balanced approach should be adopted to achieve sensing accuracy and energy efficiency.

3.4 Data Management

Data management involves three issues: data processing, data sharing and data transmission.

3.4.1 Data Processing

Collected raw data can be processed in the frontend, the backend or both. Most frameworks offer these three options for clients.

– *Processing in the Frontend:* Due to limited computation resources, processing in the frontend usually involves simple computations. For example, gPS[28] checks data integrity and filters out redundant data before uploading to the backend. MECA[30] tries to improve the semantic level of information and reduce the data volume by processing on device. SeeMon[14] uses preprocessors to remove noise and error from data and also performs data format conversions. MOSDEN[13] uses locally stored processing classes to process raw data, such as computing the decibel level from microphone readings.

– *Processing in the Backend:* Processing in the backend server can pursue complex computations such as data aggregation, privacy related processing, etc. SeeMon[14] performs Fast Fourier Transform to derive feature data. MECA[30] uses the edge layer to aggregate data from different devices and then sends data to the backend for further aggregation. gPS[28] performs more advanced processing and analytics on submitted data using machine learning, text mining and image processing.

3.4.2 Data Sharing

Data sharing allows different tasks or mobile devices to share data in pursuit of efficient and more accurate sensing capability.

– *Data Sharing among Tasks:* If multiple tasks request the same data such as GPS data, the framework should allow different tasks to share data sensed from the same sensors. Some frameworks enable data sharing among different tasks. For example, USense[1] reuses raw data and associated computations to cater to multiple application demands. MECA[30] enables sharing of both raw and processed data across application tasks.

– *Data Sharing among Mobile Devices:* There are also frameworks which enable data sharing among mobile devices. For example, DEAMON [26] suggests sharing sensors on neighboring sensing nodes to achieve energy-efficient sensor monitoring. MOSDEN[13] provides APIs to facilitate communications among mobile devices to encourage collaborative workload sharing and processing.

3.4.3 Data Transmission

Transmitting data from mobile devices to the server requires the selection of appropriate network(s), and a data collection framework should also handle synchronicity and delay.

– *Network Type:* Wi-Fi network, cellular network, and bluetooth have been used to transmit data. Bluetooth transmission usually serves as a relay between mobile devices. A majority of data are transmitted via Wi-Fi networks.

– *Synchronicity:* Data upload can be real-time synchronous or asynchronous. Asynchronous transmission is usually triggered by certain events, such as Wi-Fi access or at a certain time during the day [3]. Comparing with real-time upload, asynchronous upload is more cost-effective. A hybrid model of synchronous and asynchronous transmissions can be used as a middle ground solution: when data size is small or crucial, synchronous upload may occur. Or else, an event trigger might be used.

– *Delay:* Due to network availability or cost issues, data may not be able to be transmitted to the backend as timely as originally designed. Thus a general framework should support efficient data transmission concerning network availability, delay-tolerance, and overhead. Some possible solutions include data fusion which compresses packets based on their spatial and temporal properties [31] in transmission. Data repositories can also provide solutions to overhead and availability issues. One concern related to delay tolerance is local storage. Data compression might be used to conserve storage capacity [23].

We evaluate eight existing mobile data collection frameworks against the four requirements. The result is presented in Table 1. In this table, when a framework specifically addresses a certain issue, we mark a cross in the corresponding cell.

4. ADDITIONAL ISSUES

In addition to the basic requirements described in previous sections, issues related to privacy and security, energy management, and incentives warrant our attention to build a general purpose mobile data collection framework.

4.1 Privacy and Security

In mobile sensing, one big issue is how to protect privacy and ensure security. There are three existing approaches to cope with this issue.

– *Worker Profile*: In this approach, the framework allows each worker to specify a profile [10]. In the profile, workers can set policies such as what data are allowed to be collected, when and where to collect, and the conditions to trigger data capture. For example, USense [1] uses time-based, location-based and resource-based worker preferences to constrain underlying sensing. The profile allows workers to have more control in the data collection process but too many constraints might curtail functionality.

– *Worker Participation*: A framework might allow workers to reject erroneous data in an interactive manner [20]. A worker can also pause the collection process when he does not want to expose private information such as certain geolocations. This strategy helps protect worker privacy [28]. However, such worker participation might overload the server when many workers constantly pause and resume their collection processes. It is because the server has to update task status on a timely manner.

– *Sandbox*: Sandbox is a well-known security approach for executing untrusted programs. Both PRISM [8] and Medusa [24] use virtual machines (VM) and sandboxes to separate different data collection tasks on mobile devices. This approach improves system efficiency and security by allowing arbitrary binaries to run in a virtual machine. However, the overhead of VM may negatively impact mobile devices' performance.

4.2 Energy Management

Another hurdle for mobile data collection is battery life on mobile devices because sensing activities are major sources of power consumption. There are three approaches we can use to achieve energy efficiency.

– *Efficient Collection*: Data reduction, data prediction [8] and efficient data acquisition on certain sensors [21] have been proposed to reduce energy consumption. Context sharing has also been applied to achieve energy saving [22] though it raises privacy concerns. Another way to achieve efficient data collection is to balance pull and push approaches when communicating with the backend server [29]. For location data collection which is required for almost all mobile sensing applications [4], some location estimation methods can be used to save energy [15].

– *Piggyback*: This approach achieves energy saving by collecting data during the time another normal application runs in the foreground, such as normal app using, making calls, or uploading data. It is based on the fact that the energy needed for data collection is lowered when a device is already awake from the idle sleep state [18]. [18] also provides an algorithm to adequately predict when smartphone application opportunities may happen. One concern of piggybacking is whether user experience might be affected due to resource usage by the sensing activity.

– *Software-based Energy Saving*: Programmers for mobile application development must be aware of energy efficiency. Existing energy efficient design approaches on architecture, compiler, OS, and hardware can be used in mobile data collection frameworks.

4.3 Incentives

Mobile data collection consumes phone owners' resources such as battery and bandwidth, and exposes them to potential privacy and security threats. Therefore, workers would be reluctant to report their data unless they gain rewards for participating in data collection. Many mobile sensing works focused on developing end applications but failed to design incentive approaches to motivate workers. Existing research has evaluated reciprocity as means to encourage participation. Work on recommenders and P2P [11] have also suggested reciprocity. But the quid-pro-quo approach might affect phone performance. A pricing model that connects

Table 2: Comparison of Frameworks on Additional Issues

Additional Issues		Frameworks							
		SeeMon[14]	**DEAMON[26]**	**PRISM[8]**	**Medusa[24]**	**USense[1]**	**MECA[30]**	**MOSDEN[13]**	**gPS[28]**
Privacy and Security	Worker Profile	x			x	x			x
	Worker Participation				x				x
	Sandbox			x	x				
Energy Management	Efficient Collection	x	x	x		x		x	x
	Piggyback								
	Software-based								
Incentives		x			x				x

incentives to data quality can also be defined and explored [30, 18]. If ground truth can be verified, human observation errors in mobile sensing can also be linked to incentives [27]. We want to direct interested readers to a recent survey paper on incentives [12].

Similar to the previous section, we evaluate the same eight existing mobile data collection frameworks against the issues discussed in this section. The result is shown in Table 2.

5. CONCLUSION

In this paper, we present a survey on how to build a general mobile data collection framework. We identify the basic requirements and present an architecture for such a general framework. We also summarize existing approaches to address the basic requirements. We survey eight state of the art mobile data collection frameworks and compare them against basic requirements and additional issues. As can be seen in Table 1 and Table 2, no single framework covers all requirements and issues that need to be addressed. There are many open issues and challenges in mobile data collection framework research such as how to scale up mobile sensing projects and how to allow multiple sensing applications to run concurrently under a single app on workers' phones.

6. REFERENCES

[1] V. Agarwal, N. Banerjee, D. Chakraborty, and S. Mittal. USense - A smartphone middleware for community sensing. *MDM*, 1:56–65, 2013.

[2] X. Bao and R. Roy Choudhury. Movi: mobile phone based video highlights via collaborative sensing. pages 357–370, 2010.

[3] G. Chittaranjan, J. Blom, and D. Gatica-Perez. Mining large-scale smartphone data for personality studies. *Personal and Ubiquitous Computing*, 17:433–450, 2011.

[4] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, 84(11):1928–1946, 2011.

[5] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *MobiSys*, pages 211–224. ACM, 2008.

[6] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan. Perfalld: A pervasive fall detection system using mobile phones. pages 292–297, 2010.

[7] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan. Mobile phone based drunk driving detection. pages 1–8, 2010.

[8] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. PRISM: Platform for Remote Sensing using Smartphones. *MobiSys*, page 63, 2010.

[9] T. M. T. Do and D. Gatica-Perez. Groupus: Smartphone proximity data and human interaction type mining. In *ISWC*, pages 21–28, 2011.

[10] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. a. Peterson, G.-S. Ahn, and a. T. Campbell. The BikeNet mobile sensing system for cyclist experience mapping. *SenSys*, page 87, 2007.

[11] M. Feldman and J. Chuang. Overcoming Free-Riding Behavior in Peer-to-Peer Systems. *SigECom*, 5(4):41–50, 2005.

[12] H. Gao, C. Liu, W. Wang, J. Zhao, Z. Song, X. Su, J. Crowcroft, and K. Leung. A survey of incentive mechanisms for participatory sensing. *Communications Surveys Tutorials, IEEE*, PP(99):1–1, 2015.

[13] P. Jayaraman. Efficient opportunistic sensing using mobile collaborative platform mosden. *CollaborateCom*, 2013.

[14] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song. Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. *MobiSys*, pages 267–280, 2008.

[15] A. Khan, S. Ka, A. Imon, and K. Das. Ensuring Energy Efficient Coverage for Participatory Sensing in Urban Streets. *SMARTCOMP*, 2014.

[16] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad. Mobile phone sensing systems: A survey. *Communications Surveys & Tutorials, IEEE*, 15(1):402–427, 2013.

[17] K. Kunze and P. Lukowicz. Dealing with sensor displacement in motion-based onbody activity recognition systems. *UbiComp*, page 20, 2008.

[18] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha. Piggyback CrowdSensing (PCS): Energy Efficient Crowdsourcing of Mobile Sensor Data by Exploiting Smartphone App Opportunities. *SenSys*, pages 7:1—-7:14, 2013.

[19] D. Lazer, A. Pentland, L. Adamic, S. Aral, A.-l. Barabási, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, G. King, M. Macy, D. Roy, and M. V. Alstyne. Computational Social Science. *Science*, 323(February):721–723, 2009.

[20] H. Lu, W. Pan, N. Lane, T. Choudhury, and A. Campbell. SoundSense: scalable sound sensing for people-centric applications on mobile phones. *MobiSys*, pages 165–178, 2009.

[21] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The Jigsaw continuous sensing engine for mobile phone applications. *SenSys*, page 71, 2010.

[22] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. Darwin Phones : the Evolution of Sensing and Inference on Mobile Phones. *MobiSys*, 14:5–20, 2010.

[23] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing Meets Mobile Social Networks : The Design , Implementation and Evaluation of the CenceMe Application. *SenSys*, pages 337–350, 2008.

[24] M.-r. Ra, B. Liu, T. L. Porta, and R. Govindan. Medusa: A Programming Framework for Crowd-Sensing Applications Categories and Subject Descriptors. *MobiSys*, (Section 2), 2012.

[25] P. Rubel, J. Fayn, G. Nollo, D. Assanelli, B. Li, L. Restier, S. Adami, S. Arod, H. Atoui, M. Ohlsson, et al. Toward personal ehealth in cardiology. results from the epi-medics telemedicine project. *Journal of Electrocardiology*, 38(4):100–106, 2005.

[26] M. Shin, P. Tsang, D. Kotz, and C. Cornelius. DEAMON: Energy-efficient sensor monitoring. *SECON*, 2009.

[27] D. Wang, M. T. Amin, S. Li, T. Abdelzaher, L. Kaplan, S. Gu, C. Pan, H. Liu, C. C. Aggarwal, R. Ganti, X. Wang, P. Mohapatra, B. Szymanski, and H. Le. Using humans as sensors: An estimation-theoretic perspective. *IPSN*, pages 35–46, 2014.

[28] F. j. Wu. A generic participatory sensing framework for multi-modal datasets. *ISSNIP*, pages 21–24, 2014.

[29] Y. Xu, S. Helal, M. T. Thai, and M. Schmalz. Optimizing Push / Pull Envelopes for Energy-Efficient Cloud-Sensor Systems. *MSWiM*, page 10, 2011.

[30] F. Ye, R. Ganti, and R. Dimaghani. MECA: mobile edge capture and analysis middleware for social sensing applications. *WWW*, pages 699–702, 2012.

[31] D. Zhao, H. Ma, and S. Tang. COUPON : A Cooperative Framework for Building Sensing Maps in Mobile Opportunistic Networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(2):392–402, 2015.