A Review on Energy-Efficient Mobile Sensing

Tianli Mo
Information and Computer Science, University of Hawaii at Manoa

January 20, 2014

Abstract

Mobile sensing has emerged rapidly in the past years as a promising avenue for collecting, leveraging and querying various information around users. One reason is that smartphone or mobile device has increasingly become the central computing device in peoples daily lives, the other reason is that more affordable sensors have been embedded in smartphones. Since smartphones are battery powered devices, we have to face the critical problem of how to reduce energy consumption of mobile sensing in order to extend the operational lifetime of smartphones. In this paper, we first present the general problems we have encountered on energy-efficient mobile sensing. Second, we break down the general approaches for saving battery power. Then we introduce a comprehensive taxonomy of energy-efficient saving schemes. Based on the schemes we further survey existing solutions and techniques, as well as dicuss unsolved problems. We then conclude the paper with insights for future research directions in energy-efficient mobile sensing.

1 Introduction

Over the past years, smartphones have increasingly become the central computing and information device in peoples lives. One of the reasons is because of the various and diverse applications that are available in the Apple AppStore or Google Play. Another reason is that smartphones today come with a growing set of cheap and powerful embedded sensors such as accelerometer, GPS, microphone, digital compass and camera. These sensors are enabling different types of personal, group,

community and even global scale sensing applications (e.g. Cougar [2], TinyDB [28], HAMONI [32] and CenceMe [30]).

Smartphone sensing systems operate at multiple scales [22] . Personal sensing applications are designed for a single individual. Persona sensing applications, such as medical monitoring applications typically generate sensor data for sole consumption on the individual and are not shared with others. On the other hand, group sensing applications or community sensing applications are likely to share the local data within the social network or related groups. Increasingly applications are concerned more about the information among large populations of people rather than individual person. Therefore in the future applications are likely to not only care about the local sensor data on individual phone, but also the sensor data from large groups of phones that may capture information about the surrounding environment.

Opportunistic and participatory sensing are two modalities of how people are involved in mobile sensing applications. For opportunistic sensing, the sensors continuously collect information to meet application request without active user involvement [21]. In other words, usually the sensors run in the background and the users might not be aware of their status. Participatory sensing requires the user to be actively involved in the sensing operations: the user actively turns on the sensing, decides whether the sensing is shared, and decides what privacy mechanisms to use.

Users have a great deal of interest in recent years in using these kind of mobile sensing applications. These applications have one thing in common: collecting the data of various embedded sensors from time to time. Sensors usually consume considerable amount of power which results in the short battery lifetime in smartphones. Studies show that continuous sensing drains a smartphone battery in as little as 5-6 hours. Moreover, some applications perform continuous round-the-clock sensing on smartphones further exacerbating the problem. In addition, the mobile sensing applications also require a large amount of wireless communication (e.g. 3G, WiFi and Bluetooth) between peer and peer, or between phone and cloud base. Besides embedded sensors and networking communication, another portion of power is consumed by hardware computation. Computation energy consumption mainly is the cost of CPU to process the operation and service to support running the applications. In addition, the screen of a smartphone also consumes a large portion of total energy on smartphone [4]. However, energy consumption of screen is beyond the scope of mobile sensing focus therefore we do not spend time to discuss the energy consumption of screen or energy consumption of any other irrelevant hardware.

Energy consumption of sensors. Sensors become energy greedy components which continuously consume a large amount of power in smartphones. However, en-

Feature	Avg. Power [watt]	Std. Dev. [watt]
Idle (i_p)	0.0621	0.0173
Idle (i_p) + Logging	0.0647	0.0197
Accelerometer (a_p)	0.0503	0.035
GPS (g_p)	0.324	0.0435
Radio idle (r_p)	0.466	0.0324
Radio active (s_p)	0.645	0.0470

Figure 1: Energy comsuption of different sensors

ergy consumption varies from sensor to sensor. For example, the GPS uses a varying amount of power depending on factors such as the number of satellites available and atmospheric conditions. Therefore GPS usually consumes more power than, lets say, a digital gyroscope does within a same operation period. The energy consumption comparison table of accelerometer, GPS, communication radio is shown in Figure. 1. The communication radio consumes more energy than the accelerometer and GPS do. Moreover, the average energy consumption of an accelerometer is much lower than GPS. In general, GPS is much more expensive than most other sensors, therefore carefully use of GPS sensor can help to save battery power. Different sensors consume different amout of absolute energy on different platforms. For instance, accelerometer sensor on an iphone does not consume the same amount of power as an accelerometer sensor on a Samsung smartphone. However, we only focus on the relative amount of energy consumed by the different sensors on give a particular platform.

Energy consumption of networking technologies. Most of mobile sensing applications use some wireless networking technologies to transfer data. Therefore, it is important to take the energy consumption of wireless networking interface into account. 3G, GSM, Bluetooth and WiFi are popular networking technologies widely used around the world. Similarly, the energy consumption of these technologies is also different. For example, Bluetooth communicate consumes less energy than 3G or WiFi, even though Bluetooth has limited range. Table 2 shows a number of real energy measurements from Android smartphones (HTC with Andorid 2.3 and a Qualcomm QSD8250 ArMv7 1-Ghz processor using the benchmarking tools MobiPerf (www.mobiperf.com) and PowerTutor (powertutor.org) [5]. From the table we can find that WiFi localization consumes less energy than 3G localization.

Energy consumption of computation. Table. 2 shows how the energy profile of PARTICULAR SMARTPHONE OF TYPICAL USE. From the table, we obsever the energy consumed by CPU during heavy processing is comparable to

Basic smartphone operation	Megawatts of power (mW = mJ/s)
CPU minimal use (just OS running)	35
CPU standard use (light processing)	175
CPU peak (heavy processing)	469
Wi-Fi idle (connected)	34
Wi-Fi localization (avg/minute)	125
Wi-Fi peak (Uplink 123Kbps, -58dBm)	400
3G localization (avg/minute)	300
3G busy	900
GPS on (steady)	275
OLED economy mode	300
OLED full brightness	676

Figure 2: Energy consumption of different components on smartphones

WiFi peak usage. However, the CPU processing for mobile sensing tasks typically uses not much energy compared to sensors and communications.

From above, we know that sensors, network communication and computation play significant roles of mobile sensing and all of them consume considerable power. In general, because sensor sensing and networking communication consumes more power, so there is more opportunity for conducting research into making these operations more energy-efficient.

Table. 3 shows a comparison of energy consumption for detecting various context using sensors and wireless communications. Context sensing is the use of mobile sensing to infer the activities of the user. Some example of user contexts are whether the user is diving, whether he/she is at home or he/she is in a meeting, etc. Some contexts only require data from single sensor, while others require data from multiple sensors. The data in the table shows that detecting context using accelerometer is much cheaper than WiFi, GPS and Microphone. Such differences from energy consumption presents an opportunity for research on optimizing the use of different sensors.

In this review, we analyze the recent studies relevant to energy-efficient mobile sensing. By discussing the ideas, approaches and implementations, we aim to:

- Generalize the problems that researchers want to tackle in mobile sensing.
- Map out the existing solutions for energy-efficient sensing.

Attribute	Short	Sensors used (sample length)	Energy (mJ)
IsWalking	W	Accelerometer (10 sec)	259
IsDriving	D	Accelerometer (10 sec)	259
IsJogging	J	Accelerometer (10 sec)	259
IsSitting	S	Accelerometer (10 sec)	259
AtHome	H	WiFi	605
InOffice	O	WiFi	605
IsIndoor	I	GPS + WiFi	1985
IsAlone	A	Microphone (10 sec)	2895
InMeeting	M	WiFi + Microphone (10 sec)	3505
IsWorking	R	WiFi + Microphone (10 sec)	3505

Figure 3: Energy comsuption of sensing different context

- Identify and categorize the similarity and differences of the different approaches.
- Highlight the areas where additional research is required.

In the rest of the paper, we first briefly introduce mobile sensing applications in different categories in section 2. In section 3, we discuss the general approaches toward energy-efficient mobile sensing. We use section 4, 5, 6 and 7 to survey on the detail techniques of state of the art approaches on four major aspects. In section 4, we discuss the existing techniques for data reduction including data transmission reduction and data predication. In section 5, we introduce energy-efficient data acquisition techniques. Query processing optimization and collaborative sensing is discussed respectively in section 6 and section 7. Before drawing conclusions, we present open issues and challenges on crowdsourcing in section **??**. In the last section, we conclude the article by summarizing the strategies and discuss the future work.

2 Mobile sensing application

2.1 Location-based Applications

A location-based application is an application that users use smartphone's location. Location-based applications enable social networking and serendipitous interaction by people checking in at the various places they visit e.g., local restaurant, airport etc. The location sensing also can be used by location-based applications to provide route or localization for user navigation. For example, Waze provides users free navigation by using real-time traffic and road information generated from other users.

Also, location sensing is actively involved in location sensitive and advertisement applications. For example, Locale allows smartphone to change its settings automatically based on the current location. Google Ads send advertisements to target potential users based on the users locations. Cai et al. [3] propose an application aims at bridging the Internet with the physical world. Users can use their smartphone to sense ambient data associated with their location then share it to other users whose capable zone (a circular region centered on its current position with a user-defined radius) is close to the date geographic region. Some other LBAs such as Wechat aims at helping smartphone user to implement buddy finder, where user can request the position of their buddies (buddies may be selected based on location and preferences, assume that buddies agree to share location). This function usually has been integrated with social network applications. Position tracking is another service (Google Map, Waze, etc.) broadly used to generate trajectories of users for personal uses or community use. It may exploit multiple sensors to include GPS, accelerometer and WiFi [16]. With the increasing development of location-based applications, people found that any kind of data associated with location information are more meaningful and promising. Therefore, LBA becomes one significant mobile application category that earns more and more research attention.

2.2 Social Networking

Moible sensing is also used in social network to provide information of the users and their environment. Besides the location sensing, other sensed data such as Microphone, GPRS and Bluetooth can present the users' status. Social network application requires this kind of mobile sensing involved to provide rich sensors information of the individual users, user groups and their environment. Miluzzo et al. [30] design an application called CenceMe that combines the inference of the presence of individuals using sensor-enabled smartphone with sharing of sensing information via social networking applications such as Facebook and MySpace. CenceMe uses multiple sensors including GPS & Microphone, and networking interfaces include GPRS & Bluetooth.

2.3 Healthcare Applications

Mobile sensing can also enable many healthcare applications as well. Mohomed et al. [32] introduce one kind of compelling application—HARMONI, a long-term

remote health monitoring that is able to perform context-aware processing and event filtering on data generated from multiple body-worn mobile sensors.

Smailagic et al. [44] employ context-aware mobile computing with wearable sensors to obtain information about the users then use the information to analyze user behavior, without requiring the user's attention. It advocates using the personal cellphone as an intermediate relay, which transports health data streams from multiple body-worn sensors to a backend analytics infrastructure. A doctor is able to access the health data through backend infrastructure and then act on the data accordingly. For example, a smartphone carried by an elder patient can be used as a pedometer to monitor his exercise and send daily reports to his health database. In addition, it can detect if the patient has fallen on stairs then automatically call for emergency help.

2.4 Enviroment Applications

Mobile sensing also can be used for real-world environmental monitoring with wireless sensor that benefits for scientific communities and society [44]. Monitoring the numerous mobile sensors carried by a large number of users can enable long-term data collection at scales and resolutions that are difficult to obtain previously. Network Infomachanical System(NIMS) [39] provides the ability to explore large volumes of environment information.

3 General approaches to energy-efficient mobile sensing

In this section, we discuss the general approaches to energy-efficient mobile sensing. A taxonomy of the state of the art approaches are shown in Figure 4. Each node denotes a category, and chireden nodes denote the sub-category. There are four main categories: data reduction, energy-efficient data acquisition, query processing optimization and collaborative sensing.

3.1 Data Reduction

Data reduction approaches aim at reducing the amount of data sensed or transmitted. All these techniques can be divided according to the problem they address.

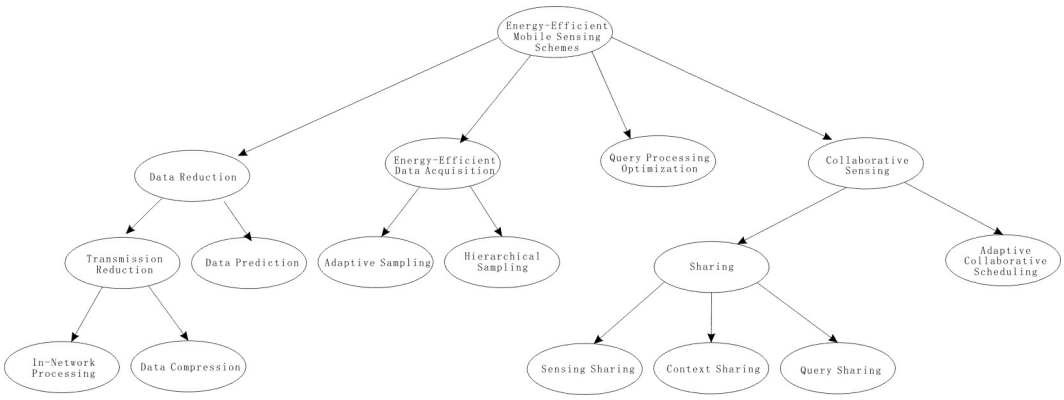


Figure 4: Taxonomy of the state of the art approaches

Specifically, data transimission reduction address the case of unneeded sensor data when transfering from peer to peer or smartphone to cloud, thus reducing the energy consumption of transmission and communication. General data transmission reduction uses a filter mechanism to eliminate the unneeded sensor data while acquiring sensor data for mobile sensing application. This type approach is able to filter out most unnecessary raw sensor readings thus saving power used by networking communication.

On the other hand, data predication techniques try to build an abstraction of a sensed phenomenon, i.e., a model that is able to describe sensed data evolution. In other words, the model can predict the sensed data without actually sensing or acquiring it, while maintaining the predicted values within certain error bounds. Data predication approaches usually predict the values of sensor data by analyzing the trend of arriving data. Data predication approaches can be used in cases which are critical to energy saving while not sensitive to accuracy requirement. In fact, many approaches consist in trading off the accuracy for the energy-efficiency.

3.2 Energy-efficient data acquisition

Energy-efficient data acquisition aims at reducing the number of acquisitions. In other words, energy-efficient data acquisition reduces the request for sensor data from the from the applicaiton side. Compared to previous data reduction approaches, it consists in reducing the data sampled by sensors. Not only is energy saved on sensors, the communications energy consumption also decrease accordingly. There are two sub-categories of energy-efficient data acquisition: adaptive sampling

and hierarchical sampling.

Adpative sampling is designed to exploit the spatio-temporal corelations similarities between data to reduce the number of sensor data to be sensed. While hierarchical sampling approach exploits the multiple sensors potential to generate optimized resolution associated with energy consumption. The techniques of hierarchical sampling can daynamically choose appropriate sensors to be activated, in order to trade off accuracy for energy conservation.

3.3 Query Processing Optimization

Some mobile sensing applications allow the user to perfrom queries on sensor data and other data in the cloud. Therefore, query processing optimization is another research aspect to address energy consumption problems. Sensing system can reduce the sensing energy consumption as well as computation energy consumption by generating optimized query plan. Optimized query plan selects the appropriate alternatives sensors and carefully manages the sensing schedules to meet the applications demands. One issue is that different applications may have different architecture and protocol resulting in the need for different techiques to optimize the query plan (e.g., in-network processing and data aggregation). An optimized query plan can effectively perform queries on target sensor data to avoid unnecessary data acquisition.

3.4 Collaborative Sensing

A recent approach of mobiling sensing is the shift of mobile sensing done on single smartphones to mobile sensing done in a large group of smartphones. When mobile sensing is peformed in a group, there is a potential for some of the query processing and some of the sensor data to be shared in a collaborative fashion. Collaborative sensing is the technique category designed to take advantages of similarity of the sensor data in the nearby smartphones to minimize the energy consumption including sensing, transimission and computation.

Collaborative sensing can be divided into two main sub-categories: sensing sharing and adaptive collaborative scheduling. In addition, sensing sharing can be further divided into three types: data sharing (raw data), state sharing (context sharing) and query sharing (similar sub-queries). Phones can use cheaper networking techniques such as bluetooth to establish the collaboration involved with those differents types of sharing. Moreover, similar to hierarchical sampling, there are many

opportunities to exploit the scheduling among multiple sensors and moble phones in order to maximize the energy-efficiency.

3.5 Discussion

Different kinds of approaches are designed to reduce energy consumption from different aspects. Data reduction approaches focus on reducing the amount of data generated at the sensor level. Energy-efficient data acquistion approaches usually reduce the amount of data required at application level. Query processing optimization reduce energy consumption by optimizing query execution. Collaborative sensing approaches reduce energy consumption by exploiting the chance of collaboration among phones. There is not an "one size fits all" approach that achieve the most energy saving. How to select from those approaches should be based on the specific scenario or condition. In the following sections, we survey the four distinct approaches in details.

4 Data Reduction

4.1 Data Transmission Reduction

Many mobile sensing applications require data transfer from peer to peer (P2P), phone to cloud or in hierachical fashion. However, communication consumes significant amounts of energy competed to computation and sometimes even more than sensing operation. In order to save energy, it is necessary to reduce the energy consumption of communication. In this section, we introduce several techniques (in-network processing and data compression) to reduce data transmission during sensing tasks.

Before discussing the techniques that used in mobile sensing, we first present some techniques that used in distributed sensor network (DSN). Distributed sensor network is a set of spatially scattered intelligent sensors for collecting measurements from the data gathered, and to derive appropriate inferences from in formation gained. Because distributed sensor network has many common characteristics as mobile sensing scenario, such as they all use sensors to collect data, each node (sensor or smartphone) is associated with location information and they both have the battery hungry issue. Hence, certain techniques of distributed sensor network can inspire the approaches of mobile sensing energy-efficient.

Distributed sensing network often requires multiple sensors to broadcast their measurements (environmental parameters or states) to the network and users application can acquire the specific data they desire. For this scenario, Mu et al. [33] present Value of Information realized Distributed Sensing (VoIDS) algorithm in Bayesian inference framework that enables efficient finding of $informative$ sensors with high Value of Information (VoI), and allow them to broadcast their measurements to networks in order to prevent broadcast energy consumption of agents with low VoI. It first pick a standard information metric (KL divergence) to measure the the VoI, then compute the VoI for each sensor by using the information metric and Bayesian inference. If the VoI exceeds a adjusted threshold, then the sensor label itself as informative and update their update readings to neighbors. This adaptation enables the mobile system to better balance between the communication cost incurred and the long term accuracy of the estimation.

Heinzelman et al. [12] propose a clustering-based protocol Low-Energy Adaptive Clustering Hierarchy (LEACH) which utilizes [localized coordination] to enable robustness and scalability for dynamic networks, and incorporates [data fusion] into the routing protocol in order to reduce the amount of information that must be transmitted to the base station. Moreover, LEACH performs local computation in each cluster to reduce the amount of data that needs to be transmitted to the base station. This strategy achieves a significant reduction in the energy consumption, as computation is much cheaper than communication.

Different from VoIDS and LEACH which consider reduction of energy consumption for communication, [26] not only studies communication energy consumption independently but also takes the energy consumed by sensing process. It focuses two types of energy constraints: 1. the sensor has a given energy budget used for sensing and data compression while a separate energy budget for communication. 2. Both tasks (sensing and communication) share a given single energy budget. For these two energy constraints, they come up with adaptive strategies respectively to minimize the overall energy consumption.

Compression is another way to reduce the amount of data that needs to be transmitted. There are many compress sensing techniques have been used for this purpose. For example, SRM (Structurally Random Matrix) [8] is designed for large-scale, real-time compressive sensing application that efficiently acquiring and reconstructing sensor data with compressibility.

Spatial query iis a special type of database query supported by spatial databases. Spatial query processing is is now an essential functionality of mobile sensing that gather sensor data within a specific geographic region. For example, a query wants to

know a specific place with most noise in one building. The data can be gathered by cellphones carried by people who work at different places of the building. However, its important to remove redundant data because it consume unnecessary communication energy. For instance, if the sound data of office 101 has been collected by sensing Jacks cellphone, there is no need to sense the microphone of Roses who sits next to Jack. In order to remove the redundant data for transmission, Gupta et al. [10] propose an algorithm for self-organization of smartphones to reduce energy consumption. The proposed algorithm constructs a near-optimal sensor cover with the following properties: 1) the sensing regions of the selected set of sensors cover the entire geographical region of the query, and 2) the selected set of sensors form a connected communication graph where there is an edge between any two sensors that can directly communicate with each other. Similar strategies will be explained in details in Section 7.

In sum, data transimission reduction approaches can be used for those cases that only certain portion of all the data is informative while the rest is insignificant. They are able to filter out uninformative or duplicate data while keeping the informative or indispensable data transmitted to destination. Compression techniques are able to further drop the amount of data needs to be transmitted.

4.2 Data Predication

In this section, we survey approaches using data predication techniques to reduce energy consumption. Data predication techniques establish models describing the sensed phenomenon. Therefore, queries can be answered using the model instead of the actual sensed data. There are two types of predications. One is sensor data prediction which uses previous data to predict futher sensor data. The other is to predict context information (as opposed to predciate sensor data) by using existing sensor data.

SAF [46] is a framework utilizing simple time series forecasting models to predict sensor readings. The idea is to build series local models (contains time-varying function) at each node, transmit the models to the root of the network, and use them to approximately answer user queries. Through exploiting properties of local forecasting models, communication occurs only when underlying data distribution changes because the local model varies, so that the communication is reduced.

Some mobile sensing applications want to query the user's activity or context such as whether a user is running or walking. Continuous sensing of user's activity or context depletes user smartphone's battery. Many approaches use Markov processes

to predict the sensor data, the user context or activity given current information. For example, Krause et al. [17] model user activity sequence in a continuity Markov chain, each state describes the activity of the user at each timestamp. The purpose is to predicate the user's activity by selecting a set of observation times, without losing expected accuracy. Similarly, Wang et al. [48] presents a mechanism that aims at solving state estimation by using a semi-Markov state estimation technique that selects the most likely user state while observations are missing, and a semi-Markov optimal sensing policy which minimizes the expected state estimation error while maintaining a given energy budget. Different from [17], [48] is able to generate an optimal sensing policy which constrain the energy consumption within a given budget while minimizing the expected activity estimation error. In [20], the author describe and evaluate a system that uses phone-based accelerometers to perform activity recognition and context predication, a task which involves identifying the physical activity a user is performing. More similar strategies will be explained in details in Section 5.2.1.

In general, data predication approaches are proper to be used in those cases that patterns of sensed data are obvious and easy to detect and used to predicate the coming sensed data without actually sensing it.

5 Energy-efficient data acquisition

In some mobile sensing scenarios, reducing data energy consumption by using data reducting techniques may not be enough: the number of requests for sensor data needs to be reduced as well. Data acquisition is a process of an application requesting for certain sensors data, and the sensors sending the required data to the application. Data acquisition contains two components: data request and sensor sampling. Note that by reducing the data sampled at source nodes, it is able to reduce the communication energy consumption as well.

A fixed sampling rate for sensor may result in a suboptimal energy-efficient and accuracy: because 1) different applications need differetn periodicity of sensor data; 2) different sensors have different precision and reliability under different conditions.

We will survey on two major strategies on energy-efficient data acquisition, one is adaptive sampling on single sensor data, while hierarchical sampling manages different sampling schedules on multiple sensors. For the sake of clarify, we re-plot relevant parts of the taxonomy figure in regards to energy-efficient data acquisition solutions.

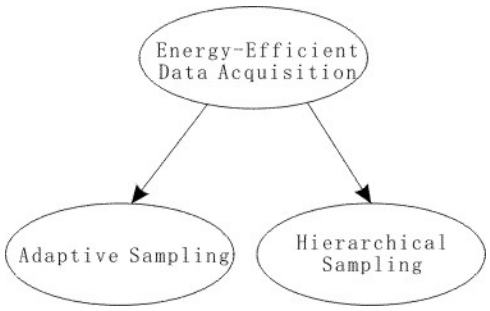


Figure 5: Approaches of data acquisition

5.1 Adaptive Sampling

Adaptive sampling is used to reduce the total number of samples by exploiting spatial or temporal correlations between sensor data. For single node or individual smartphone, adaptive sampling can automatically change the sampling rate and schedule to save energy. Several approaches applying this adaptive sampling strategy are introduced as follows.

A key challenge of sensing sensors on smartphones is to adaptively sample raw sensor data at certain robust level. Jigsaw [27] is one continuous sensing engine designed to balance the performance needs of the various applications and the energy demands of mobile sensing on the phone. Jigsaw uses a pipelined stream processing architecture that adaptively copes with the individual challenges for each sensor raw data including accelerometer, microphone and GPS. Jigsaw allows accelerometer inferences to be robust to various phone hardware, even body positions and orientation. In addition, Jigsaw uses on-demand processing and smart admission control flow to adaptively tune the pipelines when the input data is fuzzy or uninformative. Moreover, pipeline stages are triggered judiciously in order to adapt the human behavioral patterns, especially for the expensive pipeline stages. These all low energy design of sensor pipelines help smartphone to save energy at processing raw data.

There are some approaches that aim at generating a adaptive sensing plan in sleep/wake fashion to avoid unnecessary data acquisition. In [11] the authors propose adaptive data collection mechanisms for sensor environments that adjust to different sensors while at the same time optimizing the energy consumption of sensors. The authors propose four sensor models: always-active (AA), active-listening (AL), active-sleep (AS) and active-listening-sleep (ALS). It dynamically switches

from model to model according to the accuracy of approximation range of the sensor while minimizing the total energy consumption.

Little Rock [40] is a sensing architecture which uses an extra co-processor to process sensor data if possible. The designed co-processor consume less power than main processor. The smartphone can offload all the continuous sensing tasks to the co-processor. All sensors are connected to the co-processor, and the phone transits to sleep mode while co-processor is in charge of acquiring all the sensor data. The co-processor is designed to work seamlessly with main processor. Co-processor can transfer the sensor data in high speed to the main processor while the phone transits to active mode. By using this architecture, the system can benefit from using the co-processor to process continuous sensing at a low power overhead.

The current paradigm of acquiring sensor data is typically streamed (pushed) from the sensors to the smartphone. However, not all blocks of sensor data contribute to the query being processed. Therefore, an optimized solution is to adaptively acquire the appropriate blocks of data in a pull-based asynchronous model. ACQUA (Acquisition Cost-Aware Query Adaptation) [31] is one of the frameworks that is able to implement the pull-based asynchronous model for each query. A query here is a boolean combination of predicates on sensors data. By mining the historical data, ACQUA compute the probability of each predicate evaluating to FALSE or TRUE. Combining the probability with the expected energy consumption of each predicate, ACQUA is able to select the best order of predicate processing for the query. Following the order, ACQUA is able to attain short-circuiting to avoid retrieving unnecessary sensor data. In addition, ACQUA varies the range of predicate windows (evaluatino period) to maximize the buffer usage in order to minimize the energy acquisition. The algorithm will be discussed in details in section 6.2.

Similarly, CenceMe [30] is a mechanism to delay sensor data acquisition, which improves overall energy efficiency while maintaining the application's responsiveness. The delay varies according to the type of context being inferred. For example, if detecting a user's current phone state, it is acceptable to know that he is in a phone call one minute after the actual phone call began. CenceMe is capable of classifying those type of contexts so that latency is tolerated resulting in energy saving.

5.2 Hierarchical Mobile Senisng

Hierarchical mobile sensing includes two main strategies: hierarchical mobile sensing architecture and hierarchical sampling. Hierarchical mobile sensing architecture aims at designing hierarchical architecture layout including components and their

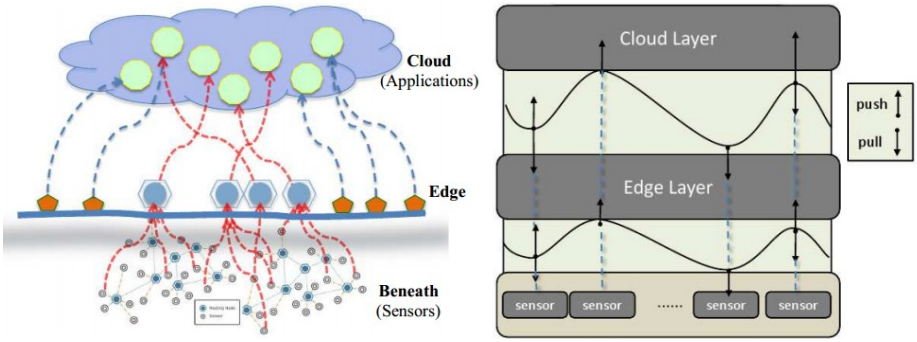


Figure 6: Cloud, edge and beneath

roles in energy-efficient fashion. Hierarchical sampling approach focuses on using smartphone equipped with different kinds of sensors or different layers of processing architecture. Different sensors can be characterized by specific performance features, most are accuracy and energy consumption. Generally simple sensors have very limited resolution. On the other hand, although complex sensors can provide more detail characterization of the sensed data at the expense of higher energy consumption. Most of the existing approaches trades off accuracy for energy efficiency at specific cases with maintaining accuracy within tolerated range or trades off energy consumption for accuracy when there are needs of greater details.

Pull and push are two modes of acquiring data from sensors. Push mode occurs when application issues subscription request on specific sensor and acquires the stream data from the sensor, while pull mode occurs when process a query. CEB (Cloud, Edge, and Beneath) [51] is a hierarchical mobile sensing architecture that tunes the pull and push mode to obtain energy saving. It initials processing of sensing occurs in- and near- network, in order to achieve system energy efficiency. The paper also introduces a concept of PPE (optimal push/pull envelope) dynamically and minimally adjusts the base push and pull rates for each sensor to avoid unnecessary data transmission. Figure 6 shows the three-layer model of sensor computing with cloud, edge and beneath. Beneath includes sensors than constantly sense environmental parameters, edge can be a mobile device or a computer controlled by an application (Cloud) designed to monitor multiple conditions or activities within a particular environment. CEB generates PPE — Adaptive and optimal sequence describes which sensors should be in pull or push mode in order to save energy usage in the sensor layer.

5.2.1 Hierarchical context/state monitoring

Hierarchical sampling, in many scenarios, consists in acquiring targets' context inferred from sensor data instead of sampling raw sensor data directly. In these scenarios, raw sensor readings need to be classified into context and then used by mobile sensing context-aware applications. For example, by comparing the patterns of certain context/state, we can classify GPS readings as walking, jogging, running or biking. In practice, usually multiple sensors involve in recognizing context/state in regard to accuracy or energy-efficient. Therefore, hierarchical context/state monitoring is spawning and attract mobile sensing applications in practice. Some existing approaches aim at addressing this issue are introduced as followed. Energy-efficient mobile sensing system (EEMSS) hierarchical sensor management strategy [49] is one of the framework which is capable to recognize user states with tracking the state transitions as well. By activating only a small set of sensors and using appropriate sensors duty cycles, EEMSS significantly improves the battery duration of smartphone. Figure. 7 shows the sequential management rules used to track state (represent in XML format) transitions when the user is walking outdoors. It illustrates the strategy to use GPS with highest priority following by WiFi, and then by sensing audio (microphone) to determine the final user context. By using the decision tree, this approach can filter out most cases that do not need to sense WiFi and audio thus reducing the total energy consumption.

Association rules is another techniques which can be used in recognizing users' context. For example, assume Jack usually has a meeting in the morning. If applicaiton detects that on Monday morning Jack is sitting at a meeting room, we can infer it is highly likely that A is having a meeting. The association rule learning is the approach for us to discover interesting relationships among various contexts. Acquisitional Context Engine(ACE) [35] is one of the frameworks that utilizes association rules for energy-efficient purposes. ACE is middleware for efficiently continuous sensing of user's context in mobile device. ACE is able to automatically learn relationships among various context attributes. It applies two strategies of optimizations to complete the dynamically learning process, inference caching and speculative sensing. It uses Apriori algorithm to implement association rule mining. The general form is $\{c_1, c_2, ..., c_n\} \rightarrow c$ where all the already known contexts $\{c_1, c_2, ...c_n$ imply context c. Proposed inference caching store the implied context as a traditional cache. If a query involves with c, ACE just simply return the implied value instead of acquiring sensors for c. Moreover, ACE is able to generate the cheaper sensing plan by using cheaper sensors instead of expensive one when both sensor can answer the query. This feature gives ACE flexibility in hierarchical

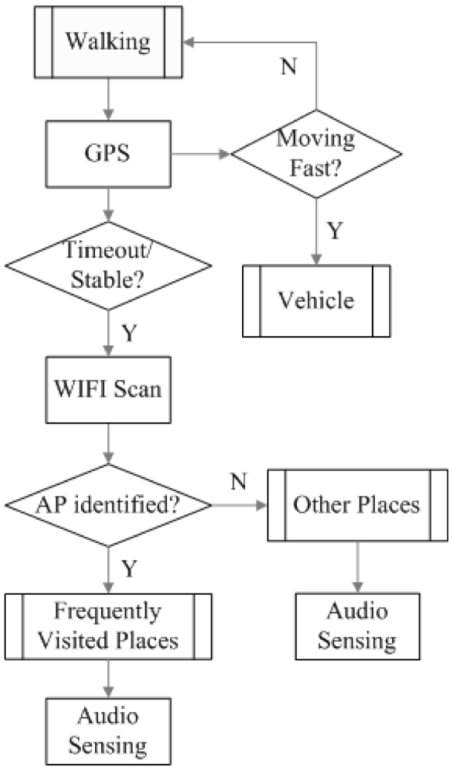


Figure 7: Sequential management rules of EEMSS

mobile sensing in an energy-efficient way.

Another energy-efficient context regonition system is SeeMon [14]. SeeMon is context monitoring approach that provides efficient processing and sensor control mechanisms. SeeMon offers multiple API for upper-level applications to acquire context monitoring. It includes efficient processing and sensor control mechanisms. It focuses on devising an efficient way to process continuous detection of context changes as well as identifies the context which is still the same avoid redundant sensing. It includes a bi-directional way instead of uni-directional way to enhance a feedback path in the pipeline and gives an opportunity to achieve a high degree of efficiency in energy consumption. This also makes it possible to elaborate the computational stages in processing pipeline and hence to make a monitoring decision at an earlier stage. SeeMon reduce the processing overhead by pruning out unnecessary context recognition at an early stage of the processing. Moreover, it proposes Essential Sensor Set (ESS) to avoid performing unnecessary sensors.

Figure 8: Positioning accuracy of different technical

Energy consumption, accuracy and latency are the three important metrics of a mobile sensing system. Kobe [7] is designed to tune the three metrics while meet the application requirements. Kobe is a tool that aids mobile classifier development. In this case, classifier is used to classify raw data into context/state. With the help of a SQL-like programming interface, Kobe helps to determine which sensors to use and what labeled training data to target. Kobe performs profiling and optimization of classifiers to achieve an optimal energy-latency-accuracy tradeoff. In [17] the author also studies the trade-off between energy consumption and prediction accuracy of context classifier. It proposes a set of selective sampling strategies that use Markov chain to model the user's activity sequence and then predicate the user's coming activity with minimized expected loss. The purpose of these strategies is to reduce the number of required sensor readings and computation cycles, from a continuous monitoring to selected time points, with keeping certain accuracy of tracking the user's current activity.

5.3 Location Sensing

Locationsensing is to attain the current location of smartphone by using positioning techniques. Location sensing is widely used to provide postioning information for location-based applications. In these location applications, GPS is often preferred over GSM/WiFi based position systems because GPS is more accurate in most cases. Figure reffig:positioning shows the accuracy of GPS is 4 times and 20 times better than WPS and GSM based positioning time to hit McDonald's, where WPS is a WiFi-based positioning system from Skyhook Wireless.

The energy consumption of GPS is well-known to be high. Keeping GPS acti-

vated continuously would drain the 1200 mAh battery on an N95 smartphone in less than 11 hours, even in the absence of any other activity. In [37] and [55], we also found that if running some traffic-monitoring location based applications (LBAs) on a smartphone, it can drain a phone battery within 2 hours. To be useful, smart strategies are needed to extend the lifetime of smartphones running location based services. Some research has been conducted to solve this problem by considering unique features of location sensing service. In this section, we discuss two existing approaches for energy-efficient location sensing. The two approaches are adaptive GPS sampling and hierarchical location sensing.

5.3.1 Adaptive GPS Sampling

Some studies [37] and [55] discover that in certain situations, turning on GPS sensing contributes little and is even unnecessary for location services. For instance, if Mike is enjoying his lunch break at Starbucks for one hour, the location based applications on his smartphone do not need to acquire GPS data continously because the location will not change during this period. In this case, GPS component should be adaptively turned off to save power until the situation changes. For another example, if Rose is driving home on a highway and there is no other turning or Rose is more likely to keep on this highway based on previous history data, a strategy is to decrease the GPS sampling rate because it will save energy without losing too much position accuracy. In crowed urban areas, the accuracy of GPS generally drops significantly. In such environment, it is pointless to keep GPS turn on all the time. On account of these cases, some approaches aim to discover when to turn off GPS or decrease GPS sampling rate to reduce energy consumption.

Rate-adaptive positioning system [37], known as RAPS, adopts a collection of techniques to adaptively determine when to turn on GPS in duty-cycle fashion. In addition, RAPS uses the location-time history of the user to estimate user velocity and adaptively turn on GPS only if the estimated uncertainty in a position exceeds the accuracy threshold. First it employs celltower-RSS blacklisting to detect whether GPS is unavailable (e.g., indoors) and avoids turning on GPS in these places. Whenever it attains a new position update, it records the current celltower ID and the received signal strength (RSS) information and associates this information with the position update. Then if it determines that it is time to turn on GPS, before activating GPS it first checks the celltower-RSS table for the historical probability of GPS availability and defers GPS activation if historical data shows that GPS is not likely to be available at that place, thus avoiding unnecessary energy

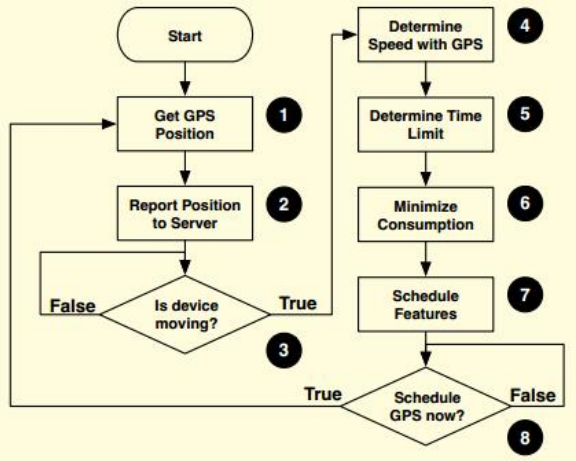


Figure 9: Logic running on Entracked Client

consumption.

Continuous position tracking is one important service provided by location based application. EnTracked [16] is a framework that estimates and predict position, and schedules position updates to both minimize energy consumption and optimize accuracy. The system is configurable to realize different trade-offs between energy consumption and robustness. EnTracked includes three components: Location-based application, EnTracked (Server) and EnTracked client (moving device). Location-based application firstly issues a request to EnTracked server for the tracking of one device with an error limit. EnTracked server propagates the request to EnTracked client which will send back a start position. Then the EnTracked Client logic schedules embedded sensors to generate next position within the given error limit. Figure. 9 shows the logic running on an EnTracked client. From the logic we find that EnTracked uses many conditional steps to attempt to filter out unnecessary GPS scheduling. At the beginning, it keeps checking if the current device is moving or stationary by using simple moving threshold algorithm on accelerometer data. Then it utilizes GPS module to estimate the device speed. Afterwards, it applies an error model to control the generated position within given error threshold. One flaw of the EnTracked system is its movement detecting algorithm. The movement detecting algorithm is too naive and causes many FALSE positive errors. For example, if a person is stationary, but gesturing with the device in hand the accelerometer will detect this movement then lead to the trigger of GPS scheduling.

5.3.2 Hierarchical Location Sensing

An alternative to use purely GPS sensor for location sensing is to use other cheap network based mechanisms (e.g. WiFi and Bluetooth) or build-in sensors (e.g. accelerometer and gyroscope) to gain location information. Some researches employ network based mechanisms and build-in sensors as a complementary way to gain location information. Hierarchical location sensing is designed to use multiple sensors instead of only GPS to gain location information. The problem is how to adjust the tradeoff between accuracy and energy-efficient of using different mechanisms to attain location information within certain error threshold. In [25] , Kaisen et al. analyze the accuracy models of GPS, Bluetooth and Cell-Tower for measuring current locations. The accuracy of GPS is in direct proportions to the number of satellites that are available, while the accuracy of WiFi depends on the number of access points. Bluetooth offers poor location information because of the limit radio range. The accuracy of Cell-Tower drops if a phone cannot decide which Cell-Tower has the strongest signal. We outline several approaches for intelligently managing the location energy and accuracy trade-offs based on available sensor capabilities of worthy attentions.

Beside tuning GPS sampling rate based on historical information, RAPS [37] also engages with other sensors (e.g. accelerometer and Bluetooth) to adjust GPS sampling cycles as well as reduce positioning uncertainty especially when GPS is less accurate. It carefully uses continuous accelerometer duty-cycle to efficiently estimate user movements and activities: being stationary, frequent walking and stopping, fast walking, driving in a car, and wandering about in a coffee shop. RAPS utilizes user movement to avoid activating GPS such as if user is stationary. It calculates the average velocity relative to the previous position, and associates this velocity and the recent activity ratio with the previous space-time coordinate. This in turn enables RAPS to estimate positioning uncertainty, allowing it to selectively activate GPS.

Current mobile platforms lack the capability of adaptively selecting the most appropriate location sensing mechanism on-the-fly to strike the balance amongst energy consumption, availability and accuracy. Zhuang et al. [55] propose a solution Sensing Substitution (SS) which select the most appropriate location sensing mechanism on-the-fly. It then performs location sensing in a more energy-efficient manner by choosing the best sensing mechanism, given the current situation. As shown in Figure. 10 , assume that the currently registered location-sensing mechanism is GPS. When the user moves into an area where network is available and its accuracy can fulfill the LBAs requirement, then the LBA uses Net to replace GPS.

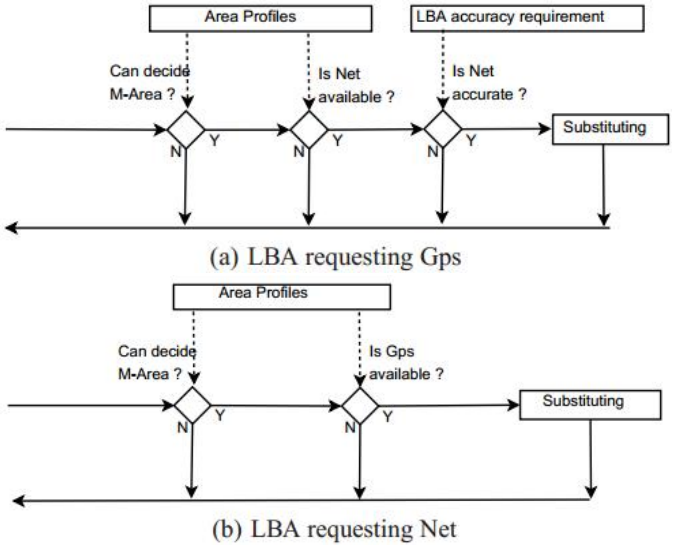


Figure 10: Sensing substitution mechanism

On the other hand, when the current location-sensing mechanism is network and the phone moves into areas where network is not working, SS invokes GPS, instead of network.

Similar to RAPS [37] , it also proposes a mechanism called Sensing suppRession (SR) to sensors such as accelerometer and orientation sensors to profile smartphones states. The difference is that it give users choice to determine the configuration at the first place. It allows the users to manually enable/disable a suppression option. Second, it applies confidence levels of the learned mobility context to reflect the familiarity with the current mobility contexts such as commuting routes.

In addition to the tradeoff for energy-accuracy, Lin et al. [25] propose a-Loc that automatically determines the dynamic accuracy requirement for mobile search-based applications and use cheap sensors to attain the user's current location. As the user moves, both the accuracy requirements and the location sensor errors change. A-Loc continually tunes the energy expenditure to meet the changing accuracy requirements using the available sensors. A-Loc uses a second order Hidden Markov Model to yield a probability distribution of location of current time step for each sensor before actually spending energy on sensing. Then it computes possible error for each sensor then puts the eligible (accuracy meet given threshold) sensors into a candidate list. The final step is to select the one with the lowest energy

consumption from the list and execute the sensing accordingly. A-Loc focuses on selecting cheapest sensor to without actually losing much accuracy. The feature benefits the location based applications that can tolerant on accuracy but has strict energy constraint.

Vtrack [45] is a system for tracking device trajectory using this sensor data that addresses two key challenges: energy consumption and sensor unreliability. The main technique is to estimate location using a variety of sensors—GPS, WiFi, and/or cellular triangulation. Similar to a-Loc, Vtrack also uses a Hidden Markov Model to model a moving objects trajectory. The difference is that Vtrack tracks trajectory over a block-level map of the area. All the objects can only follow the road segments on the map. Vtrack is able to generate the trajectory by using Viterbi [47] decoding technique to find the maximum likelihood sequence of states to establish the trajectory. Another difference from a-Loc is that Vtrack does not have an adaptive sensor selection mechanism, it only takes in any coming position samples which are available at current time.

6 Query Processing Optimization

Mobile sensing can also be thought of as query over sensor data. Moible applications specify the query. There is a query processing engine on each smartphone that processes those queries. Many researchers have been working on creating optimization architecture to support fast and energy-efficient query processing. One aspect is to create query processing optimization algorithms, the second aspect is to design query processing architecture to support advanced query processing approaches.

6.1 Query Models

We first introduce the queries that are used in some typical mobile sensing applications. Boolean expression has been broadly used to express query/sensing condition because it is simple to understand and extendable to represent complicated query. Boolean expression has two states: T for TRUE and F for FALSE. Boolean expression includes two major categories: Conjunctive normal form (CNF) and Disjunctive normal form (DNF, shown in Figure. 13). Two examples are shown as follows:

$$A \wedge (B \vee C) \wedge (A \vee B)$$
$$(A \wedge B) \vee (A \wedge C)$$

```
CONTEXT <context element>
        (AND <context element>)*
ALARM <type>
DURATION <duration>
```

Figure 11: CMQ example

Where A, B and C are predicates such as speed <10m/s or acceleration < 1m2/s. In [24] [Adaptive Data Acquisition Strategies for Energy- Efficient, Smartphone-based, Continuous Processing of Sensor Streams] all such queries are compiled into a uniform Query Tree representation, such that the root of the query tree represents the entire set of concurrent query predicates, an internal node is associated with a Boolean conjunction or disjunction operator, and a leaf node is associated with a predicate.

There are also other formats of queries, in [14] the author uses xml to represent query. For example, Context Monitoring Query (CMQ) is an intuitive monitoring query language that supports rich semantics for monitoring a wide range of contexts. The template of CMQ is shown in the Figure. 11. Where CONTEXT describes the context of interest and represented in a CNF query. ALARM supports two transition states: T $\rightarrow$ F and F $\rightarrow$T, where T $\rightarrow$ F denotes from True to FALSE and F $\rightarrow$ T denotes FALSE to True. DURATION specifies how long a registered CMQ should run.

Kalpakis et al. [13] model a query with DAG (a rooted expression directed acyclic graph). Internal nodes of DAG are operators or functions while children as their operands. Its leaves are constants or variables. Each vertex of DAG has a size for its value and a collection of candidate network nodes to which it may be placed. Each variable vertex has a set of source sensor nodes, whose measurements are used to assign values to that variable. An example of DAG is shown in Figure. 12

Ellipses denote operators, and shaded/not shaded squares denote variables/constants. Each vertex is tagged with candidates, unless it contains all network nodes such as vertex B and E. In addition, operators are tagged by their associated function, and variables their data source sets ({candidates)}/{source}). Each edge u$\rightarrow$v is tagged with the size of us value.

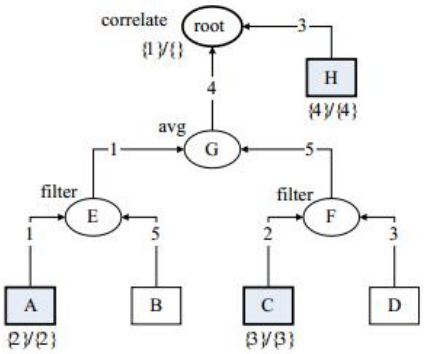
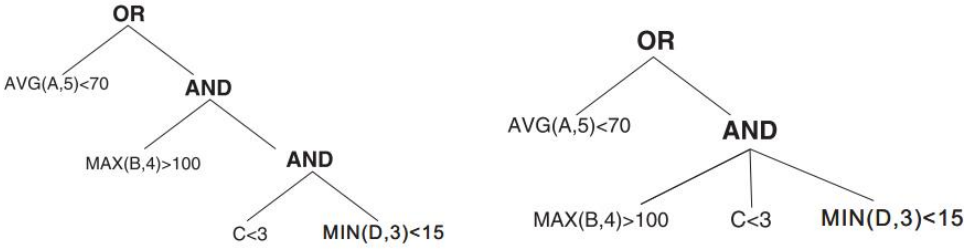


Figure 12: DAG example



(a) Binary query tree representation (b) DNF query tree representation

Figure 13: DNF example

6.2 Query Processing Optimization

When processing a query, the Naive method is to acquire all the sensor data relevant to the query. Then evaluate all the predicates in the query based on the sensor data and get the final result for the query expression. However, we observe that in most cases we do not need to acquire all the sensor data to process a query. For example, if an LBA needs to determine whether a user is at a library, it processes a query $GPS = library \vee WiFi = libraryfreewifi$. GPS and WiFi has different energy consumption ratio. Obviously WiFi is cheaper than GPS, so the best strategy is to evaluate WiFi first. If the query can be answered by WiFi then it does not need to bother activating GPS. Therefore, it saves the energy acquired for evaluating GPS data. In general, there can be an exponential number of strategies according to the number of predicates. Each strategy returns the correct answer, but which vary in terms of their expected costs. The task of finding the strategy with the minimum expected energy consumption is referred to as PAOTR problem (Probabilistic And-

Or Tree Resolution) [9]. For and-or tree query with co-related predicates, the problem is NP-hard problem. Some [heroustic] solutions has been conducted to solve this problem.

In order to find the optimal strategy, ACQUA [31] proposes ASRS sequential retrieval algorithm. ACQUA does not acquire all the sensor data of a binary tree structure query but sequentially acquire the data which has the highest probability to determine the result of the query. The purpose of the algorithm is to determine the truth value of the query as early as possible. In order to find the evaluation strategy/order/sequence, ACQUA computes the NAC (normalized acquisition cost), a ratio of the acquisition cost normalized by the probability of predicate being false. For each node in binary query tree, it compares two NAC of left sub-query and right sub-query, and then chooses the sub-query with the less NAC to be evaluated first. ACQUA recursively computes the order for each node and generates a final evaluation order. Then it acquires the sensor data one by one following the evaluation order.

As mentioned in section 6.1, the binary tree query structure has less possible orders than DNF query structure. Therefore, Lim et al. [24] further propose to convert binary query to DNF structure in order to increase the number of possible orders. More possible orders means more chances to find the best orders. Moreover, instead of static ordering, it proposes a dynamic ordering algorithm which takes into account the fact that optimal ordering varies dynamically over time due to changes in sensor data. During the evaluation period, acquisition cost varies according to the acquired sensor data. Therefore, it adaptively changes the acquisition cost to get better strategy. Third, different from ACQUA, it does not compute the acquisition cost of a predicate, instead it computes the acquisition cost based on sensor data streams. Therefore, the evaluation order is based on sensor data not predicates. How to sort the sensor order follows two criteria:

- Sensor data that enable more predicates in the query to be evaluated should be ranked higher
- Sensor data that enable higher probability of short-circuiting the evaluation should be ranked higher

Therefore, if one sensor data stream meets these two criteria, the sensor data should be in the front the evaluation ordering.

Wireless sensor networks (WSN) also earns lots of attention recently. The WSN is collection of sensor nodes from a few to several hundreds or even thousands,

where each node is connected to one (or sometimes several) sensors. WSN is similar to mobile sensing scenario because they both contain sensors on each nodes/phones, and the applications want to process queries among sensor data of nodes/phones, and there needs to establish communication among nodes/phones. Therefore, query processing approaches on WSN can also be adpatted to mobile sensing.

In WSN, query processing on a large number of nodes also needs optimization to consider energy-efficient problems. Energy-efficient dynamic routing tree (EDRT) is an algorithm designed by [15] [Energy-Efficient Dynamic Query Routing Tree Algorithm for Wireless Sensor Networks] to maximize in-network processing opportunities by utilizing parent nodes and sibling nodes. [INTRUODUCE THE NODES] Enlarging in-network processing leads to the reduction of the number of message transmission by partially aggregating results of an aggregate query in intermediate nodes, or merge the results in one message. The algorithm models sensor networks as an undirected graph, and divides all nodes in WSN into different levels (relative to root/beginning node). Nodes far (how many jumps to root) from the root node are on low levels while nodes near the root node are high levels. First the algorithm determines the parent candidates and sibling candidates for each node. Second when query arrives it constructs EDRT by calculating the minimum distance and chooses the neighbor nodes with the smallest minimum distance to be the query message target, sometime the query message is integrated with partial results of current node and previous processed nodes. [41] uses the similar strategies to minimize the communication in-networking processing, in addition, they adopt *runlength* [50] compression algorithm to compress its data before sending it back to the access point (query start point).

Top-k query is one of the fundamental operators in many WSN applications. chen et al. [6] present an algorithm to optimize in-network processing when process top-k query especially for environmental monitoring. To achieve energy-efficient evaluation of top-k query, it applies a scalable, filter-based localized evaluation algorithm which is able to eliminate as many unlikely results as possible within the network transmission among nodes. Briefly, it uses a bottom-up and iteration fashion to filter out unlikely results. Each node (no leaf) has multiple children nodes each associated with a sorted decreasing list of top-k results acquired from its children. Then each node sends its α-quantile value (the border between possible top-k value and impossible top-k value) to its parent. Its parent choose one fo the α-quantile values for broadcasting to its children. Finally the children send all the values no less than the broadcasting value back to parent. Therefore, the sensor network eliminates unlikely top-k results level by level.

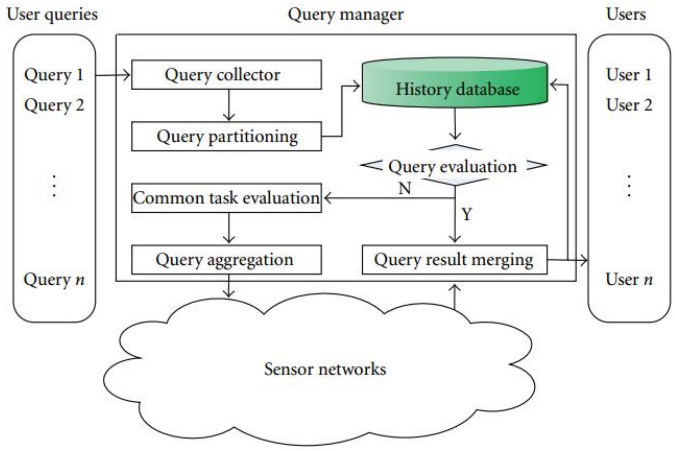


Figure 14: Scheme for finding common sub-queries

Besides optimizing the data aggregation and in-network processing, query management to optimize the query aggregation plan at the gateway side is another considerable approach to reduce energy consumption. In [34] the author proposes a multi-query management framework aims at finding common sub-queries and execute them only once to reduce redundant task executions.

The main components of the framework are shown in the Figure. 14. The workflow can be briefly described as follows: 1. First partition query into sub-queries according to query partitioning scheme which based on query region, query time, and query attributes; 2 then it looks up the history database to find if there is any or partial query records that can be used to process the query. 3. For those query are not done by using history database, they are sent to appropriate regions to achieve the query result. 4. If the query result is accurate and reliable enough, they can be stored in the history database for future usage.

7 Collaborative Sensing

7.1 Sensing Sharing

Recent research works have begun to look at the sharing of sensing tasks among a group of people, especially if those people have daily interaction. Another case is that mobile devices can share sensing data if they are close enough to share expensive data by cheap communication interfaces. There are three different types of sharing,

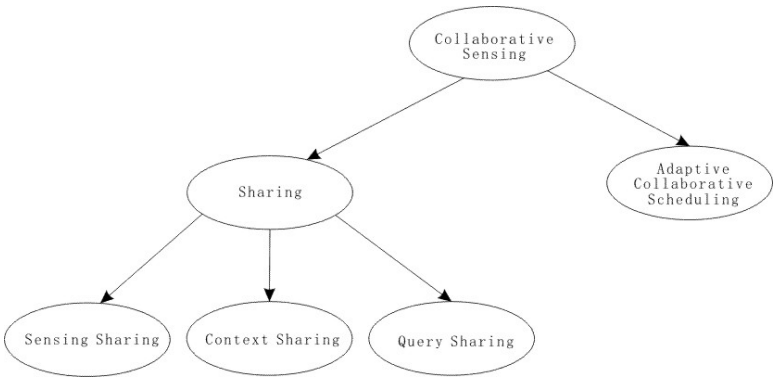


Figure 15: Collaborative approaches

one is raw sensor data sharing, one is share personal activity or ambient context, and one is sharing query results or partial query results. We re-plot the relevant part of the taxonomy figure in order to be clarify. (Figure. 15).

7.1.1 Data Sharing

When similar applications are run on a number of phones which are neighboring to each other, then the sensor data can potentially be shared among those mobile devices in order to avoid acquisition of data from multiple sensors data.

In RAPS, Bluetooth communication is used to reduce position uncertainty among neighboring devices. Whenever it receives a new position update, it broadcasts this position to Bluetooth peers so that they can update their position without activating GPS themselves if they believe the provider is trustworthy. In other words, whenever a device obtains a position update from a peer that has greater uncertainty than its own estimation, it replies with its more accurate information. Eventually, this kind of data sharing enables all devices in the neighborhood to synchronize their position information within a certain accuracy range and does avoid unnecessary power usage.

In [38] the authors propose a collaborative data reduction approach to remove the redundancy existing in the sensed data acquired from multiple sensors. It uses a tree-based data propagation model to characterize the collaboration structure among multiple sensors. Then it applies a collaboration phase to detect 1). the local data redundancy and 2). between-sensor redundancy. Therefore, by wiping out the redundancy, the energy efficiency of the system will be increased accordingly.

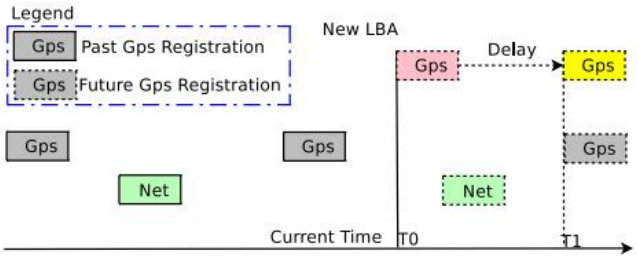


Figure 16: Piggybacking workflow

The previous two approaches enable data sharing between phones. It is also possible to share data among applications on single phone to prevent redundant sensor data acquisition. Zhuang et al. [55] propose Sensing Piggybacking (SP) which can re-use the existing sensing registrations by piggy-backing new sensing requests on existing ones, thus eliminating some location-sensing invocations. As shown in the Figure. 16, the joining/new LBA requests GPS, and the new registration is delayed to piggyback on other GPS registrations. This method saving energy consumption analogous to decrease the frequent/sampling rate of GPS. Although this techiques can save energy, if the LBA has the real-time constrain can not be applied for that LBA.

7.1.2 Context Sharing

Many mobile applicaitons now try to infer people's behavior and their ambient environment. In collaborative scenarios, contexts sharing focuses on sharing personal activities context or ambient information contexts rather than just raw data.

Darwin [29] system is a distributed system for mobile sensing that combines collaborative inference sensing and classification techniques to reason about human behavior and context on smartphones. There are three main steps implemented in Darwin system, evolution, pooling and collaborative inference 17. The evolution or classifier evolution step aims at automatically updating user models over time so that the classifiers can be robust to the variability in sensing conditions. In the pooling step, smartphones exchange classification models whenever the model is already available from another phone. In the collaborative inference step, Darwins applies machine learning techniques to combine the classification results from multiple phones to obtain better and relatively more robust inference with higher confidence.

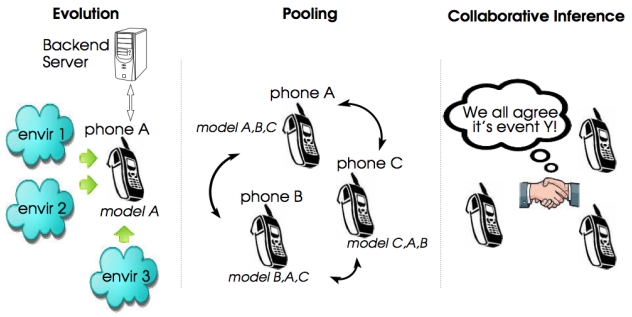


Figure 17: Evolution, pooling and collaborative in darwin system

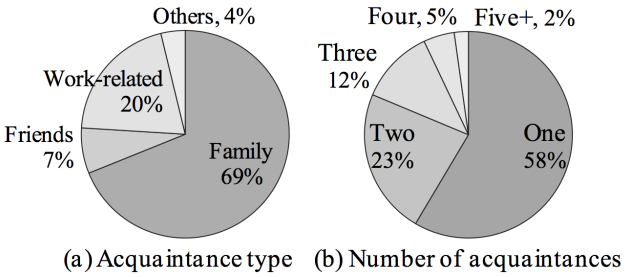


Figure 18: Acquaintance in daily life

Opportunistic cooperation is another important concept in collaborative state sharing approaches. Figure. 18 demonstrates the many opportunities for cooperation among nearby smartphones by analyzing the ATUS (The American Time Use Survey) dataset [1]. It shows that 42% of the time, and individual is with more than one acquaintance, thus there are more chances of cooperation. CoMon [23] is a cooperative ambience monitoring platform, which reduces the energy consumption via opportunistic cooperation among nearby mobile users. First, it employs a heuristics and bluetooth-based cooperator detection method to detect potential cooperators who remain in a certain range for a relative long period of time. Then CoMon automatically generates a cooperation plan that would provide benefits for the collabrative devices.

7.1.3 Query Sharing

Query sharing processing is a technique for sharing the resources used to process multiple concurrent queries by discovering the similarities among these queries. This

technique is in contrast to the traditional model where each query is processed separately in an isolated fashion. Query sharing usually automatically identifies the shareable parts of multiple executing queries. The shared parts are executed once instead of multiple times, and that eliminates repetitive execution and data transmissions. In addition, query sharing enables systems to extend the workloads with large numbers of concurrent queries, as well as with large number of mobile devices [19].

Some research have been conducted into discovering the chances of sharing the sub-query among all queries being processed. CQP [52] is a collaborative query processing framework that exploits the overlap (in both the sensor sources and the query predicates) across multiple smartphones. The overlap parts of sensor sources or query predicates are only executed once instead of [redundently] processed on individual smartphones. In CQP framework, a leader takes the responsibility of processing the common parts then dispatches the intermedia results and partial sensor data to its "member" phones. The detailed techniques are explained in the next section.

Another approach that exploits similarities in the queries and sharing is shown in [18]. It proposes several ways to efficiently share the process of aggregate queries in data streaming system with differing periodic windows and arbitrary selection predicates. A key difference from other approaches is that their approach support queries joining and leaving the system at any time. The main idea is to slice time It first applies shared time slice (STS) to slice stream data with non-overlapping paired windows. The slices can be combined with partial aggregates or other paired slices, which is used to be aggregated to answer each query in turn. They use predicates to partition the slice into disjoint subsets call $fragments$. Then the slices in each fragment can be aggregated to form partial fragment aggregates which can be processed in turn to produce the; query results. Afterwards, possible shared fragments can be found and used by a dynamic implementation to reduce the redundancy processing operations.

DEAMON (Distributed Energy-Aware Monitoring) [43] is an energy-efficient distributed algorithm for long-term sensor monitoring. DEAMON is designed to support both CNF and DNF queries. In the DEAMON algorithm, if some of the required sensor data are missing, then the node/device needs to obtain sensor data from outboard sensor nodes/devices or from neighboring helpers. Even if it has all the required sensor data, it can choose to use some of the required sensor data from neighboring nodes to improve data quality or to save the energy at local device. The data structure is shown in the Figure. 19. From top to bottom, each level repre-

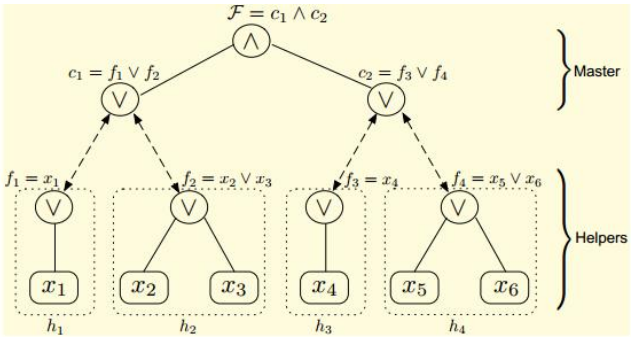


Figure 19: Data structure of DEAMON

sents query, sub-queries, and predicates. DEAMON breaks the query into a set of sub-clause/sub-query according to the sensor distributions. Then it applies a greedy algorithm to assign each sub-clause/sub-query to a specific helper in order to minimize the energy consumption. The upper part of the data structure is maintained by the master node while the lower part is broken into sub-queries and assigned to helpers. The dashed lines between sub-clause and clause denote wireless communication between the master and the helpers. However , DEAMON only optimizes the local nodes but does not consider the helpers' resources. In a realistic scenario, if the helpers cannot benefit from the collaboration, they will not share the sensor data. It is important to design a mechanism which fosters the collaboration among nodes/devices.

Another idea for maximizing the sharing query processing among phones is to cluster similar queries together. More similarities among a set of queries implies more chances to share the sensor data or partial sub-query result. The database query processing field has studied this problem for database queries. [53] [Efficient Exploitation of Similar Subexpressions for Query Processing] shows that queries, especially for the complex queries, often contain common or similar subexpressions. Therefore query execution efficiency can be improved by evaluating a common subexpression once and reusing the result in multiple times for many queries. This paper proposes a light-weight architecture to detect potential sharing opportunities among expressions/queries.

The Figure. **??** shows the overall architecture and the key component is covering subexpression (CSE) manager. At step 1, it computes the table signature of expression and registers it in CSE manager. Each expression has one table signature, a simple abstract of the expression, which contains potentially sharable SPJG

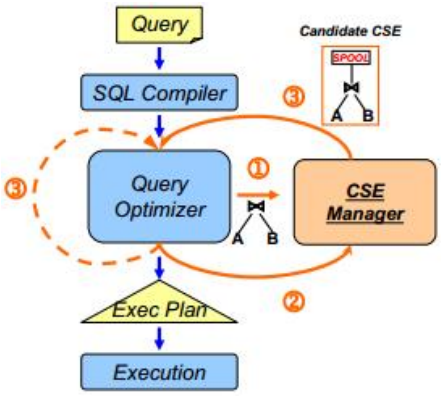


Figure 20: Architecture of CSE

expression. [Stacked indexed views in microsoft sql server]. At step 2, CSE manager checks its registered table signatures which reference two or more expressions, and the expressions are the potentially sharable expressions. Then it constructs candidate CSE for each set of potentially sharable expressions. At step 3, the optimizer optimizes the query with the sets of candidates. This architecture avoids redundantly evaluating similar subexpresssions from different queries so it can save time and energy consumption.

7.2 Collaborative Scheduling

Different from sensing scheduling on single phone, collaborative scheduling requires multiple phones or even cloud server to involve with. Collaborative scheduling adaptively modifying the sensing and communication scheduling among collaborative groups of phones can achieve considerable energy saving.

MECSS (Minimum Energy Collaborative Sensing Scheduling) problem is proposed to seek a sensing schedule for each user to minimize the energy consumption subject to the constraint that that given region is totally covered by the selected users mobile phone sensing range before a given deadline. In order to solve this type of problem, Sheng et al. [42] introduce a polynomial-time algorithm to get optimal solutions. The solutions achieve energy saving by using collaborative sensing in smartphone sensing applications with keeping fairness in mind.

Mobile sensing query are usually used to detect events. One concept of energy-aware approaches is to detect the event as early as possible so that undesired sensing

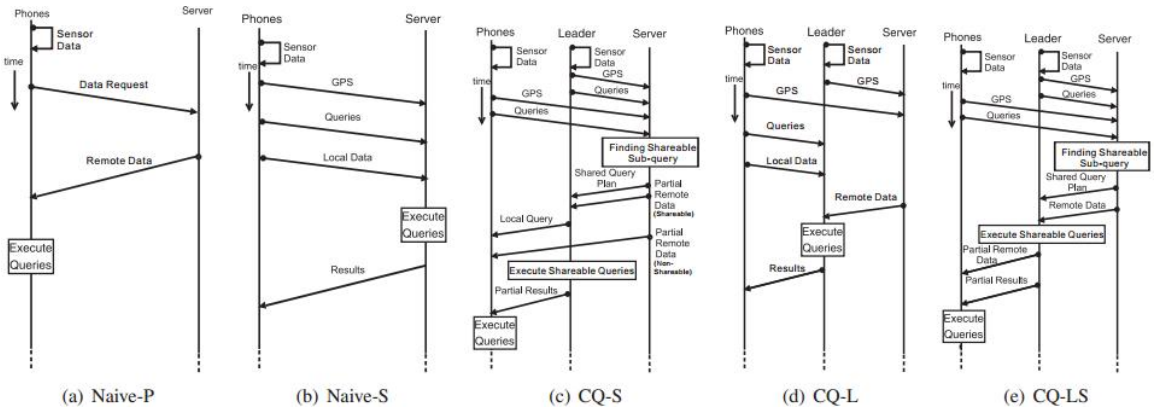


Figure 21: 5 methods of CQP work flow

operations can be filtered out. ICEQ [54] adopts a collaborative way for determining events at an early stage. It chooses range-nearest neighbors as the basic components of surrounding nodes. Then it identifies the approximate boundary between the surrounding candidate nodes and tries to pick the nearest neighbor collaborative nodes for enclosing the event in the node selection phase, which can avoid redundant sensor nodes to join surrounding nodes via identifying a collection of association surrounding nodes between the nearest surrounding sensor nodes and the query events.

In [52] , the authors propose a query processing framework CQP (collaborative query processing). CQP identifies the sharable parts of multiple executing queries, and then assigns a set of leader mobile nodes to execute and disseminate these sharable partial results. There are three main roles: 1. Group leader (one) receive, send and process data on behalf of member devices in a group; 2. Member devices (multiple) belong to a group and register its queries on local data or remote data on a server; 3. Server (one) manages a collection of groups, optimizes and coordinates the collaborative processing of groups. Also, without loss of generality, server can be treated as proxy of any remote data.

Figure. 21 above describe 5 methods work flow. For Naive-P, all devices request remote data from server and process the queries locally. For Naive-S, different from Naive-P, all procedures are done on server. Devices send all local sensor data and queries to server, after processing the queries server sends back the results respectively. CQ-S, CQ-L and CQ-LS are all involved with collaborative processing features. For CQ-S, server generates CQEP (collaborative query execution plans) in-

cludes respective responsibility of processing queries. Each device has its own CQEP fragment to execute. Beside personal CQEP fragment, leader has extra responsibility of processing sharable queries on behalf of its member devices. After processing the sharable queries, leader dispatches the partial result of sharable queries to its member devices. Then all member devices execute the un-sharable part of queries combine the partial result from leader to get the final results. Different from CQ-S, leader not only takes charge of dispatching partial result, it also acquires local data from the member device and then all queries are executed on leaders. For CQ-LS, remote data is only transmitted once from server to leader. Similar to CQ-LS, leader only processes the sharable queries and then member devices process query after receiving partial result of sharable queries from their leader. These three strategies perform differently on different situation (ratio that average local data vs. remote data).

Continuous query processing is also a fundamental feature of mobile sensing applications. BOSe* is a energy-aware energy operator placement algorithm introduced by [36]. BOSe* is able to determine which part of a CQ (continous queries) plan should be executed at the data stream management system while which part should be executed at the mobile devices. Through executing appropriate parts respectively on the two different ends to reduce the total energy consumption.

8 Conclusion

This review surveys the main techniques used for energy-efficient mobile sensing. We have paid special attention to demonstrating the comprehensive taxonomy of the state of the art approaches, such as data reduction, query processing optimization, data acquisition and collaborative sensing, of the reviewed literature. In section 4, we discussed data reduction approaches that reduce the amount of the data sensed or transmitted by using filter mechanism and data predication techniques. In section 6, we discussed query processing optimization approaches that reduce the sensing energy consumption by selecting the appropriate alternative sensors and managing the sensing schedule to optimize the query plan. In section 5we discussed data acquisition approaches that reduce the number of acquisition from application point of view. In section 7, we discussed collaborative sensing approaches that reduce the redundant processes among a large group of smartphones by using sensing sharing and collaborative scheduling. We compare existing methods by systematically discussing their advantages and disadvantages in different scenarios. Moreover, We

compared and contrasted different approaches in order to understand the big picture of current research focus in this field.

As social applications become increasingly adopted on mobile platform, it may be more useful to consider a conmunity consisting of a large number of smartphones rather than individual phones. Managing the relationship between phone to phone or phone to base station will be critical to mobile sensing energy-efficiency. Hence, we discuss the energy-efficiency issues by looking at different mobile applications in order to understand the general power consumption problems that occur in large communities.

To summarize, in this paper we consider different approaches to reduce the energy consumption of mobile sensing for mobile applications. In many practical appliations, especially the applications involved with large numbers of users, collaborative mobile sensing seems to gain even more attention within the research community in the furture.

Reference

[1] American time use survey.

[2] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. In *Mobile Data Management*, pages 3–14. Springer, 2001.

[3] Ying Cai and Toby Xu. Design, analysis, and implementation of a large-scale real-time location-based information sharing system. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 106–117. ACM, 2008.

[4] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, pages 21–21, 2010.

[5] Georgios Chatzimilioudis, Andreas Konstantinidis, Christos Laoudias, and Demetrios Zeinalipour-Yazti. Crowdsourcing with smartphones. 2012.

[6] Baichen Chen, Weifa Liang, Rui Zhou, and Jeffrey Xu Yu. Energy-efficient top-k query processing in wireless sensor networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 329–338. ACM, 2010.

[7] David Chu, Nicholas D Lane, Ted Tsung-Te Lai, Cong Pang, Xiangying Meng, Qing Guo, Fan Li, and Feng Zhao. Balancing energy, latency and accuracy for mobile sensor data classification. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 54–67. ACM, 2011.

[8] Thong T Do, Lu Gan, Nam H Nguyen, and Trac D Tran. Fast and efficient compressive sensing using structurally random matrices. *Signal Processing, IEEE Transactions on*, 60(1):139–154, 2012.

[9] Russell Greiner, Ryan Hayward, Magdalena Jankowska, and Michael Molloy. Finding optimal satisficing strategies for and-or trees. *Artificial Intelligence*, 170(1):19–58, 2006.

[10] Himanshu Gupta, Zongheng Zhou, Samir R Das, and Quinyi Gu. Connected sensor cover: self-organization of sensor networks for efficient query execution. *Networking, IEEE/ACM Transactions on*, 14(1):55–67, 2006.

[11] Qi Han, Sharad Mehrotra, and Nalini Venkatasubramanian. Energy efficient data collection in distributed sensor environments. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 590–597. IEEE, 2004.

[12] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2000.

[13] Konstantinos Kalpakis and Shilang Tang. Maximum lifetime continuous query processing in wireless sensor networks. *Ad Hoc Networks*, 8(7):723–741, 2010.

[14] Seungwoo Kang, Jinwon Lee, Hyukjae Jang, Hyonik Lee, Youngki Lee, Souneil Park, Taiwoo Park, and Junehwa 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*, pages 267–280. ACM, 2008.

[15] Si Gwan Kim and Hyong Soon Park. Energy-efficient dynamic query routing tree algorithm for wireless sensor networks. *Energy*, 3(2), 2012.

[16] Mikkel Baun Kjærgaard, Jakob Langdal, Torben Godsk, and Thomas Toftkjær. Entracked: energy-efficient robust position tracking for mobile devices. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 221–234. ACM, 2009.

[17] Andreas Krause, Matthias Ihmig, Edward Rankin, Derek Leong, Smriti Gupta, Daniel Siewiorek, Asim Smailagic, Michael Deisher, and Uttam Sengupta. Trading off prediction accuracy and power consumption for context-aware wearable computing. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 20–26. IEEE, 2005.

[18] Sailesh Krishnamurthy, Chung Wu, and Michael Franklin. On-the-fly sharing for streamed aggregation. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 623–634. ACM, 2006.

[19] Saileshwar Krishnamurthy. *Shared query processing in data streaming systems*. PhD thesis, Citeseer, 2006.

[20] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 12(2):74–82, 2011.

[21] Nicholas D Lane, Shane B Eisenman, Mirco Musolesi, Emiliano Miluzzo, and Andrew T Campbell. Urban sensing systems: opportunistic or participatory? In *Proceedings of the 9th workshop on Mobile computing systems and applications*, pages 11–16. ACM, 2008.

[22] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[23] Youngki Lee, Younghyun Ju, Chulhong Min, Seungwoo Kang, Inseok Hwang, and Junehwa Song. Comon: cooperative ambience monitoring platform with continuity and benefit awareness. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 43–56. ACM, 2012.

[24] Lipyeow Lim, Archan Misra, and Tianli Mo. Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams. *Distributed and Parallel Databases*, pages 1–31, 2013.

[25] Kaisen Lin, Aman Kansal, Dimitrios Lymberopoulos, and Feng Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 285–298. ACM, 2010.

[26] Xi Liu, Osvaldo Simeone, and Elza Erkip. Energy-efficient sensing and communication of parallel gaussian sources. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 1341–1345. IEEE, 2012.

[27] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 71–84. ACM, 2010.

[28] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on database systems (TODS)*, 30(1):122–173, 2005.

[29] Emiliano Miluzzo, Cory T Cornelius, Ashwin Ramaswamy, Tanzeem Choudhury, Zhigang Liu, and Andrew T 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*, pages 5–20. ACM, 2010.

[30] Emiliano Miluzzo, Nicholas D Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B Eisenman, Xiao Zheng, and Andrew 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*, pages 337–350. ACM, 2008.

[31] Archan Misra and Lipyeow Lim. Optimizing sensor data acquisition for energy-efficient smartphone-based continuous event processing. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 88–97. IEEE, 2011.

[32] Iqbal Mohomed, Archan Misra, Maria Ebling, and William Jerome. Context-aware and personalized event filtering for low-overhead continuous remote health monitoring. In *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, pages 1–8. IEEE, 2008.

[33] Beipeng Mu, Girish Chowdhary, and Jonathan P How. Efficient distributed sensing using adaptive censoring-based inference. 2013.

[34] Guofang Nan and Minqiang Li. Energy-efficient query management scheme for a wireless sensor database system. *EURASIP Journal on Wireless Communications and Networking*, 2010:19, 2010.

[35] Suman Nath. Ace: exploiting correlation for energy-efficient and continuous context sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 29–42. ACM, 2012.

[36] Panayiotis Neophytou, Jesse Szwedko, Mohamed A Sharaf, Panos K Chrysanthis, and Alexandros Labrinidis. Optimizing the energy consumption of continuous query processing with mobile clients. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 98–103. IEEE, 2011.

[37] Jeongyeup Paek, Joongheon Kim, and Ramesh Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 299–314. ACM, 2010.

[38] Chiwoo Park, Yu Ding, and Eunshin Byon. Collaborative data reduction for energy efficient sensor networks. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 442–447. IEEE, 2008.

[39] Richard Pon, Maxim A Batalin, Victor Chen, Aman Kansal, Duo Liu, Mohammad Rahimi, Lisa Shirachi, Arun Somasundra, Yan Yu, Mark Hansen, et al. Coordinated static and mobile sensing for environmental monitoring. In *Distributed Computing in Sensor Systems*, pages 403–405. Springer, 2005.

[40] Bodhi Priyantha, Dimitrios Lymberopoulos, and Jie Liu. Littlerock: Enabling energy-efficient continuous sensing on mobile phones. *Pervasive Computing, IEEE*, 10(2):12–15, 2011.

[41] Ross Rosemark, Wang-Chien Lee, and Bhuvan Urgaonkar. Optimizing energy-efficient query processing in wireless sensor networks. In *Mobile Data Management, 2007 International Conference on*, pages 24–29. IEEE, 2007.

[42] Xiang Sheng, Jian Tang, and Weiyi Zhang. Energy-efficient collaborative sensing with mobile phones. In *INFOCOM, 2012 Proceedings IEEE*, pages 1916–1924. IEEE, 2012.

[43] Minho Shin, Patrick Tsang, David Kotz, and Cory Cornelius. Deamon: Energy-efficient sensor monitoring. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on*, pages 1–9. IEEE, 2009.

[44] Asim Smailagic and Daniel Siewiorek. Application design for wearable and context-aware computers. *Pervasive Computing, IEEE*, 1(4):20–29, 2002.

[45] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

[46] Daniela Tulone and Samuel Madden. An energy-efficient querying framework in sensor networks for detecting node similarities. In *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 191–300. ACM, 2006.

[47] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.

[48] Yi Wang, Bhaskar Krishnamachari, and Murali Annavaram. Semi-markov state estimation and policy optimization for energy efficient mobile sensing. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, pages 533–541. IEEE, 2012.

[49] Yi Wang, Jialiu Lin, Murali Annavaram, Quinn A Jacobson, Jason Hong, Bhaskar Krishnamachari, and Norman Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 179–192. ACM, 2009.

[50] Wikipedia. Run length encoding.

[51] Yi Xu, Sumi Helal, and Mark Scmalz. Optimizing push/pull envelopes for energy-efficient cloud-sensor systems. In *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 17–26. ACM, 2011.

[52] Jin Yang, Tianli Mo, Lipyeow Lim, Kai-Uwe Sattler, and Archan Misra. Energy-efficient collaborative query processing framework for mobile sensing services. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 147–156. IEEE, 2013.

[53] Jingren Zhou, Per-Ake Larson, Johann-Christoph Freytag, and Wolfgang Lehner. Efficient exploitation of similar subexpressions for query processing. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 533–544. ACM, 2007.

[54] Rongbo Zhu. Intelligent collaborative event query algorithm in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2012, 2011.

[55] Zhenyun Zhuang, Kyu-Han Kim, and Jatinder Pal Singh. Improving energy efficiency of location sensing on smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 315–330. ACM, 2010.