# State-machine driven Collaborative Mobile Sensing Serving Multiple Internet-of-Things Applications

Radhika Loomba*, Lei Shi†, Brendan Jennings†
*Intel Labs Europe, Ireland
†TSSG, Waterford Institute of Technology, Ireland

***Abstract*—The myriad of sensor information that can be collected using smartphones, wearables and other IoT devices greatly benefits context-aware applications. These applications rely heavily on mobile devices, present in locations of interest, to offload raw or processed sensor data in order to accurately capture, recognize and classify the surrounding real-time context. However, continuous sensing and offloading of large volumes of mainly redundant sensor data significantly impacts energy-constrained mobile devices. This results in a trade-off between sensing accuracy and the energy consumed by these devices. We propose the use of application-specific state machines that encode the context of interest to determine when sensed data should be offloaded to the cloud. Our control algorithm, 'Assisted-Aggregation' applies frequent pattern mining to reduce the number of active devices by sharing sensed data between multiple applications. Our evaluation shows an improvement in terms of the residual energy of the mobile devices, the number of devices actively offloading and the volume of the offloaded data.**

## I. Introduction

Nowadays embedded sensors in mobile devices have a rich set of sensing capabilities, and have become increasingly more sophisticated. These sensors include specialized environmental sensors (ambient light, barometers, photometers, and thermometers), motion sensors (accelerometers, gravity sensors and gyroscopes), positional sensors (compasses and magnetometers) as well as general purpose sensors (microphones, proximity sensors and cameras). This has led to a proliferation of applications that rely on mobile devices to collect sensed data from their embedded sensors and/or from the heterogeneous Internet-of-Things (IoT) devices present in the physical environment, providing a personalized and context-aware experience to the users.

Minimizing energy consumed during sensing and reporting of such sensed data continues to be an important challenge. However, the most commonly used continuous sensing mechanism has been reported to reduce the standby time of mobile devices from 20 hours to 6 hours [25]. To overcome the inefficiency of this energy consumption, recent studies of mobile sensing systems have proposed a combination of hardware and software based energy-saving methods including energy-accuracy trade-offs [19], low-power processors [29] and sensing pipelines [24]. An important factor of energy efficiency is the volume of offloaded data [14], [35]. But few research studies assess the trade-off between the volume of data offloaded into the cloud and the energy consumed during sensor data collection, aggregation and dissemination. Harnessing the ability of a mobile device to act as a travelling corresponding agent to interact with the IoT sensors, and with the added benefits of mobile cloud computing techniques, we present a collaborative mobile sensing approach with a centralized middleware architecture. By identifying the event-driven nature of most mobile applications, we propose an algorithm that dynamically alters the reporting rates of mobile devices to save energy. In contrast to continuous offloading, our approach encodes application requirements into application-specific state machines with reporting thresholds for different sensors.

Our main contribution is the 'Assisted-Aggregation' algorithm, which creates novel consolidated state machines for each sensor in an identified physical area of internet, embodying the thresholds specified by multiple applications. For reducing redundancy due to overlapping streams of sensor data from similar environments, it applies frequent pattern mining to identify the best combination of embedded sensors in mobile devices and available IoT sensors (within transmission range of the mobile devices) to simultaneously satisfy the requests of multiple applications, whilst reducing the volume of offloaded data and the number of devices that actively offload information. Additionally, depending on the size of the consolidated state machine, the algorithm factors it into independent smaller state machines in order to improve scalability. This work extends our previous work [22] where we presented the 'Info-Aggregation' algorithm, which exploited the embedded-sensors in a mobile device to satisfy requirements of multiple applications but assumed continuous sensing and reporting of sensed data. Other previous work [20] explored the use of state-machines to reduce energy consumption, but did so on an application-by-application basis, without employing state-machine consolidation as done here. Moreover, our previous work assumed all sensors were embedded in the mobile device itself, whilst here we consider the augmentation of the sensing capabilities of the mobile device with the heterogeneous IoT sensors, as it moves through the environment. Thus, the set of sensor types available to a mobile device typically varies over time.

This paper is structured as follows. §II describes the related work in the area of energy-efficient mobile sensing techniques and collaborative sensing approaches. §III specifies our problem scenario and provides a mathematical formulation. §IV discusses the algorithm design while §V details the experimental setup and presents our results. Finally, §VI summarizes the paper.

978-3-901882-89-0 @2017 IFIP

## II. Related Work

The increase of sophisticated embedded sensors has provided an attractive platform for the development of two mobile sensing paradigms, namely participatory (involving active user participation) and opportunistic sensing (involving automatic collection of sensor data) [4], [5]. Several mobile applications such as StressSense [23], CenceMe [25], NoiseTube and MobiShop [13] use these approaches to provide situational awareness feedback to users. The integration of sensors surrounding the user, other than those embedded in mobile devices, has been presented in the OPPORTUNITY framework [15]. However, as mobile applications become increasingly dependent on sensed information, it becomes important to reduce the energy consumption during this process. Ranging from system-level designs during continuous sensing to include low-power processors [18], [29], bidirectional feedback pipelines [12], context correlation with association rules [27] and sensing pipelines [24], multiple techniques have been presented by researchers for this purpose. Furthermore, energy-accuracy trade-offs using adaptive sampling intervals and dynamic sampling frequencies [3], [19], [32], [36] have also been investigated. In contrast, we consider energy-saving by adopting a state-machine driven approach to leverage application-specific knowledge that dynamically alters the reporting rate for the mobile device and reduces number of cloud transmissions. Additionally, we consolidate application-specific state machines so that sensed information can be simultaneously provided to multiple applications as required, reducing the volume of sensed data that is offloaded.

Other collaborative sensing techniques include iCoMe [37], an incentive-based cooperative resource management approach to increase service provider revenue, a broker-based mobile cloud to present a double-sided bidding mechanism for resource sharing [40], Transient Clouds [28] a cloud-on-the-fly approach for collaboration and AnonySense [6], [39] considering privacy and threats in such systems. Additional methods include taking advantage of computational resources of nearby mobile devices [26], [38] and multiple access links for collaborative downloading and gateways [1]. In contrast, our collaborative middleware focuses on satisfying multiple application requirements by aggregating sensor data streams and offloading the information into the cloud for processing by using a small number of mobile devices.

## III. Problem Definition

### A. Scenario

We present a smart-city scenario, similar to [17], [33], with connected infrastructure components and services like education, healthcare, transportation etc. along with secure smart housing. The city is modelled as a collection of interest areas that need to be actively monitored to provide a range of user-oriented services. This task is fulfilled by several context-aware applications by either tracking environmental factors like temperature, humidity, pressure and the presence of unwanted gases, or reacting to rising noise levels for traffic management or promoting social interactions [13].

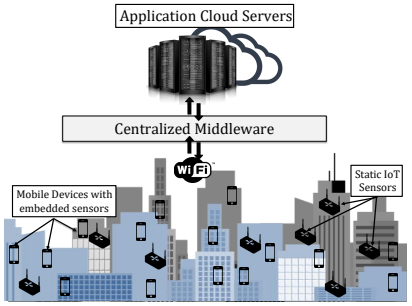
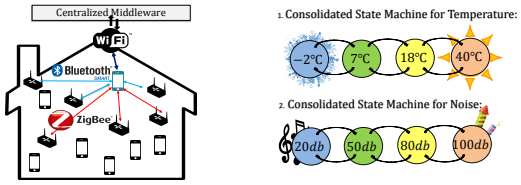



Fig. 1. Representation of our smart city scenario showing the interaction of the application cloud with static IoT sensors and mobile devices using WiFi to connect to a trustworthy collaborative sensing coordination middleware.




(a) (b)

Fig. 2. (a) shows how one request location is covered by an active mobile device using its own embedded sensors and available static IoT sensors in the request location whilst offloading the collected sensor-data to the cloud for processing using WiFi, when required; (b) shows two consolidated state machines for sensor-types temperature and noise present for a request location.

In this setting, as presented in Fig. 1, we identify the benefit of installing a trusted, centralized intelligent middleware that supports collaborative sensing, ensures content-integrity and privacy, and provides a platform for applications with improved real-time analytics. The middleware is responsible for receiving requests from the applications, interacting with the applications clients and simultaneously connecting to the hundreds of embedded sensors of the mobile devices and accessible static sensors present within the interest areas. Leveraging the ability of mobile devices to act as travelling agents, the middleware creates an aggregation and transmission schedule that instructs the devices to collect sensor data either from its own sensors or from the nearby static sensors using Bluetooth, ZigBee, or similar technologies. Depending on the reporting constraints for the applications available within the middleware, sensor data is transmitted as anonymous data streams to the application cloud for further processing and storage. Such an architecture supports both participatory and opportunistic sensing [16] as mobile devices are able to tag which sensors can be accessed by the middleware and adapt to the availability of the static sensors surrounding the device.

### B. Mathematical Formulation

The problem is formulated for a physical sensing area $\mathcal{L}$, composed of a set of square interest areas or request locations for which sensed data is requested by the applications over a duration of $\mathcal{T}$ seconds. An individual request location $\mathcal{L}_i \in \mathcal{L}$

is a set of all 2D coordinates that define the $i$th location in the sensing area. After every $\Delta t$ seconds, pre-processing of the collected sensor data is done to ensure that all requirements of the applications have been met. For $t \in \mathcal{T}$ and $k \in [1, \mathcal{T}/\Delta t]$, these time intervals are denoted by $(k-1)\Delta t < t < k\Delta t$. Two kinds of sensors have been modelled in this study that cover a range of different sensor-types denoted by set $\mathcal{H}$, namely embedded sensors within the mobile device and static sensors that have been installed in the request locations. We assume that the sensed information present for any sensor does not vary much during a time interval and does not affect how we address the application requirements.

For each mobile device $n \in \mathcal{N}$, we define $\mathcal{M}_n \subseteq \mathcal{H}$ as the set of unique sensor-types embedded within the device. The individual sensor-type $m \in \mathcal{M}_n$ can sense data with accuracy $\rho^n_m$ and accumulate $v^n_m$ volume of data (if activated) in every time interval. The request location covered by the mobile device is determined by the 2D position coordinates of the device $(x_n(k), y_n(k))$, defined as a function of the time interval $k$. Additionally, we also define the set $\mathcal{N}^k_i$ such that $\forall n \in \mathcal{N}^k_i, (x_n(k), y_n(k)) \in \mathcal{L}_i \in \mathcal{L}$ for time interval $k$.

Each static sensor, $s \in \mathcal{S}_i$ for request location $\mathcal{L}_i \in \mathcal{L}$ contains $\mathcal{M}_s \subseteq \mathcal{H}$ unique sensor-types. Each individual static sensor-type $m \in \mathcal{M}_s$ can sense with accuracy $\rho^s_m$ and accumulate $v^s_m$ volume of data (if activated) for every time interval. These static sensors do not directly connect to the middleware and only interact with the mobile devices present in the request location for security reasons. This implies that only those request locations which have at least one mobile device present for the entire duration of the $k$th time interval can be sensed for any application. For simplicity of presentation, we assume that each request location meets this constraint and we will cover more complex scenarios in our future work. Fig. 2a depicts how one mobile device covers the request location (for e.g. a house) by using its own embedded sensors and the surrounding static sensors via Bluetooth or ZigBee communication technologies whilst offloading the collected sensor-data to the cloud for processing using WiFi, as needed.

The sensor-types requested by application $a \in \mathcal{A}$ are defined by the set $\mathcal{M}_a \subseteq \mathcal{H}$ for which sensing is required for every request location in the set $\mathcal{L}_a \subset \mathcal{L}$. Applications also specify minimum accuracy requirements $\rho^{*a}_M$ and a constant sensing time-interval which can be translated into a set of time-points $\tau^a_M$ for each sensor-type $M \in \mathcal{M}_a$. The time interval $k$ then contains all the sensing time-points for all applications: thus $\bigcup_{a \in \mathcal{A}, M \in \mathcal{M}_a} (k-1)\Delta t < t \in \tau^a_M < k\Delta t$ indicates the applications and the sensor-types for which sensed data is requested in the time interval $k$. However, we determine that applications do not require updated sensor-information at every time-point in $\tau^a_M$. This is due to our understanding that most applications make decisions based on the information derived regarding the context of a particular location, which is composed of individual raw sensor readings and as such, continuous offloading of sensor data does not serve a useful purpose. Thus, we model application-specific state machines, where the context is encoded in each state along with a reporting threshold required for that sensor-type. This is beneficial as it supports seamless identification of the information required by the application. This instructs the middleware to collect and pre-process the data according to time-points in set $\tau^a_M$ and to transmit updated sensor data to the cloud only when it crosses a reporting threshold. For example, an application requests for temperature readings to be sensed at a time-interval of 10 minutes, but reported only when the temperature value crosses 7 °C or 18 °C. Since multiple applications request for sensed data for the same sensor-type, individually maintaining and accessing each state-machine in the energy-constrained mobile device can be infeasible. Thus, the middleware creates a consolidated state machine from all application requirements requesting for sensor information from a particular location. We denote the state machine for sensor-type $h \in \mathcal{H}$ from location $\mathcal{L}_i$ by a set of states $\mathcal{Z}^i_h$ where each state is composed of the threshold value for that state, the state-transition rules and the transmission rules for offloading data into the cloud. Currently, we assume that these state machines are simplistic in nature and only allow neighbouring state transitions. Fig. 2b depicts two consolidated state-machines for sensor-type temperature and noise maintained for a request location by the middleware. At the end of each time interval $k$, the reporting mobile device pre-processes the data and checks with the state machine to decide whether it should offload the data or not.

Given the assumptions outlined above, this optimization problem focuses on creating a trade-off between a) the selection of mobile devices offloading sensed information into the cloud, b) the global energy consumption of the mobile devices and c) the volume of data being offloaded into the cloud.

*1) Decision Variables:* Since the number of sensors available for getting one particular sensor-type data are more than one, we define two decision variables to identify which sensor is being used for accessing the data for one sensor-type $m$ within a time-interval $k$. These are denoted by $p^k_{mn} \in \{0, 1\}$ for every mobile device $n$ and $q^k_{ms} \in \{0, 1\}$ for every static sensor-type $s$. Additionally, for fairness and scalability, we model the system to ensure that a single mobile device is not always reporting and storing the entire consolidated state machine for each request location. As such, the middleware needs to decide a set of candidate reporting devices and factor the state machine, so that each candidate reporting device has a part of the state machine. For this purpose, a mobile device is considered active if it is responsible for reporting the sensed data to the application for one or more sensor-types during time interval $k$. This is indicated by the decision variable $y^k_n \in \{0, 1\}$ which is equal to one if the device is a candidate reporting device.

*2) Constraints:* We define constraints relating to the sensor coverage area, accuracy and minimum device battery level as follows.

**Coverage Constraint:** As stated above, each application $a$ has specified sensor-types $\mathcal{M}_a$ for request locations $\mathcal{L}_a$ that need be covered for time interval $k$. This constraint ensures that these areas have the specified sensors and at least one

mobile device that can offload sensed data if required:

$$\begin{array}{l}\forall k \in [1, \mathcal{T}/\Delta t], \forall a \in \mathcal{A}, \forall \mathcal{L}_i \in \mathcal{L}_a, \forall M \in \mathcal{M}_a \cap (\mathcal{M}_s \cup \mathcal{M}_n), \\ \forall n \in \mathcal{N}_i^k, \forall s \in \mathcal{S}_i : \\ \mathcal{N}_i^k \neq \emptyset \\ \sum_{\mathcal{N}_i^k} y_n^k \geq 1 \end{array} \tag{1}$$

**Accuracy Constraint:** The accuracy constraint for each application must be met by either the embedded sensors on the mobile device or the available static sensors that can be accessed by the mobile device for each request location. Thus, we have:

$$\begin{array}{l}\forall k \in [1, \mathcal{T}/\Delta t], \forall a \in \mathcal{A}, \forall \mathcal{L}_i \in \mathcal{L}_a, \\ \forall n \in \mathcal{N}_i^k : y_n^k = 1, \forall s \in \mathcal{S}_i, \forall M \in \mathcal{M}_a \cap (\mathcal{M}_s \cup \mathcal{M}_n) : \\ {\rho^*}_M^a \leq max(\{\rho_M^n\} \cup \{\rho_M^s\}) \\ p_{mn}^k = 1 \implies \exists k : {\rho^*}_M^a = \rho_M^n \\ q_{ms}^k = 1 \implies \exists k : {\rho^*}_M^a = \rho_M^s \end{array} \tag{2}$$

**Battery Life Constraint:** Energy costs are incurred by a mobile device $n$ related to its monitoring and sensing activities. We denote $E_n^k$ as the energy consumed to collect sensor data from all the embedded-sensors and from the surrounding static sensors during time interval $k$. Additionally, the device loses further energy if it is selected as a candidate reporting device for request location $\mathcal{L}_i$ and needs to pre-process sensed data whose volume $V_n^k = \sum_{m \in \mathcal{M}_n} v_m^n \cdot p_{mn}^k + \sum_{s \in \mathcal{S}_i} \sum_{m \in \mathcal{M}_s} v_m^s \cdot q_{ms}^k$. First, it consumes energy to store and access a factored state machine or part of the consolidated state machine for each of the sensor-types, which is represented by $F_n^k = \phi_1 \cdot \left(\sum_{m \in \mathcal{M}_n} p_{mn}^k + \sum_{s \in \mathcal{S}_i} \sum_{m \in \mathcal{M}_s} q_{ms}^k \cdot y_n^k\right) \cdot \frac{|\mathcal{Z}_m^i|}{\sum_{\forall n \in \mathcal{N}_i^k} y_n^k}$. Next, it consumes energy to locally pre-process the data, identify the state which might involve communication with surrounding devices and offload the collected sensor data which is represented by $G_n^k = \beta_2 \cdot V_n^k$. The battery constraint below thus states that the battery level of the mobile device $n$ at the start of time interval $k$, denoted $b_n^k$, is below $\theta$ of the full battery level $B_n$. This restriction is imposed on all mobile nodes to ensure battery availability for monitoring, sensing, local data pre-processing and information offload into the cloud.

$$\begin{array}{l}\forall n \in \mathcal{N}, \forall k \in [0, \mathcal{T}/\Delta t] : \\ b_n^k >= \theta B_n \\ b_n^k - E_n^k - F_n^k - G_n^k = b_n^{k+1} \end{array} \tag{3}$$

### C. Objective Function

During each time interval $k$, a deployment has a set of possible mobile devices with embedded sensors and static-sensors installed within a location that are activated to provide information at different accuracies for applications. Our objective function uses summation summation to balance three terms within each time interval. These are:

- Energy consumption due to sensing, local data processing and information offload into the mobile cloud;
- Number of active mobile devices in the sensing environment—A weight and normalizing factor denoted by $\gamma$ is attached to this term;
- Volume of data offloaded into the cloud—A weight and normalizing factor denoted by $\delta$ is attached to this term.

These can be expressed formally as:

$$\text{minimize} \sum_{k=1}^{\mathcal{T}/\Delta t} \left( \sum_{n \in \mathcal{N}} (E_n^k + F_n^k + G_n^k) + \gamma \sum_{n \in \mathcal{N}} y_n^k + \delta \sum_{n \in \mathcal{N} : y_n^k = 1} V_n^k \right) \tag{4}$$

## IV. Algorithm Specification

The 'Assisted-Aggregation' algorithm, detailed in Algorithm 1, focuses on aggregating the sensed data, by multi-tasking the capabilities of one mobile device, whilst ensuring that the consolidated state machine of the various sensor-types is factored and distributed between a set of reporting devices. This leads to a reduction in redundancy of offloaded streams of data and supports dynamic altering of the reporting rate of mobile devices for better energy management.

We now describe some terminology which helps us explain the flow of our algorithm. As stated before in §III, every application requires sensed information from specific sensor-types for request locations $\mathcal{L}_a$ at a minimum accuracy level of ${\rho^*}_M^a$, at sensing time-points $\tau_M^a$ with reporting thresholds based on an application-specific state machine. We use the term *Application-Sensor pair* $< a, M >$ to uniquely identify this relationship. Additionally, multiple mobile devices can collect sensor-data either from the embedded sensors or from the surrounding static sensors for one sensor-type for a particular location. This helps us create multiple combinations of mobile devices which, combined together, cover all the coverage constraints as specified by the Application-Sensor pair. We term this as a *NodeSet*, which refers to one such set of mobile devices that can be used for the Application-Sensor pair. We define the energy consumed for sensing by the NodeSet as a summation of the individual energy consumed by the mobile devices in the set and ensure that each mobile device satisfies the accuracy constraint.

The algorithm then works in the following way. First, for every time interval $k$, we predict the mobility pattern of the mobile devices in a look-ahead manner. We assume that the starting position can be retrieved by the middleware using GPS, WiFi Positioning or some similar technology. Using the function *predictLocOfMobileDevice(n,t.start, t.end)* in line 5 and an instance of the mobility model, we determine the location which is covered by the mobile device $n$ for the specified time interval. Next, by using the set $\tau_M^a$, those Application-Sensor pairs are determined in line 7 for which sensing needs to be done within that time-interval. When calculating NodeSets to satisfy the pair in line 10, a NodeSet is only considered if each mobile device in the set covers its current location for entire time interval $k$. We make an assumption that the variance in the movement of the mobile device within the interval does not affect the sensing readings if it is still in a position to cover the location. Additionally, the ability of a mobile device to communicate with all static sensors in its surrounding area is exploited. This determines which sensor (embedded or static) should be used for every

sensor-type according to the requested accuracy $\rho^{*a}_{M}$. We make a trade-off between the cost involved in communicating with the static sensors and accuracy of the sensor data provided to the Application-Sensor. As such, mobile devices will access the static sensors only if the embedded sensors in the mobile device are unable to satisfy the accuracy constraints. The selection of one NodeSet for every pair is made only after calculation the frequency of every subset of mobile devices amongst all NodeSets present in the time interval. This is done to ensure that subsets containing devices which can sense and report for more than one application are selected. For this, the FP-Growth algorithm [9] is deployed, with output available in the variable $P$ in line 13. This widely studied pattern mining technique [10], [11] is chosen as it enables unsupervised learning and allows patterns to be found for all kinds of data and large-datasets. The pseudo-code and algorithmic descriptions for these functions are present in [22].

Next, a *base subset* termed as $fSet$ is created by concatenating all the subsets of mobile devices whose size is equal to the largest or second largest most frequently-occurring set with available battery (in contrast to our previous approach [22] of selecting one such subset) in line 17. Iteratively, one NodeSet is selected for each Application-Sensor pair that utilizes the maximum number of mobile devices in the base subset as defined in Function *calcAppSensor* in line 24. Each mobile device in this set is then designated as a candidate reporting device for the sensor-types that are either embedded in it or whose data can be collected from the surrounding static sensors. In line 28 and 29, the candidate reporting device receives a factored state machine with a range of reporting threshold stored in array $reportThreshold$[] along with a list of the neighbouring reporting candidates stored in array $list$[]. Currently, a simple parallel factoring technique is used by the middleware to split the state machines and more complicated techniques like those defined by Devadas *et al.* [7] will be explored in our future work. After minimal pre-processing of sensor data for its location in line 30, the device checks whether the raw value lies within the reporting thresholds of its factored state machine. In case the value lies outside the range of the reporting thresholds, the device starts a one-2-many connection with its neighbouring reporting devices in line 34 to determine whether the data needs to be offloaded. The sensor data is offloaded using the function *offloadAccToState* accordingly. Finally, the energy consumed and the volume offloaded by each mobile device for the time interval $k$ is calculated using functions *calcEnergyConsumption (k)* and *calcVolumeOffloaded (k)*.

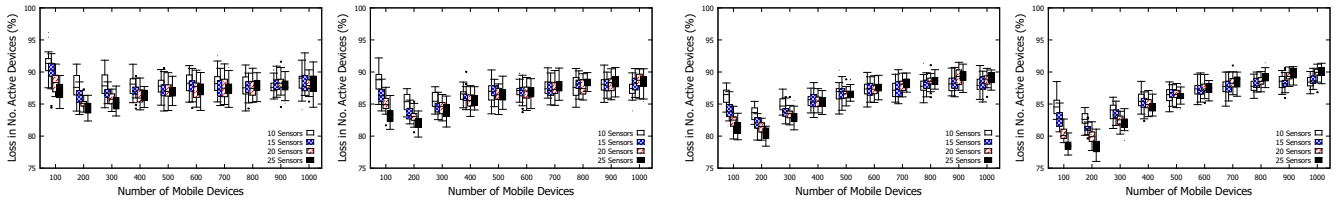
**Time Complexity Analysis:** We now present the worst case running time analysis for our algorithm when each application requests for sensed data from all sensor-types. Assuming that the maximum number of request locations to be covered by one application is $l$ which contains a maximum $d < |\mathcal{N}|$ mobile devices for the time interval $k$, the number of NodeSets for each Application-Sensor pair will be $d^l$. The complexity of the Algorithm depends on the complexity of the FP-Growth algorithm is which is proportional to the number of unique elements $d \cdot l$ present in the header table and the depth of the FP-Tree. In the worst case, the tree is an unbalanced tree and its depth is upper-bounded by $d \cdot l$. Thus the complexity of the algorithm is $O(d^2 \cdot l^2)$. Using the FP tree ensures that the complexity is much less than searching through all possible combinations which is $2^{dl}$.

**Algorithm 1** Assisted-Aggregation

```
1:  for k=1 to T/Δt do
2:      t.start=(k − 1)Δt;
3:      t.end=kΔt;
4:      for n =1 to N do
5:          predictLocOfMobileDevice(n, t.start, t.end);
6:      end for
7:      AppSensorPairs[]=getAllAppSensorPairs(k);
8:      for AppSensor ∈AppSensorPairs[] do
9:          loc l[]=getLocs(AppSensor,k);
10:         NodeSets=CALCNODESETFORLOC(l[],k);
11:     end for
12:     Tree T=FP-TREE(NodeSets);
13:     FP P=FP-GROWTH(T.root, null)
14:     maxSizes[]= calcTwoHighestForMostFrequent(P)
15:     for Set pinP do
16:         if p.size ∈ maxSizes[] & hasBattery(p)=true then
17:             fSet=fSet ∪ p
18:         end if
19:     end for
20:     CALCAPPSENSOR(fSet, AppSensorPairs[]);
21: end for
22: Run calcEnergyConsumption();
23: Run calcVolumeOffloaded();
24: function CALCAPPSENSOR(fSet, AppSensorPairs[])
25:     for AppSensor ∈ AppSensorPairs[] do
26:         NodeSet=getSetWithMinDist(AppSensor, fSet);
27:         for all n ∈ NodeSet do
28:             reportThreshold[]= setReportingDevice(n)
29:             list = getNeighbours(n)
30:             rawValue= preProcess(n)
31:             if rawValue ∈ reportThreshold[] then
32:                 offloadAcctoState()
33:             else
34:                 getStateFromNeighbours(list)
35:                 offloadAcctoState()
36:             end if
37:         end for
38:     end for
39: end function
```

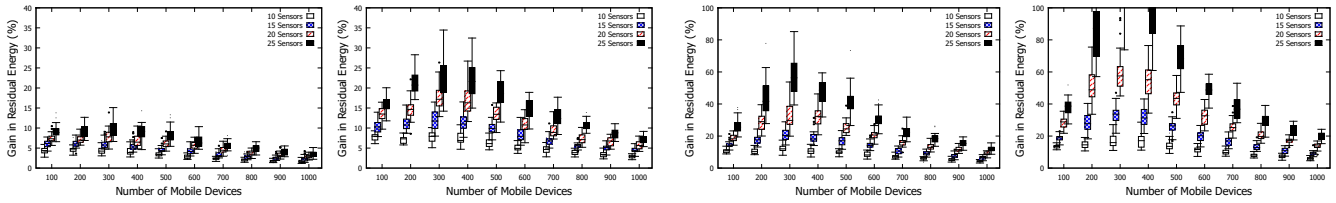
## V. EVALUATION

This section describes the performance assessment of our algorithm, 'Assisted-Aggregation'. The previously modelled and studied 'Info-Aggregation' algorithm [21], [22] had also identified the scope of aggregating and serving sensor data to multiple applications from one device but the ability to



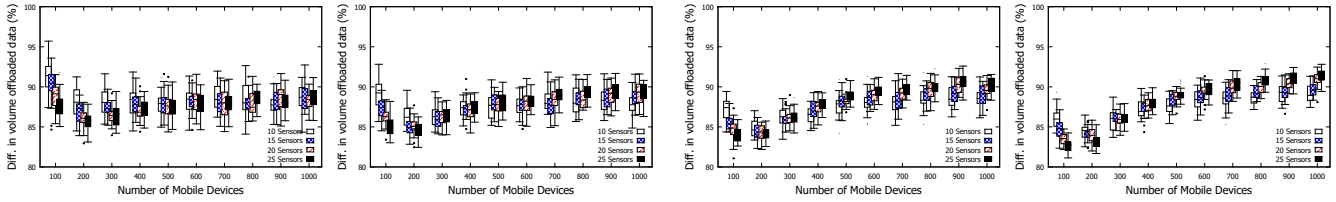

(a) 50 apps, mobile devices (b) 100 apps, mobile devices (c) 150 apps, mobile devices (d) 200 apps, mobile devices

Fig. 3. Comparative percentage difference in the number of active mobile devices between Info-Aggregation and Assisted-Aggregation vs. the total number of mobile devices in the sensing environment for four cases using 50, 100, 150 and 200 applications. We see that in all cases Assisted-Aggregation requires the activation of significantly fewer mobile devices.




(a) 50 apps, mobile devices (b) 100 app, mobile devices (c) 150 apps, mobile devices (d) 200 apps, mobile devices

Fig. 4. Comparative percentage difference in the cumulative residual energy held in mobile device batteries between Info-Aggregation and Assisted-Aggregation vs. the total number of mobile devices in the sensing environment for four cases using 50, 100, 150 and 200 applications. We see that Assisted-Aggregation results in a lower level of cumulative energy use by mobile devices, due both to the lesser reporting mobile devices and to a lower number of messages with sensed data being transmitted due to the use of aggregation.




(a) 50 apps, mobile devices (b) 100 apps, mobile devices (c) 150 apps, mobile devices (d) 200 apps, mobile devices

Fig. 5. Comparative mean percentage difference in the volume of data offloaded by two algorithms, Info-Aggregation and Assisted-Aggregation vs. total number of mobile devices in the sensing environment over four cases using 50, 100, 150 and 200 applications. The use of aggregation and state machines means that the Assisted-Aggregation algorithm offloads a lower volume of data than does Info-Aggregation.

tailor reporting needs of the application had not been explored. By updating the parameters for the IoT scenario, a comparison with this algorithm enables us to quantify the 'Assisted-Aggregation' algorithm, which exploits the multi-tasking capability of a device and uses application-specific state-machines for reporting.

### A. Simulation Model

The values/parameters used to define the simulation model are based on the scenario outlined above. The physical sensing area is modelled using a grid (100m × 100m) subdivided into request locations (10m × 10m), considering the dimensions of an average house in Ireland/United Kingdom[1]. This represents our simulation study area and bounds the trajectory of each mobile device. We have chosen the Truncated Levy-Walk mobility model [34], represented by the tuple $(l, \theta, t_f, t_p)$ to determine the path of a mobile device. Here, $l$ is the flight length randomly picked up from a Levy distribution with coefficient $\alpha = 1.5$, $\theta$ is the angle of flight which follows a uniform distribution, $t_f$ is the flight time, and $t_p$ is the pause time which is Levy distributed with coefficient $\beta = 0.5$. The truncation factors are defined as 100m and 1000s respectively for the flight length and pause times. One instance of the model is used to define the real path taken by the mobile device during the simulation study while another instance is used to predict the path taken by the mobile device for the next time-interval. These values are motivated by the fact that the mobile devices are within a mean value of one metre from their original position when a time interval has elapsed. These time-intervals of $\Delta t = 2$ minutes represent time progression within the simulation over a total simulated duration of $\mathcal{T} = 240$ minutes.

An individual run is identified by a fixed number of applications $|\mathcal{A}|$, mobile devices $|\mathcal{N}|$ and the maximum sensor-types in a location/mobile device $\mathcal{H}$, which is taken to be equal to the maximum number of sensor-types requested by

[1]How Big is a House? http://shrinkthatfootprint.com/how-big-is-a-house

an application. It is further assumed that each location has at most two static sensors with the same sensor-type. Each of these applications, mobile devices and sensor-types are identified by using a unique integer id and the sensor-types in a location/mobile-device are randomly selected using a uniform distribution.

At the beginning of every simulation run, a mobile device is assigned a maximum energy of 5Wh[2] which decreases over time, attributed to energy for general usage, for accessing the embedded sensors and static sensors, and for transmission of offloaded sensed data. Every embedded sensor-type that is accessed from this mobile-device contributes to the loss of battery which is specific to the sensor-type. Sensirion offers environmental sensors for mobile devices with energy consumption as low as $2\mu$W[3] while LittleRock [29] presents a table on power consumed by different sensor-types. We randomly select the energy consumption for the sensor-type in the range 0.002mW to 2.24mW [29] to cover the different sensor-types. The percentage of decrease for general usage of the mobile device is randomly selected. Another variable associated with the sensor-type is the volume of sensed data that it collects over time, this is randomly selected between 8 bits/second[4] to 50 bits/second.

The static sensors in the location are accessed by the mobile device using Bluetooth, only if the embedded sensors cannot provide sensor data with the accuracy needed by an application. The collected data is then processed by the mobile device. For our 'Assisted-Aggregation' algorithm, the reporting device first accesses its factored state machine for that sensor to determine whether the data needs to be offloaded. In case, the pre-processed data is found to be beyond the reporting thresholds saved in the device, it starts a simultaneous dialogue with the other candidate reporting devices for the location using WiFi-Direct [30], [31]. This supports one-2-one and one-2-many operations over WiFi-enabled mobile devices but does not require a WiFi access point, allowing peer-2-peer transmissions between the mobile devices. For energy transmission calculations, the mobile device loses 0.05W [2] to maintain WiFi connections. For accessing static sensors over Bluetooth, interacting with the candidate reporting devices using WiFi direct or transferring sensed data to the cloud, we use the transmission energy as presented by Friedman *et al.* [8] across Bluetooth, WiFi (ad-hoc and with access-points) to send/receive data. We randomly select one of the communication protocols for WiFi networks(ad-hoc or access-points) to cover different transmission channels.

Each Application-Sensor pair defines different request locations within the grid, dependant on location constraints, that need to be covered by mobile devices. We have limited the number of request locations to a maximum of four locations for each pair in our study. The sensing time period and minimum accuracy for the pair are randomly picked from uniform distributions. Each request location contains a consolidated state machine for every sensor-type and it is assumed that the probability of offloading sensor-data for an Application-Sensor pair in a time-interval, is 20%. This assumption helps in defining the reporting thresholds for the Application-Sensor pair.

[2]Apple iPhone: https://www.apple.com/iphone/

[3]Sensirion- New Dimensions in Environmental Sensing: http://www.sensirion.com/en/mobile-solutions/environmental-sensing/

[4]SHT2x - Digital Humidity & Temperature Sensor (RH/T) by Sensirion: http://www.sensirion.com/en/mobile-solutions/environmental-sensing/

### B. Results and Analysis

With the advent of the Internet of Things, the number of sensors that can provide useful data and the number of context-aware applications that make use of this data is increasing. Considering this, we varied the number of applications between 50, 100, 150 and 200 applications, requesting for sensed data between 10, 15, 20 and 25 unique sensor-types. The number of mobile devices in our experiments that sense, collect, pre-process and offload this sensor data is also varied in the range of 100 to 1,000. Each experiment was run 30 times using different random number generator seeds and the performance of the Assisted-Aggregation algorithm in comparison with the Info-Aggregation algorithm was recorded. The results are presented as the percentage difference in the values for the two algorithms in terms of the gain in residual battery values of all mobile devices, the difference in volume of data offloaded and the difference in the mean number of mobile devices that report the sensed data during the planning horizon, in each case. The boxplot representations of these results allow us to visualize the confidence intervals as well as the distribution of our experimentation.

Fig. 3 shows the mean reduction in the number of mobile device reporting sensed data to the cloud while Fig. 5 shows the mean percentage difference in the cumulative volume of data offloaded for for the Assisted-Aggregation algorithm in comparison to the Info-Aggregation algorithm. These results show that more than 80% reduction can be achieved by aggregating sensed data from mobile devices for multiple applications and by incorporating state-machines to determine reporting needs of these applications, for both parameters. Additionally, this also highlights how the mean cumulative residual energy present in mobile devices can be saved as depicted in Fig. 4. This is attributed to the decreased number of reporting mobile devices as well as to the fewer message transmissions due to the use of state-machines. The effect is amplified as the number of applications increase. Additionally, it can also be observed that the cumulative energy gain decreases as the number of mobile devices in the sensing area increase. This relates to the increase in the number of mobile devices capable of covering one location. For improved global energy-efficiency, both algorithms select a larger set of active mobile devices to cover application requirements, which decreases the energy gain for Algorithm Assisted-Aggregation, despite the difference in the number of active mobile devices offloading sensed data between the two algorithms. Further data analysis and model implementation for improved energy-efficiency will be done as part of future work.

## VI. Conclusion

The diverse collection of sensors embedded within mobile devices or present in surrounding IoT devices is gaining increasing interest with researchers for developing innovative context-aware applications and collaborative sensing platforms. The design of such systems highlights the important technical challenge of optimally selecting mobile devices for accurately satisfying application constraints in an energy-efficient manner, whilst ensuring proper emulation of mobile device movements. Our approach of using application-specific state machines to determine reporting constraints of applications and using a revised frequent pattern mining algorithm, 'Assisted-Aggregation' succeeds in delivering a significant improvement in terms of energy utilisation and reducing volume of offloaded sensed data.

A natural extension to this work is the integration of dynamically adapting sensing rates embedded into the application-specific state-machines for further energy improvements. Additionally, we will also study how the collaborative framework can respond and adapt to more complex state machine diagrams, along with considering additional performance metrics apart from energy-utilization. Furthermore, enhanced data analysis and model improvement is required to completely understand the decrease that is noticed with cumulative energy gain and the trends observed with the number of active devices and volume of offloaded data, when the Info-Aggregation algorithm is compared with the Assisted-Aggregation algorithm. Lastly, future work would include comparison with other alternative solutions, study of network issues relating to delay/congestion and adoption of security/privacy features. This includes, but is not limited to the identification of malicious devices in the network, maintenance of precision of sensed data delivered to the applications, restriction on a device to join the network when posing a threat and ensuring encryption techniques to ensure sensitive data is not leaked. Lastly, we will work on machine learning strategies that understand the application needs to automatically create/improve the state machines along with real-data analysis.

## Acknowledgements

This work was funded by: 1) the Irish Research Council Enterprise Partnership Scheme Postgraduate Research Scholarship, co-funded by Intel Labs Europe (grant no. EPSPG/2012/407); 2) by the Irish Research Council via the ELEVATE Fellowship 2013 (grant no. ELEVATEPD/2013/26); and 3) by Science Foundation Ireland (SFI) via the CONNECT Research Centre (grant no. 13/RC/2077).

## References

[1] G. Ananthanarayanan, V. N. Padmanabhan, L. Ravindranath, and C. A. Thekkath. COMBINE: Leveraging the power of wireless peers through collaborative downloading. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, MobiSys '07, pages 286–298. ACM, 2007.

[2] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *Proceddings of the 9th ACM SIGCOMM Conference on Internet Measurements Confernce*, IMC, pages 280–293, 2009.

[3] F. Ben Abdesslem, A. Phillips, and T. Henderson. Less is More: Energy-efficient Mobile Sensing with Senseless. In *Proceedings of the 1st ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds*, MobiHeld '09, pages 61–62. ACM, 2009.

[4] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn. The Rise of People-Centric Sensing. *IEEE Internet Computing*, 12(4):12–21, July 2008.

[5] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson. People-centric urban sensing. In *Proceedings of the 2nd Annual International Workshop on Wireless Internet*, WICON. ACM, 2006.

[6] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: Privacy-aware people-centric sensing. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 211–224. ACM, 2008.

[7] S. Devadas and A. R. Newton. Decomposition and factorization of sequential finite state machines. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(11):1206–1217, Nov. 1989.

[8] R. Friedman, A. Kogan, and Y. Krivolapov. On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones. *IEEE Transactions on Mobile Computing*, 12(7):1363–1376, July 2013.

[9] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, Aug. 2007.

[10] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, Apr. 2006.

[11] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[12] 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. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 267–280. ACM, 2008.

[13] W. Khan, Y. Xiang, M. Aalsalem, and Q. Arshad. Mobile Phone Sensing Systems: A Survey. *IEEE Communications Surveys Tutorials*, 15(1):402–427, 2013.

[14] K. Kumar and Y. H. Lu. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51–56, Apr. 2010.

[15] M. Kurz, G. Holzl, A. Ferscha, A. Calatroni, D. Roggen, G. Troster, H. Sagha, R. Chavarriaga, J. del R. Millan, D. Bannach, K. Kunze, and P. Lukowicz. The OPPORTUNITY framework and data processing ecosystem for opportunistic activity and context recognition. *International Journal of Sensors Wireless Communications and Control*, 1(2):102–125, Dec. 2011.

[16] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[17] S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, and P. Demeester. City of things: An integrated and multi-technology testbed for iot smart city experiments. In *Smart Cities Conference (ISC2), 2016 IEEE International*, pages 1–8. IEEE, 2016.

[18] D. Liaqat, S. Jingoi, E. de Lara, A. Goel, W. To, K. Lee, I. De Moraes Garcia, and M. Saldana. Sidewinder: An energy efficient and developer friendly heterogeneous architecture for continuous mobile sensing. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 205–215. ACM, 2016.

[19] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 285–298. ACM, 2010.

[20] R. Loomba, L. Shi, and B. Jennings. State-machine driven opportunistic sensing by mobile devices. In *2014 IEEE Global Communications Conference (GLOBECOM)*, pages 2739–2744, Dec. 2014.

[21] R. Loomba, L. Shi, B. Jennings, R. Friedman, J. Kennedy, and J. Butler. Information Aggregation for Collaborative Sensing in Mobile Cloud Computing. In *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 149–158, Apr. 2014.

[22] R. Loomba, L. Shi, B. Jennings, R. Friedman, J. Kennedy, and J. Butler. Energy-aware collaborative sensing for multiple applications in mobile cloud computing. *Sustainable Computing: Informatics and Systems*, 8:47–59, Dec. 2015.

[23] H. Lu, D. Frauendorfer, M. Rabbi, M. S. Mast, G. T. Chittaranjan, A. T. Campbell, D. Gatica-Perez, and T. Choudhury. StressSense: detecting stress in unconstrained acoustic environments using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 351–360. ACM, 2012.

[24] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 71–84. ACM, 2010.

[25] 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. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 337–350. ACM, 2008.

[26] A. Mtibaa, A. Fahim, K. A. Harras, and M. H. Ammar. Towards Resource Sharing in Mobile Device Clouds: Power Balancing Across Mobile Devices. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing*, MCC, pages 51–56, 2013.

[27] S. Nath. ACE: exploiting correlation for energy-efficient and continuous context sensing. *IEEE Transactions on Mobile Computing*, 12(8):1472–1486, Aug. 2013.

[28] T. Penner, A. Johnson, B. Van Slyke, M. Guirguis, and Q. Gu. Transient clouds: Assignment and collaborative execution of tasks on mobile devices. In *Proceedings of the IEEE Global Communication Conference (GLOBECOM)*, pages 2801–2806, 2014.

[29] B. Priyantha, D. Lymberopoulos, and J. Liu. LittleRock: enabling energy-efficient continuous sensing on mobile phones. *IEEE Pervasive Computing*, 10(2):12–15, Apr. 2011.

[30] A. Pyattaev, K. Johnsson, S. Andreev, and Y. Koucheryavy. 3GPP LTE traffic offloading onto WiFi Direct. In *Proceedings of the IEEE Wireless Communication and Networks Conference Workshops (WCNCW)*, pages 135–40, 2013.

[31] A. Pyattaev, K. Johnsson, S. Andreev, and Y. Koucheryavy. Proximity-Based Data Offloading via Network Assisted Device-to-Device Communications. In *Proceddings of the 77th IEEE Vehicular Technology Conference (VTC Spring)*, pages 1–5, 2013.

[32] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow. SociableSense: Exploring the Trade-offs of Adaptive Sampling and Computation Offloading for Social Sensing. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 73–84. ACM, 2011.

[33] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho. Urban planning and building smart cities based on the internet of things using big data analytics. *Computer Networks*, 101:63–80, 2016.

[34] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong. On the levy-walk nature of human mobility. *IEEE/ACM Transactions on Networking*, 19:630–643, June 2011.

[35] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning. Saving portable computer battery power through remote process execution. *SIGMOBILE Mobile Computing and Communications Review*, 2(1):19–26, Jan. 1998.

[36] S. Sarker, A. K. Nath, and A. Razzaque. Tradeoffs between sensing quality and energy efficiency for context monitoring applications. In *2016 International Conference on Networking Systems and Security (NSysS)*, pages 1–7, Jan. 2016.

[37] H. Shah-Mansouri and V. Wong. iCoMe: A novel incentivized cooperative mobile resource management mechanism. In *Proceedings of the IEEE Global Communication Conference (GLOBECOM)*, pages 4996–5001, Dec. 2014.

[38] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura. Serendipity: Enabling Remote Computing Among Intermittently Connected Mobile Devices. In *Proceedings of the 13th ACM International Symposium on Mobile Ad-Hoc Networking and Computing*, MobiHoc, pages 145–154, 2012.

[39] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos. AnonySense: A system for anonymous opportunistic sensing. *Pervasive and Mobile Computing*, 7(1):16–30, 2011.

[40] L. Tang, S. He, and Q. Li. Double-sided Bidding Mechanism for Resource Sharing in Mobile Cloud. *IEEE Transactions on Vehicular Technology*, in press, 2016.