VeSense: Energy-Efficient Vehicular Sensing

Jong Hoon Ahnn
Department of Computer Science, UCLA, USA
jhahnn@cs.ucla.edu

Miodrag Potkonjak
Department of Computer Science, UCLA, USA
miodrag@cs.ucla.edu

***Abstract*—Although vehicular sensing where mobile users in vehicles continuously gather, process, and share location-sensitive and context-sensitive sensor data (e.g., street images, road condition, traffic flow) is emerging, little effort has been investigated in a model-based energy-efficient network paradigm of sensor information sharing in vehicular environments. Upon these optimization framework, a suite of optimization subproblems: a program partitioning and network resource allocation problem, we propose a distributed vehicular sensing platform, called VeSense where mobile users in vehicles publish/access sensor data via a cloud computing-based distributed P2P overlay network. The key objective is to satisfy the vehicular sensing application's quality of service requirements by modeling each subsystem: mobile clients, wireless network medium, and distributed cloud services. By simulations based on experimental data, we present the proposed system can achieve up to 37 times more energy-efficient and 73 times faster compared to a standalone mobile application, in various vehicular sensing scenarios applying a realistic mobility model.**

I. Introduction

We have observed that rising popularity of smartphones with onboard sensors (e.g., GPS, compass, accelerometer) and always-on mobile Internet connections via 3/4G sheds lights on using smartphones as a platform for large-scale vehicular sensing. Recent reports estimated that smartphone users will catch and surpass feature phone users in the U.S. by 2011, reaching more than 150 million users [6]. For instance, 10 million mobile users could generate sensor data at the rate of 1KB/s per user (e.g., GPS, accelerometer, WiFi scanning data) and also send queries, requiring networking systems with a sheer amount of bandwidth (>80Gbps), storage space(>36TB/hr), and computational power. Thus, there is a need for location-aware and energy-aware sensor networking systems that can facilitate information sharing among millions of mobile users via always-on 3/4G connections.

Although many researchers have studied vehicular sensing, little attention has been paid in a model-based cost optimization with the consideration of energy saving of mobile devices and cloud services at the same time. Furthermore, without having concrete models in the performance of application and wireless communication medium, it is difficult to quantify the cost of operations due to dynamic nature of vehicular sensing applications. As depicted in Figure 1, the overall system model may include several subsystems: mobile terminals, multiple wireless network interfaces, and cloud services. We model the computation cost statistically while we model the communication cost theoretically. The reasoning behind is that once an application is profiled on a specific mobile device and cloud machine, the computing cost stays similar unless network conditions change. Note that mobile network traffic is highly bursty in many times. Thus, obtaining real-time network parameters can be costly due to the heavy scanning cost, and this situation is against our goal to save energy. To profile network conditions in an energy-efficient manner, we adopt an analytic cost of communication by compromising the high accuracy. We believe combining empirical and analytical profiling costs can enhance the overall system performance in providing real-time optimal offloading strategies for resource- and energy-constrained mobile clients.

In Vehicular Sensing, Internet-based approaches for *generic* sensor data sharing have a simple multi-tier structure. In ArchRock and SensorBase [9], sensor data from a sensor network is aggregated at the local gateway and is published to the front-end server through which users can share the data. SensorMap [8] is a web portal service that provides mechanisms to archive and index data, process queries, and aggregate and present results on geocentric Web. In IrisNet [10], each organization maintains database servers for its own sensors, and a global naming service is provided for information access. VeSense [1] differs from these approaches in that it focuses on location-sensitive information sharing via a scalable structured P2P overlay.

In Mobile Cloud Computing, MAUI [11] seems promising because their model incorporates a cost model for deciding best execution configuration. Cloudlets [12] allows high abstraction and personalization of the computing environment by using VMs, but lack from fine-grained execution adaptation. Prior work mostly focused on saving energy consumption on mobile devices; in contrast, VeSense provides analytical cost models to optimize the entire energy consumption including network and cloud at the same time.

The key contributions are summarized: we explicitly model subsystems of energy-efficient vehicular sensing platform using two aspects: computation and communication cost; we propose a distributed optimized solution of complex energy-efficient vehicular sensing; we propose a location-aware sensor data retrieval scheme called VeSense that supports geographic range queries, and a location-aware publish-subscribe scheme that enables energy-efficient multicast routing over a group of subscribed users.

[1]This work is funded in part by Samsung global research outreach program 2011-2012, award number 20112465.

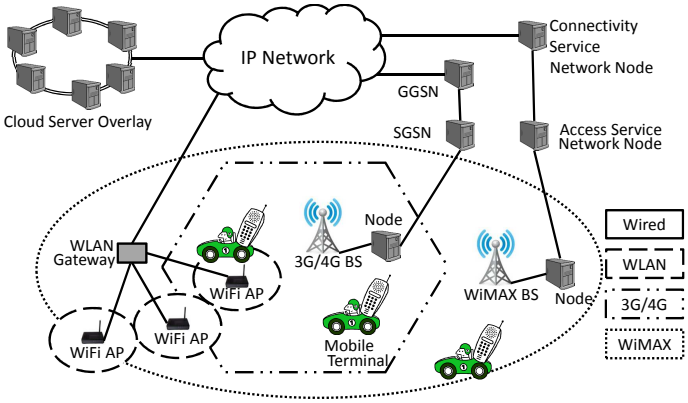

Fig. 1. A high-level overview of smartphone-based vehicular sensing network architecture in a heterogeneous wireless network interfaces scenario.

II. System Model

The system model of vehicular sensing falls in two fold. The communication cost acts as fat mobile client that perform all the computation locally, while the computation cost splits its computation into local and remote parts, thus may incur additional communication cost to transfer its necessary binary and data, however save the total amount of local computation.

A. Computation Model

We define a software program as a set of basic functional blocks (BFB)s, where a basic functional block corresponds to a single method or function in a program. Each BFB consists of a set of inputs as required knowledge of computation, and a set of outputs as an outcome of computation. These include both global and local variables defined in a program.

In order to model the performance of mobile sensing applications in heterogeneous hardware environments, we apply a regression theory to derive statistical inference models, by taking a small number of samples, where each sample denotes the execution time of a BFB on a particular machine. In our regression model, a response is modeled as a weighted sum of predictor variables. By adopting statistical techniques, we then assess the effectiveness of model's predictive capability.

We suppose there are a subset of observations $\hat{\Theta}$ in a large observation space Θ for which values of response and predictor variables are known. A observed response vector is denoted by $\mathbf{y} = [y_1, ..., y_i, ..., y_\theta]$, where y_i denotes it response variable for a single observation $i \in \hat{\Theta}$ and a Φ predictor vector is denoted by $x_i = [x_i^\phi, ..., x_i^\phi]$. The corresponding set of regression coefficients is expressed by a vector $\Gamma = [\gamma_0, ..., \gamma_\phi]$. Thus, a linear function of predictors Φ is given by,

$$f(y_i) = \Psi(x_i)\Gamma + \epsilon_i = \gamma_0 + \sum_{j=1}^{\Phi} \Psi_j(x_i^j)\gamma_j + \epsilon_i \qquad (1)$$

γ_i can be seen as the expected change in y_i per unit change in the predictor variable x_i^j. An independent random error ϵ_i has mean $E(\epsilon_i) = 0$ and constant variance $Var(\epsilon_i) = \sigma^2$.

In order to determine the best fitting model, we consider least squared errors commonly used in minimizing $\Omega(\Gamma)$ the sum of squared deviations of predicted responses give by the model from observed responses.

$$\Omega(\Gamma) = \sum_{i=1}^{\hat{\Theta}} (y_i - \gamma_0 - \sum_{j=1}^{\Phi} \gamma_j x_i^j)^2 \qquad (2)$$

Obtaining estimates of the coefficients Γ is the goal of this approach. The correlation of response-to-predictor relationship is used in identifying the significance of the estimates. We express residuals to answer the problem of how well the model captures observed trends as,

$$\hat{\epsilon} = y_i - \hat{\gamma}_0 - \sum_{j=0}^{\Phi} \hat{\gamma}_j x_i^j \qquad (3)$$

Model fitting can be assessed by the F-test which is a standard statistical test method using multiple correlation statistic R^2 given by

$$R^2 = 1 - \frac{\sum_{i=1}^{\hat{\Theta}} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{\hat{\Theta}} (y_i - \frac{1}{\hat{\Theta}} \sum_{i=1}^{\hat{\Theta}} y_i)^2} \qquad (4)$$

A larger R^2 value indicates better fits, while over-fitting if R^2 is close to 1. The over-fitting may occur when data sets are small and the number of predictors is large. A typical strategy is to set the number of predictors less than the number of observations given by $|\Phi| < \frac{\hat{\Theta}}{20}$ according to [2].

$$\hat{y}_i = E[\gamma_0 + \sum_{j=1}^{\Phi} \gamma_j x_i^j + \epsilon_i] \quad = \gamma_0 + \sum_{j=1}^{\Phi} \gamma_j x_i^j \qquad (5)$$

The Equation 5 presents the expected value of y_i, $E[y_i]$ and its corresponding estimate $\hat{y}_i$ with $E[\epsilon_i] = 0$. We herein define a coefficient of performance η of a computer such as a mobile client and cloud server. The coefficient converts the performance in time $\hat{y_i}$ into the one in power or energy $\hat{P}_i$ on a computer j in Equation 6.

$$\hat{P_i^j} = \eta \cdot \hat{y}_i^j, \qquad (6)$$

where η can be experimentally obtained.

B. Wireless Network Model

We consider multiple wireless network interfaces scenario where heterogeneous radio access technologies (RAT)s such as WIFI, UMTS, and GSM with their overlapping network coverage in a given area. According to many researchers such as [3] and [5], RATs can be largely characterized into two categories based on means to share their channels: interference constrained RATs and orthogonal RATs. In this paper, we focuse on interference constrained RATs.

Interference constrained RATs such as UMTS are network interfaces where their bandwidth is equally distributed in

mobile sensing terminals, thereby their resources are assigned according to the assigned power within a base station. The signal to interference and noise ratio (SINR) between a sensor client m, m' and BS b, b' can be given by

$$\nu_{m,b} = \frac{q_{m,b}p_{m,b}}{\delta q_{m,b}\sum_{m' \neq m} p_{m',b} + \sum_{b' \neq b} q_{m,b'}P_{b'} + \omega} \quad (7)$$

δ denotes a factor for orthogonality which represents the degree of intercell interference, and $q_{m,b}$ denotes the channel gain for a mobile sensor client m in BS b. $p_{m,b}$ denotes the assigned power to a mobile sensor client m in BS b, and $\sum_m p_{m,b} = P_b$ which is constrained by $\bar{P}_b$. The data rate experienced from each mobile user is sensitive to both intracell and intercell interference. The assigned data rate $D_{m,b}$ for a mobile client m in BS b can be modeled as,

$$D_{m,b} = a\log(1 + c\nu_{m,b}) \quad (8)$$

We use positive constants a and c as system parameters such as bandwidth, modulation, and bit-error rates. Note that the feasible data rate $D_{m,b}$ is not convex. In order to formulate a convex optimization problem, we further assume all base stations have a fixed transmission power, thus we can approximate a mobile user's data rate $\bar{D}_{m,b}$ by,

$$\tilde{D}_{m,b} = a\log(1 + c\frac{p_{m,b}}{\frac{\delta q_{m,b}\bar{P}_b + \sum_{b' \neq b} q_{m,b'}P_{b'} + \omega}{q_{m,b}} - \delta p_{m,b}}) \quad (9)$$

$$= a\log(a + c\frac{p_{m,b}}{\pi_{m,b} - \delta p_{m,b}}) \approx (\frac{\alpha}{\pi_{m,b}})p_{m,b} \quad (10)$$

$$= \bar{D}_{m,b}p_{m,b} \quad (11)$$

III. Optimization Problem Formulation

We mainly solve a program partitioning and network resource allocation problem. A solution to the partitioning problem gives an optimal set of code offloading decisions in terms of computation cost and communication cost, while a solution to the network resource allocation problem gives an optimal allocation strategy toward maximizing the utility of network systems. It is obvious that solving the latter problem provides a way to choosing the best communication cost in the former problem.

A. Program Partitioning Problem

Let us consider a mobile application A and its call function graph $G = (V, E)$, where each vertex $v \in V$ denotes a method in A. An invocation of method v from one another u thereby is denoted by an edge $e = (u, v)$. We annotate each vertex with the execution time T_v of the method v and each edge with the data transfer time $T_{u \to v}$ incurred when the method v is offloaded from the method u. We reconstruct a new graph $G' = (V', E')$ from G by adding corresponding offloading methods to V. The code partitioning problem based on G' can be formulated as,

$$\begin{aligned} min \quad & \textstyle\sum_{v' \in V'} T_{v'} + \sum_{e' \in E'} T_{e':u' \to v'}, \\ s.t. \quad & \frac{\sum_{v' \in V'} T_{v'} + \sum_{e' \in E'} T_{e'}}{\sum_{v \in V} T_v + \sum_{e \in E} T_e} \leq 1, \\ & T_{v'} \geq 0, T_v \geq 0, T_{e'} \geq 0, T_e \geq 0 \end{aligned} \quad (12)$$

The calculation of computation cost $T_v, T_{v'}$ depends upon the performance estimate $\hat{y}_i$ for each basic functional block (BFB) v, v' (see Equation 5). Furthermore, the calculation of communication cost incurred due to code offload is given by,

$$T_{v'} = n_{v'} \times D_{m,b}, \quad (13)$$

where the assigned data rate is denoted by $D_{m,b}$ for a mobile client m in BS b, and the size of data to be transferred due to offloading for BFB v' is given by $n_{v'}$. The data rate for interference constrained RATs is presented in Equation 8. We formulate further problems for how to assign the data rate to each mobile client in Section III-B. The problem formulated in Equation 13 consists of a concave objective over linear constraints, and it becomes convex. Therefore, there are various convex optimization algorithms to solve it from [4].

B. Network Resource Allocation Problem

We consider a utility metric as the effectiveness of allocated resources of networked systems in our optimization problem as,

$$U = \sum_m \sum_b D_{m,b} \quad (14)$$

In order to deal with fairness in resource allocation among mobile clients, the utility function with a weight variable w can construct the α proportional fairness as,

$$U = \sum_m \frac{w_m}{1 - \alpha} \sum_b D_{m,b}^{1-\alpha}, \quad (15)$$

where $0 \leq \alpha < 1$. Now, we present an optimization problem as,

$$\begin{aligned} max \quad & U, \\ s.t. \quad & \textstyle\sum_m \frac{D_{m,b}}{\bar{D}_{m,b}} \leq \Gamma_b, \\ & \textstyle\sum_b D_{m,b} \geq D_{min,b}, \\ & D_{m,b} \geq 0, \end{aligned} \quad (16)$$

where $D_{min,b}$ is the minimum data rate assigned to mobile clients. Our goal is for a network operator to maximize the sum of utility of all mobile users in all base stations. Note that Equation 16 consists of a concave objective over linear constraints and thus is convex. That means there exists various algorithms to solve the problem immediately [4].

IV. Cloud-based Vehicular Sensing Architecture

VeSense is a two-tier sensor networking platform that exploits the P2P-based Cloud servers similar to GeoServ [13]. Since most sensor data is generated on the roads (and most

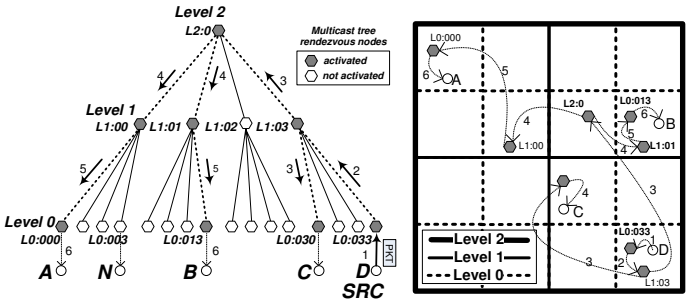

Fig. 2. Subscription-based multicast example: D (source) and A, B, C

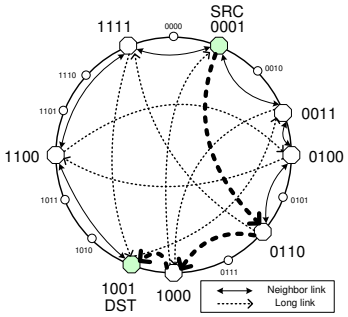

Fig. 3. Illustration of unicast routing: Each node has neighbor links and one long link to a random location. Source located at 0001 sends a packet to the destination node located at 1001. It uses a long link to 0110 followed by neighbor links to 1000 and 1001 sequentially (thick dotted lines).

queries are location sensitive), we assume that the primary search key (or key space) is geographic location. We exploit the computation power of mobile nodes to reduce upload traffic whenever that is possible. Mobile users carry raw sensor data, and the processed data (e.g., average reading, image thumbnails) will be published to the P2P sensor storage.

A. Location-aware sensor data retrieval service

In VeSense, we divide the geographic area of interest into fixed size grids (say $R \times R$), and there are total $2^M \times 2^M$ grids where M is the smallest exponent that covers the entire area. Given this 2D grid space, we use the Hilbert space filling curve [14], a linear mapping function where successive points are nearest neighbors in the 2D grid. The current GeoTable prototype supports simple rectangular area based addressing as $\{(x1, y1), (x2, y2)\}$ that denotes lower left and upper right corners, respectively. Our system can be extended to support more complex shapes using polygons, defined by a set of line segments. For a given rectangular area, nodes first translate the area to find a set of *ordered* segments on the Hilbert curve where a segment is composed of contiguous grid points. Recall that the Hilbert curve looses some of data locality (50% to be precise as the curve connects only two of its neighbors). Thus, it requires a set of segments to cover a rectangular area. We analyze that the expected routing cost of geocasting depends on the size of the target area. The following theorem shows that once a query is routed to the target area at the cost of $O(\log^2 n)$. On the other hand, concurrent geocasting is considered in our prior work [13]. GeoTable uses Mercury's load balancing mechanism to preserve locality of content retrieval [15].

B. Location-aware publish-subscribe service

We have discussed geocasting in the previous section where a one-shot query is routed from an application to the region of data sources. We propose GeoPS, a *publish-subscribe service* where the data updates on a region are *published* to all users who have *subscribed* to that region. This section details GeoPS's locality-preserving multicast tree construction and management methods and their performance bounds via mathematical proofs [13].

For multicast true construction, each group has a unique group ID which is the hash of the group's textual name concatenated with random string, e.g., hash("congestion at grid x, y + !?*2@"). This group ID is used for building a multicast tree per group, similar to node ID in HGLS. For a given $groupID$, we construct a multicast tree rooted at the rendezvous point in level M (top level) using HGLS-like geographic partitioning as follows. Recall that the geographic area is divided into $2^M \times 2^M$ fixed grids where each grid is given as $R \times R$. At each hierarchy level i, we have a rendezvous point located at $(h_{i,x}(groupID), h_{i,y}(groupID))$. This location is mapped to Hilbert curve space, and the overlay node with node ID closest to this mapped address is selected as a rendezvous point in the overlay network.

When a node joins, the join request message propagates to upper levels starting from level 0 (where the node is currently located), and at each level, a node stores subscription information in the routing table for $groupID$. Note that routing to a rendezvous point is done via geocasting (with a single grid point) described in the previous section. When the message finds that there is an existing subscription entry for a given $groupID$, the rendezvous points in its upper levels were already initialized by other group members (a subscription entry of the group is already present). Thus, the message stops there, and the child node is simply added to the table (i.e., a direct path to the child). In Figure 2, when mobile user A joins, the subscription message is installed at L0:000, L1:00, and L2:0 sequentially. The leave process is similar to the join process.

For mobility handling, a mobile client's subscription needs to be updated (to upper layers) whenever the client crosses the level boundary (via explicit leave and join). When there is a single subscriber for a given group, and this client crosses level m boundary, all rendezvous points at and below level $m + 1$ need to be updated. In Figure 2, when mobile client C moves to the adjacent grid on the left (crossing level 1 boundary), rendezvous points at level 0, 1, 2 are updated; and when mobile client D moves to the adjacent grid upward (crossing level 0 boundary), those at level 0, 1 are updated. Interestingly, given that an overlay node typically keeps a fraction of grid space, one possible optimization would be not notifying updates as long as a mobile client is associated with the same overlay node.

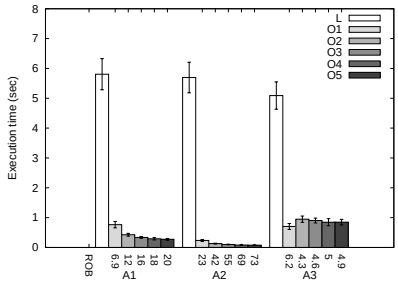

Fig. 4. The average execution time and relative offloading benefit (ROB) of three vehicular sensing applications with various offloading scenarios are measured and presented with 95% CIs.

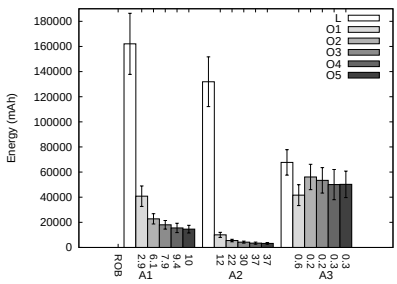

Fig. 5. The average energy consumption and relative offloading benefit (ROB) of three vehicular sensing applications with various offloading scenarios are measured and presented with 95% CIs.

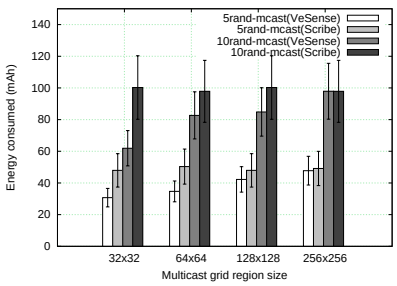

Fig. 6. Subscription-based multicast routing comparison in energy consumption: VeSense vs. Scribe.

V. System Evaluation

Evaluating the performance consists of two parts: the performance in a mobile device and the one in a cloud machine. The former is presented in the computation cost and communication cost in five offload scenarios. We consider three different vehicular sensing applications on the road. In *street-level traffic flow information services* (A1), vehicles as sensors collect GPS measurements and can share data using wireless connectivity. In *vehicular safety warning services* (A2), non-time critical safety warning messages can be timely delivered over 3/4G networks (due to large RTT). For *ride quality monitoring services* (A3), municipalities have been profiling roads using expensive profiling devices mounted on the vehicles that use GPS, accelerometer/laser sensors [7]. For simplicity, interface constrained RATs such UMTS are only considered. Unless specified, network parameters are given form [5].

Figure 4 compares the execution time between the mobile and the cloud by applying to three different vehicular sensing applications: A1-A3, while Figure 5 compares energy consumption in the same settings. These results present all tested vehicular sensing applications can benefit up to 73 times faster in time and 37 times more energy-efficient by utilizing parallel offloading. As discussed, we also study how concurrent offloading requests help save time and energy in various scenarios: O1-O5. We observe the overall time saving and energy saving rate increases as the number of concurrent requests increases. This is done by a non-blocking (asynchronous) offload request.

To show the geographic locality of our subscription-based multicast routing in the cloud server overlay, we increase the width of the region in which all the multicast receivers lie. The region size ranges from 32×32 to 256×256. We vary the origin of the region from (0, 0) to the maximum allowable, e.g., for 32×32, it is (224, 224), and report the average hop count, querying time, and energy consumption for such queries. We compare the performance of GeoPS with Scribe multicast routing protocol [1]. Recall that Scribe destroys the locality by using consistent hashing. We randomly choose 5 or 10 random grids within the region, and each grid is assigned with a subscriber (5 or 10 subscribers). We measure the aggregated number of hop counts to deliver a packet to all the multicast receivers as detailed in Figure 2. Figure 6 clearly shows that our multicast routing exploits the locality of receivers in energy. As the area size (where the subscribers lie) increases, geographic locality among subscribers disappears, and accordingly, the cost of VeSense increases, converging to that of Scribe in the case of 256×256.

VI. Conclusion

This paper presented a distributed vehicular sensing platform and showed it can perform up to 73 times faster and 37 times more energy-efficient compared to a standalone vehicular sensing application.

References

[1] M. Castro, P. Druschel, A-M. Kermarrec, A. Rowstron. Scribe: A Large-scale and Decentralised Application-level Multicast. *JSAC*, 2002.
[2] F. Harrell. Regression modeling strategies. *Springer*, 2001.
[3] I. Blau, G. Wunder, I. Karla, R. Sigle. Decentralized Utility Maximization in Heterogeneous Multicell Scenario. *EURASIP*, Jan. 2009.
[4] S. Boyd, L. Vandenberghe. Convex Optimization. *Cambridge University Press*, New York, NY, USA, 2004.
[5] J. Buhler, G. Wunder. An optimization framework for heterogeneous access management. *WCNC*, 2525-2530, 2009.
[6] USA to Add 80 Million New Smartphone Users by 2011. http://twittown.com/mobile/mobile-blog/usa-add-80-million-new-smartphone-users-2011.
[7] Pavement Interactive Core: Roughness. http://pavementinteractive.org/index.php?title=Roughness.
[8] S. Nath, J. Liu, and F. Zhao. SensorMap for Wide-Area Sensor Webs. *IEEE Computer Magazine*, 40(7), Jul. 2007.
[9] S. Reddy, G. Chen, B. Fulkerson, S. J. Kim, U. Park, N. Yau, J. Cho. Sensor-Internet Share and Search. *DSI*, 2007.
[10] P. B. Gibbons, B. Karp, Y. Ke, S. Nath. IrisNet: An Architecture for a Worldwide Sensor Web. *IEEE Pervasive Computing*, 2(4):22-33, 2003.
[11] E. Cuervoy, A. Balasubramanianz, D-K Cho, A. Wolmanx, S. Saroiux. MAUI: Making Smartphones Last Longer with Offload. *MobiSys*, 2010.
[12] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE PerCom*, 8(4):14-23, 2009.
[13] J. H. Ahnn, U. Lee, and H. J. Moon. GeoServ: A Distributed Urban Sensing Platform. *CCGRID*, 2011.
[14] H. V. Jagadish. Linear Clustering of Objects with Multiple Attributes. *SIGMOD*, 1990.
[15] A. R. Bharambe, M. Agrawal, S. Seshan. Mercury: Supporting Scalable Multi-Attribute Range Queries. *SIGCOMM*, 2004.
[16] J. Härri, M. Fiore, F. Fethi. VanetMobiSim: Generating Realistic Mobility Patterns. *VANET*, 2006.