Public Sensing – Using Your Mobile Phone for Crowd Sourcing

Kurt Rothermel, Frank Dürr, Stuttgart

ABSTRACT

Public sensing is a new concept for gathering sensor data by utilizing ubiquitously available mobile devices such as smartphones. By leveraging the crowd of mobile devices, we can build a mobile sensor network with excellent coverage in urban areas without large up-front investments for building a dedicated sensor network.

In this paper, we present several concepts and methods to improve the usability and efficiency of public sensing systems. In detail, we present a virtual sensor abstraction allowing applications to query for sensor values without having to know the population of mobile devices. Moreover, we show how model-driven sensing can improve both the quality of sensing as well as the energy-efficiency of mobile devices participating in public sensing. Finally, we present a method for optimizing the communication of sensor values gathered by mobile devices by exploiting the characteristics of mobile communication technologies.

1. INTRODUCTION

Modern mobile phones – so called smartphones – have developed into powerful devices: They are equipped with powerful multi-core processors to perform complex computational tasks. They feature gigabytes of main memory and can store tens of gigabyte on flash memory. They can communicate virtually anywhere and anytime at high speed through wireless communication technologies like LTE or WiFi. Moreover, they integrate various sensors like GPS, accelerometers, cameras, microphones. Even more sensors, such as those integrated into a mobile fitness tracker, can be connected through short range communication technologies like Bluetooth to access the user's heart rate, stress level, or ambient temperature. These technology trends effectively turn smartphones – and also other "smart" mobile devices like smart glasses – into powerful mobile sensor platforms and enable a new paradigm called public sensing, which sometimes is also called participatory sensing or mobile phone sensing.

The basic idea of *public sensing* is to leverage the crowd of mobile devices to gather sensor data, which in turn can be used for various applications like building maps, gathering environmental data like air pollution or noise level, or serve as basis for smart transport services, just to name a few. This idea is attractive for two reasons. Firstly, mobile devices are ubiquitous, in particular in urban areas, and hence, billions of sensor nodes can be used if the owners of these devices participate in sensing. Since these devices are carried by people, they naturally capture data at places which are typically most interesting for human users, including outdoor as well as indoor locations. Secondly, no investment for building a large sensor network infrastructure is required since the already existing devices shared by the crowd of users can be utilized. This enables small companies or non-commercial communities like the Open Street Map community to build new applications and services exploiting sensor data, literally over night by just offering an "app" in a mobile app store.

However, in order implement a public sensing system, three major problems have to be solved. Firstly, we have to make sure that gathered data has a certain *quality*. Typically, crowd-sensed data might not be as reliable regarding precision and accuracy as data gathered by a set of dedicated sensors with well-known specification, installed at strategically selected places, and operated by a trustworthy entity. Thus, in crowd sourcing, typically a set of redundant values are needed to increase precision and accuracy. However, as explained next, we should also not sense excessively in order to

preserve precious energy resources on mobile devices. Secondly, we need to ensure *user acceptance*, so they are willing to gather sensor data with their private device. There are basically two major reasons why users might deny participating in public sensing. The sensing application might have negative effects on their own usage of the mobile device. In particular, we have to be concerned about the *energy* spent for sensing since energy is still a very limited resource on smartphones – and even more on devices with smaller batteries like smart glasses. If participation in public sensing reduces the re-charging cycle to a period smaller than one day, users might face an empty battery during daily operation while making phone calls or browsing the Internet and, thus, turn their backs on public sensing. The second reason that might lower user acceptance is user *privacy*. Sensor data might carry privacy sensitive information. In particular, location information is one of the most privacy sensitive information since it has been shown that it can be used to identify users [MHV+13] and even reveal their interests or religious/political inclinations. Finally, we have to ensure the *usability* of the public sensing system implemented by a huge population of mobile devices by offering the right abstractions to the application (APIs). Typically, applications want to query information at certain locations rather than from certain devices. Thus, we need to offer an API to query sensor values that is transparent with regard to the population and mobility of mobile devices.

Consequently, in order to make public sensing a success, we need methods to make the sensing process *quality-aware*, *energy-efficient*, *privacy-preserving*, and *usable*. Obviously, these are large fields for research, and thus we cannot cover all of them exhaustively in this paper. Therefore, our goal for this paper is to present a selection of concepts and methods to tackle these problems which we have investigated in our research on public sensing systems within the DFG Com'N'Sense project [ComNsense] at University of Stuttgart – methods for ensuring location privacy can be found in our work of the DFG PriLoc project [PriLoc]. In detail, we focus on the following three concepts and methods:

- *Virtual sensor abstraction:* Virtual sensor provide an elegant API for applications to task the public sensing system and let applications query sensor values without having to know the internal details of the public sensing system like mobile node population or node positions.
- *Model-driven sensing:* The data to be gathered by the public sensing system is often correlated or follows certain patterns. For instance, sensors at similar locations might report similar values. If we can capture these correlations in appropriate models, we can avoid physical sensing operations by reading data from the model instead. Obviously, this reduces the energy consumption of mobile devices. Moreover, with those models we can fill "white spots", which are currently not covered by any mobile node, by deriving values from the model to increase the quality of sensing in terms of coverage.
- *Efficient mobile communication of sensor values:* Communication technologies as typically used on mobile phones have certain characteristics that can be exploited to improve the energy-efficiency of communicating sensor values.

The rest of this paper is structured according to these three topics: In Section 2, we introduce the virtual sensor abstraction. In Section 3, we explain model-driven sensing using two different application scenarios: capturing basic sensor information like temperature values, and automatically deriving indoor models from crowd-sensed mobility traces. In Section 4, we show how to optimize the communication of sensor values by considering the characteristics of mobile communication technologies. Finally, we summarize the paper in Section 5.

2. VIRTUAL SENSOR ABSTRACTION

We start our presentation of concepts and methods for public sensing systems with a virtual sensor abstraction tailored to querying values from a public sensing system.

In a public sensing system consisting of potentially billions of constantly moving mobile devices it would be cumbersome for applications to query sensor values by specifying the identity of the mobile nodes which should capture sensor data since then the application would need details about the node population and node locations. Instead, it is much more attractive for the application programmer to offer him an API that abstracts away from the population and mobility of mobile nodes. Virtual sensors are such an abstraction, which we first presented in [PDR11].

Basically, a *virtual sensor* $v = (s, t, p)$ looks similar to a stationary physical sensor. It has a position, called the sensing site s, a type t (for instance, temperature, light, noise, etc.), and a sampling period p. Since the sensor is purely virtual, the application can specify arbitrary positions and sampling periods according to the application requirements, for instance, a grid layout of (virtual) temperature sensors sampling every 10 sec, or virtual sensors placed in certain rooms sensing only every minute. During query execution, the public sensing system maps these virtual sensors to a subset of the physical sensors attached to mobile devices that are with a certain maximum distance to the specified sensing site (see δ_{max} below). The selected physical sensors are read with suitable rate to fulfill the requested sampling period.

A *query* $Q = (V, k, \delta_{max})$ issued to the public sensing system consists of a set V of virtual sensors, a covering value k, and a maximum distance δ_{max}. Value k defines, how many sensor readings are required per sample. Remember that sensor values taken by unknown mobile nodes might be inaccurate and imprecise. Using several values, the application can improve the sensing quality by exploiting this redundancy, e.g., by simply averaging these values. Moreover, a set of samples can be used to evaluate the quality using statistical metrics like the standard deviation. Parameter δ_{max} defines a maximum distance that is allowed between the sensing node and sensing site s. This gives the application the possibility to define, which physical readings are considered to be close enough to cover the sensing site.

Based on these parameters, the public sensing system selects mobile nodes to perform physical readings. One possibility to perform this mapping is to provide the public sensing system with a global view onto all sensors including their positions and implemented sensor types, and explicitly task nodes to perform sensing operations. However, this also implies some communication and energy overhead for tracking mobile devices and might raise privacy concerns. Therefore, a more promising possibility is to distribute the sensing queries through broadcasting to all devices and let them decide whether they want to participate in sensing. The challenge with this approach is to adapt to dynamic node densities such that not too many or too less readings are performed without having a global coordinator making the selection and tasking. To this end, we can use a probabilistic approach where every device locally decides to sense with a certain probability, which is adapted according to the node density and/or number of returned results in the past.

3. MODEL-DRIVEN SENSING

We continue by focusing on the question how to improve the quality of public sensing and its energy-efficiency through model-driven sensing.

The basic idea of model-driven sensing is to exploit correlations or patterns of the phenomenon to be sensed. If it is possible to describe these correlations and patterns in models, queries can be answered by consulting the model instead of reading remote physical sensors. Consequently, with model-driven

sensing mobile devices can substantially save energy by reducing the communication and sensing effort. Moreover, a model might also help to alleviate one of the inherent problems of a public sensing system, namely sensor coverage. Although mobile device are ubiquitous today, the coverage of locations might vary over time. For instance, the node population in a city center might be high during week days, but sparser during night or weekends. Thus, it might be impossible to gather a physical sensor value at certain locations in times when no mobile node is close to the requested sensing location. However, if we can learn a model over time, this gap in coverage can be closed by deriving the requested values from this model.

Obviously, such a model depends very much on the application and phenomenon to be sensed. In the following, we present two different model-driven approaches: one for classical sensor values like temperature or noise, and one for crowed-sensed indoor models (floor plans).

3.1. DrOPS: Model-driven Optimization of Public Sensing

Our first approach (called DrOPS: Model-**dr**iven **O**ptimization for **P**ublic **S**ensing [PSA+2013]) is focused on capturing sensor data of spatially distributed environmental phenomena exhibiting a certain correlation between sensor values at different locations. For instance, it is reasonable to assume that the temperature at similar geographic locations like in an air-conditioned building or outdoor temperatures in a quarter of a city are correlated. We utilize multivariate Gaussian distributions (MGD) as a popular modeling technique for such phenomena (e.g., [GKP05, Cre93]).

Note that correlations might not solely depend on the geographic location. For instance, in an air-conditioned building, office workers in different offices might have different preferences and settings for the air-conditioning system. Thus, the correlation of temperatures of non-neighboring offices might be higher in same cases than in neighboring rooms. Consequently, defining a model a priori is not reasonable in many cases. Therefore, learning the model is a more promising approach for model creation. To this end, we can as well use public sensing to learn and continuously adapt the model while executing sensing tasks. Therefore, we distinguish between two operational phases in our approach: the non-optimized and the optimized execution phase.

During the *non-optimized execution phase*, we try to answer all queries through readings of physical sensors, i.e., without exploiting the model. Data gathered in this phase is also used for training the model. In this specific model, a set of virtual sensors is modelled as a set of one-dimensional Gaussian distributions stored as mean vector and covariance matrix. As soon as the model has been created, the system switches to the *optimized operation phase* and uses the model-driven sensing algorithm, which tries to minimize the physical readings conducted during query execution without falling below a given target quality. The problem of selecting a minimum set of virtual sensors has been shown to be NP hard. However, heuristics with good performance have been proposed in the literature [GKP05]. We modified the algorithm from [GKP05] to find a minimal set of sensors such that the variance of sensor values is smaller than a user-defined quality constraint.

Another problem to be solved is that the model might become outdated over time and we have to train it again. Since training requires a significant number of physical readings and thus, energy of mobile nodes, it should be performed only if needed, i.e., when inaccurate values are deferred from an outdated model. In order to detect when the model needs to be trained again, the sensing system selects a small number of additional sensing locations to collect control readings. The physical readings from these locations are compared to the values inferred from the model, and a training phase is started if these values differ too much.

We also evaluated the performance of this approach using a proof-of-concept implementation and simulations with two real data sets: The Intel Lab data set [DGM+04] and data from the Lausanne Urban Canopy Experiment (LUCE) [NBP+09]. Both data sets contain environmental readings, e.g.,

temperature reported by a large set of fixed sensors. For our simulations, we placed virtual sensors at the places of fixed sensors and let 200 mobile nodes capture this data. Figure 1 shows the relative energy consumption of mobile devices using the model-driven approach compared to the non-optimized approach where physical readings are taken for every requested location. In summary, 59 % of energy is saved on average for the Intel Lab data and 74 % for the LUCE data.

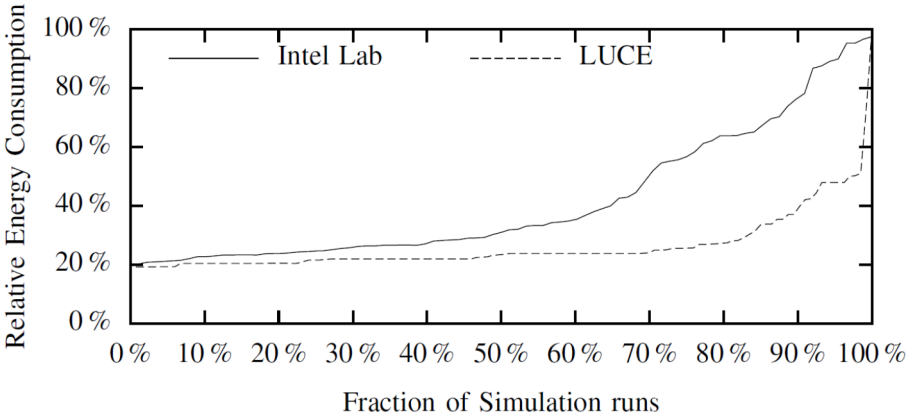


Figure 1: Energy savings of model-driven sensing.

3.2. MapGENIE: Automated Generation of Indoor Maps with Public Sensing

Our second example of a model-driven public sensing approach is the MapGENIE system [PBD+14], which was developed in collaboration with researchers from the Institute for Photogrammetry (IfP) of University of Stuttgart. MapGENIE is motivated by the fact that while outdoor maps are widely available, indoor models, such as floor plans, are not available for most buildings. One reason for this situation is that although it is technically possible to create high-quality 2D and 3D indoor models, this often requires expensive equipment like laser scanners and substantial man power to operate the equipment. We try to improve this situation by developing public sensing methods that enable crowds of (private) users carrying commodity smart phones to automatically generate floor plans from crowed-sensed mobility traces. The basic idea of this approach is that users collect mobility traces with their smartphones while moving through the building. The captured traces are then processed automatically into floor plans containing the outline of rooms and corridors.

Although we are not the first to propose a public sensing system to automatically generate floor plans from crowed-sensed mobility traces, our approach is the first that improves the quality of plans and effort for collecting traces using a model-driven approach. This specific model is called an *indoor grammar*, which defines structural information about a building to support the mapping process. The basic idea is that the layout of a building like size of rooms and position of rooms is not completely arbitrary but follows certain architectural rules. For instance, in an office building an assistant's office might be likely located next to an executive's office and both rooms have specific dimensions. Looking at the example in Figure 3, which depicts the computer science building of University of Stuttgart, we can also see a quite regular structure with only few room sizes (e.g., offices for one or two office workers; seminar rooms; maintenance rooms) and well-defined positions for rooms with certain functionality like rest rooms.

These rules can be exploited in several ways. The grammar can be used to compensate for partially missing trace information like inaccessible rooms not covered by any mobility trace. For instance, we can often assume a certain similarity of the layout of different floors in an office building. Thus, it is sufficient to collect a portion of traces for some areas to derive the grammar, and then complete the plan by filling white spots using the grammar. Obviously, this also reduces the effort for

collecting traces since fewer traces are required for a complete model. Secondly, the grammar improves the quality by compensating for trace inaccuracies resulting from imperfect indoor positioning or the impossibility to access parts of a room, for instance, because of furniture.

Figure 2 gives an overview of the approach, which is divided into several steps. *Trace data acquisition* collects mobility traces from mobile devices. This is an opportunistic process, where mobile devices collect traces while users follow their daily routine. However, we can also imagine an approach where users (for instance, as part of a community similar to Open Street Map) actively decided to walk certain routes which require more traces. In our prototype, we used a foot-mounted IMU for positioning and a zero-velocity update protocol [Fox05] to limit the sensor drift. However, using the built-in sensors of smartphones is subject to active research, and could also be integrated with our approach.

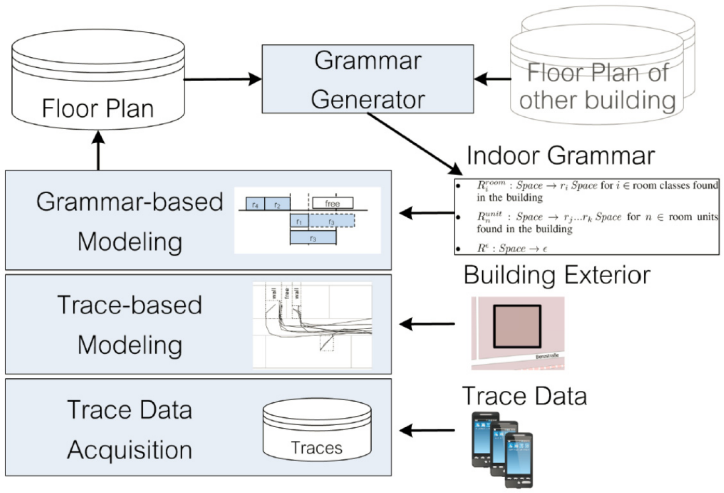


Figure 2: Automatic floor plan generation with MapGenie.

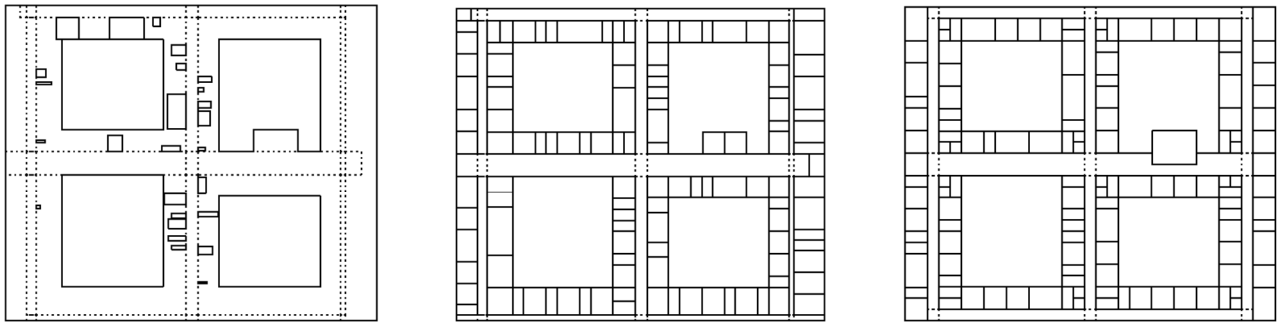
Trace-based modeling processes collected traces into a trace-based model. In this step, we improve the quality of traces by aligning traces, for instance, along building axis, to compensate for the remaining drift. Moreover, we filter out unreliable and corrupted traces. From the corrected traces, we derive hallways and rooms.

In general, the trace-based model is still inaccurate and incomplete. Therefore, we correct and complete the trace-based model in the final *grammar-based modeling* step. Details about the indoor grammar can be found in [BPF+13]. In short, an indoor grammar $G = (N, T, P, S)$ is defined by a set of non-terminal symbols N, a set of terminal symbols $T = \{\varepsilon, r_a, r_b, \ldots\}$, a set of production rules P, and the axiom $S = Space$. The terminal symbols r_i represent classes of rooms with a certain geometric extent. Production rules define room units, which can fill the space with certain sequences of rooms. Furthermore, probabilities are defined for the relative frequency of rooms or room units and conditional probabilities for the neighboring relationship of rooms or room units. A Markov model is used to apply the probabilistic grammar.

Observations from collected mobility traces are used as constraints. A layout error is calculated to define how well the derived model matches the observations. A heuristic is then used to find the layout that minimizes the layout error. Finding the exact solution of this optimization problem is an NP-hard problem.

We evaluated the performance of grammar-based indoor modeling using experiments in a real building. To this end, we collected 250 odometry traces (foot-mounted IMU and zero-velocity update

protocol) of a total length of over 22 km from four volunteers of the 2nd floor of the computer science building in Stuttgart (see Figure 3c). Figure 3a shows the trace-based indoor model using all traces. Out of a total of 74 office rooms in the building, only 26 have been found. Moreover, rooms are only of the right size if they are either very narrow or were detected from multiple (in our case three) traces. Figure 3b shows the grammar-based indoor model. First of all, we can see that this model is complete. In areas where no rooms were found from traces, the grammar provides a plausible layout of rooms. Moreover, we can observe that the layout error is reduced compared to the purely trace-based model. For more detailed evaluation results, we refer to [PBD+14].



(a) Trace-based Indoor Model (b) Grammar-based Indoor Model (c) Ground-Truth Indoor Model

Figure 3: Trace-based/grammar-based indoor models and ground truth model.

4. ENERGY-EFFICIENT UPDATE PROTOCOLS

Finally, we would like to present another method that improves the energy efficiency of public sensing by optimizing the communication between mobile phones and server infrastructure, which we first presented in [BDR13]. Once sensor values have been sensed by the mobile phone, they have to be sent to a server (the "cloud") to store and process the values gathered by the crowd of mobile phones. This communication of sensor values consumes significant energy of mobile phones [BDR12, SSM09], and, thus, should be as small as possible. In the following, we will focus on the communication of position information as required, for instance, by the MapGENIE system presented above to generate maps from mobility traces. However, our general findings also apply to updating other types of values on a server.

To minimize the energy spent for communicating position values, different optimized position update protocols have been proposed in the literature (see [LR01] for an overview), for instance, using dead reckoning. Although these protocols certainly improve energy efficiency, they neglect the properties of the network interface of mobile phones, which have strong impact onto the energy spent for transferring data. In general, these protocols assume that every update consumes the same amount of energy. However, this assumption does not hold true due to the power management implemented by mobile phones. Typically, after communication the network interface is not directly entering power-save (IDLE) mode, but stays in high-power mode for a certain amount of time called tail time (cf. Figure 4), which might last several seconds – actually, the high-power state consists of two states DCH and FACH as shown in Figure 4a, where the latter is slightly less energy-intensive but still consumes much more energy than IDLE mode. Consequently, sending even only a single byte might lead to a large energy overhead if the network interface has to be woken up from IDLE mode and then has to wait the tail time before entering the IDLE mode again (if no other data has been sent meanwhile).

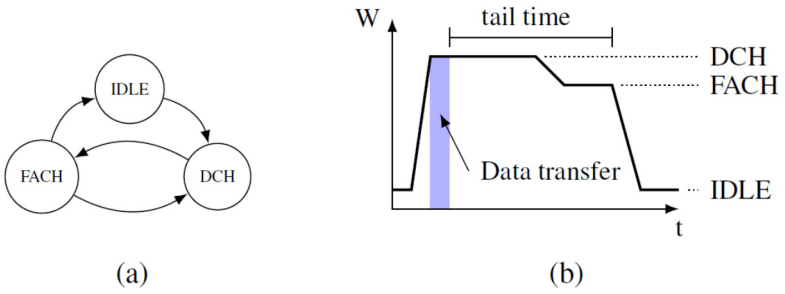


Figure 4: Energy characteristics of mobile communication.

Therefore, we argue that we have to take these characteristics into account when designing update protocols. To this end, we proposed extended position update protocols, namely, an optimized time-based protocol, distance-based protocol, and an optimization of dead reckoning. In the following, we will focus on the time-based protocol; details about the other protocols can be found in [BDR13].

The basic idea is to send position updates opportunistically together with other "background" traffic of the mobile phone stemming from web-browsing, e-mail, instant messaging, etc. The rationale for this is to send position updates primarily when the network interface is powered up anyway. By doing so, an extra tail time for position updates can be avoided. Obviously, this opportunistic strategy is only effective if there are enough opportunities to send the position updates back-to-back with existing background traffic. Note that in order to avoid stale position data on the server, we cannot wait arbitrarily long before sending an update. If there is a position update to be sent and no other background traffic occurs within a certain time span, we have to force a send operation powering up the network interface. In order to show that an opportunistic strategy is indeed reasonable, we conducted a study with 12 users using their (Android) smartphones during their daily routine to measure the inter-arrival time of traffic. We found out that inter-arrival times follow and exponential distribution, where short inter-arrival times occur very frequently (89 % of the inter-arrival time are smaller than 1 s). Thus, even in the case of high position update rates there are many opportunities to send updates back-to-back with background traffic.

Knowing that an opportunistic update protocol has the potential to improve energy efficiency, it remains the question, when is the *optimal* time to send an update? Sending an update with every outgoing traffic might be sub-optimal since we need to guarantee only a certain update rate at the server, e.g., one position update every 10 seconds. Obviously, exceeding this update rate is just a waste of energy. On the other hand, if we miss to send an update together with background traffic within the 10 seconds time span, we might have to force an update without background traffic. Clearly, if we had a priori knowledge of all background traffic, we would select the last background message just before the 10 seconds deadline and send the update back-to-back. However, since in practice no perfect knowledge of future traffic is available, the best we can do is to *predict* future background traffic. To this end, we have utilized a Markov decision process (MDP), which is basically a Markov chain, where actions (send/don't send sensor value) can be executed in states, and costs (modelling energy consumption) are associated with state transitions. Using this MDP, an optimal action can be determined for each state to minimize the sum of expected costs. Further details about this process can be found in [BDR13].

To evaluate the performance of our optimized update protocol, we implemented a simulation using a realistic energy model for 3G mobile communication from [LZZ11] based on empirical measurements on an HTC G1 smartphone. Node mobility was simulated using real GPS traces from the Open Street Map project [OSM]. Figure 5 shows the results for the time-based update protocol,

which tries to send one position update every δ_T seconds. We compare our opportunistic predictive approach with MDP (denoted as "extend") to a basic non-opportunistic approach ("basic"), which sends updates periodically every δ_T seconds. Moreover, we show the benefits of using a predictive approach based on MDP by also comparing to a non-predictive opportunistic approach ("nopred") sending updates with every outgoing message. We can see that our opportunistic predictive approach consumes only 70 % of the energy of the basic approach. Interestingly, the non-predictive opportunistic approach "nopred" consumes even more energy than the basic approach for longer update intervals since it sends too many updates. Thus, prediction is essential for longer update periods.

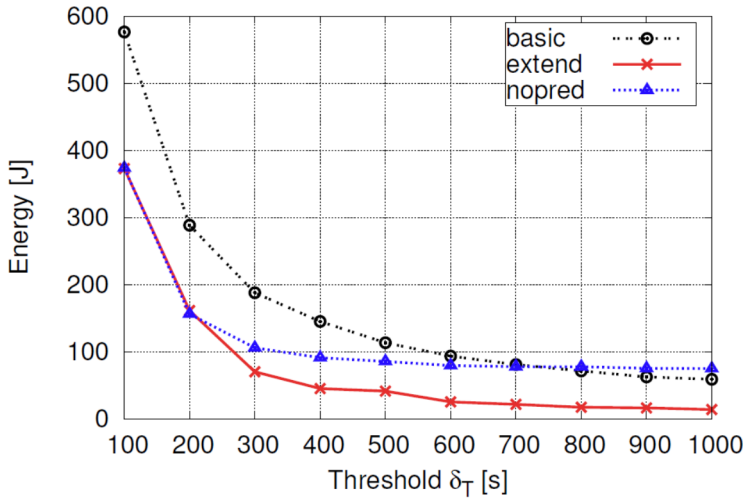


Figure 5: Energy consumption of update protocols.

5. SUMMARY

In this paper, we first motivated public sensing as a new concepts to gather sensor data from mobile devices carried by a large population of mobile users. By leveraging the crowd of mobile users, we can build a mobile sensor network with excellent coverage in urban areas without large up-front investments for building a dedicated sensor network.

In order to enable the implementation of public sensing systems, we first presented a virtual sensor abstraction facilitating the usability of a public sensing systems consisting of huge numbers of mobile devices through a node-transparent and mobility-transparent API.

Moreover, we presented two methods to ensure a specified quality of sensing and to optimize the public sensing process w.r.t. the energy-efficiency of mobile devices involved in sensing – an essential requirement to ensure the user acceptance of public sensing: (1) Model-driven sensing exploits models to fill "white-spots" which are not covered by mobile devices and to reduce the number of physical sensing operations by reading data from the model to save energy. (2) Energy-efficient update protocols reduce the amount of energy required for communicating sensor values from the mobile device to a server infrastructure ("the cloud").

6. ACKNOWLEDGEMENTS

We would like to acknowledge the work from all of our partners, who significantly contributed to the results presented in this paper, namely, Patrick Baier, Susanne Becker, Dieter Fritsch, Michael Peter, and Damian Philipp.

The work presented in this paper was partially funded by German Research Foundation (DFG) within the Com'N'Sense project.

7. REFERENCES

[BDR12] P. Baier, F. Dürr, K. Rothermel: TOMP: opportunistic traffic offloading using movement predictions. In Proceedings of the 37th IEEE Conference on Local Computer Networks (LCN 2012), Clearwater, Florida, USA, October 2012

[BDR13] P. Baier, F. Dürr, K. Rothermel: Opportunistic position update protocols for mobile devices. In Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013). pp. 787-796, Zurich, Switzerland, September 2013

[BPF+13] S. Becker, M. Peter, D. Fritsch, D. Philipp, P. Baier, and C. Dibak: Combined grammar for the modeling of building interiors. In ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2013.

[ComNsense] Web page of the ComNsense project: http://www.comnsense.de/. Last accessed June 2015.

[Cre93] N. A. C. Cressie: Statistics for spatial data. Wiley-Interscience, 1993.

[DGM+04] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, W. Hong: Model-driven data acquisition in sensor networks. In Proceedings of the 30th International Conference on Very Large Data Bases (VLDB 2004), pp. 588-599, Toronto, Ontario, Canada, 2004.

[Fox05] E. Foxlin: Pedestrian tracking with shoe-mounted inertial sensors. IEEE Computer Graphics and Applications, 25, pp. 38-46, 2005.

[GKP05] C. Guestrin, A. Krause, A. P. Singh: Near-optimal sensor placements in Gaussian processes. In Proceedings of the 22nd International Conference on Machine Learning, pp. 265-272, Bonn, Germany, 2005.

[LR01] A. Leonhardi, K. Rothermel: A comparison of protocols for updating location information. Cluster Computing, 4 (4), pp. 355-367, 2001.

[LZZ11] H. Liu, Y. Zhang, and Y. Zhou: TailTheft: leveraging the wasted time for saving energy in cellular communications. In Proceedings of the 6th International Workshop on MobiArch (MobiArch 2011), pp. 31-36, New York, NY, USA, 2011.

[MHV+13] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, V. D. Blondel: Unique in the Crowd: The privacy bounds of human mobility. Scientific Reports 3, 2013

[NBP+09] D. Nadeau, W. Brutsaert, M. Parlange, E. Bou-Zeid, G. Barrenetxea, O. Couach, M.-O. Boldi, J. Selker, M. Vetterli: Estimation of urban sensible heat flux using a dense wireless network of observations. Environmental Fluid Mechanics, 9, pp. 635-653, 2009.

[OSM] Web page of the Open Street Map project: http://www.openstreetmap.org/. Last accessed June 2015.

[PBD+14] D. Philipp, P. Baier, C. Dibak, F. Dürr, K. Rothermel, S. Becker, M. Peter, D. Fritsch: MapGENIE: Grammar-enhanced indoor map construction from crowd-sourced data. In Proceedings of the 12th IEEE International Conference on Pervasive Computing and Communications (PerCom 2014), Budapest, Hungary, March 2014.

[PDR11] D. Philipp, F. Dürr, K. Rothermel: A sensor network abstraction for flexible public sensing systems. In Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011), pp. 460-469, Valencia, Spain, October 2011.

[PriLoc] Web page of the PriLoc project: http://www.priloc.de/. Last accessed June 2015.

[PSA+2013] D. Philipp, J. Stachowiak, P. Alt, F. Dürr, K. Rothermel: DrOPS: model-driven optimization for public sensing systems. In Proceedings of the 11th IEEE International Conference on Pervasive Computing and Communications (PerCom 2013), pp. 185-192, San Diego, CA, USA, March 2013.

[SSM09] A. Shye, B. Scholbrock, G. Memik: Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. In Proceedings of the 42nd International Symposium on Microarchitecture (MICRO-42), pp. 168-178, New York, NY, USA, 2009.