ARTICLE IN PRESS

Pervasive and Mobile Computing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

VeSense: High-performance and energy-efficient vehicular sensing platform☆

Jong Hoon Ahnn*, Miodrag Potkonjak

Department of Computer Science, UCLA, USA

ARTICLE INFO

Article history:
Received 1 February 2013
Received in revised form 4 April 2013
Accepted 20 May 2013
Available online xxxx

Keywords:
Vehicular sensing
Mobile cloud computing
Program partitioning
Convex optimization
Resource allocation
Network selection

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 frameworks, 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.

1. 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 US by 2011, reaching more than 150 million users [1]. In 2013, we expect to have billions of mobile users. For instance, 10 million mobile users could generate sensor data at the rate of 1 kB/s per user (e.g., GPS, accelerometer, WiFi scanning data) and also send queries, requiring networking systems with a sheer amount of bandwidth (>80 Gbps), storage space (>36 TB/h), 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.

Many researchers and engineers traditionally consider vehicular sensing based on embedded sensors. We however observe billions of mobile devices on the move that are a different form of sensors which require energy saving. Electric vehicles powered by batteries are another big trend. We observe that these trends incur lots of energy issues in the case where mobile devices are connected with gas-powered vehicles or electric vehicles (e.g., Google UAV).

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

* Corresponding author. Tel.: +1 3104871938.
E-mail addresses: jhahnn@gmail.com, jhahnn@cs.ucla.edu (J.H. Ahnn), miodrag@cs.ucla.edu (M. Potkonjak).

1574-1192/$ – see front matter © 2013 The Authors. Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.pmcj.2013.05.001

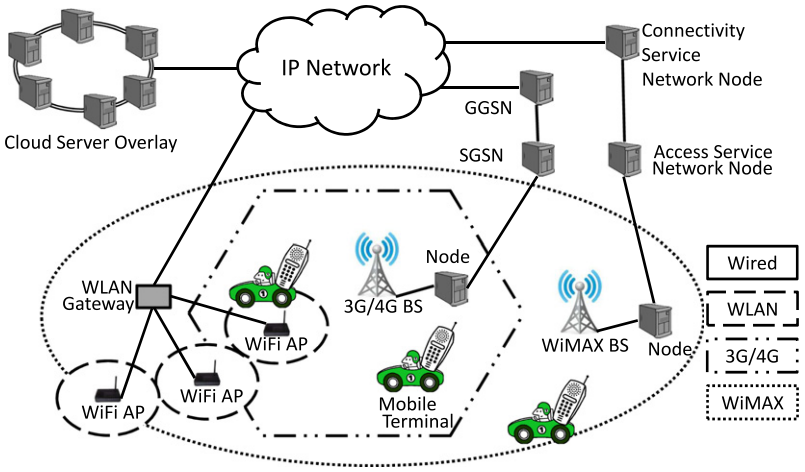

Fig. 1. A high-level overview of smartphone-based vehicular sensing network architecture in a heterogeneous wireless network interfaces scenario.

cost of operations due to dynamic nature of vehicular sensing applications. As depicted in Fig. 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 that 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 [2], 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 [3] 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 [4], 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 [5] seems promising because their model incorporates a cost model for deciding best execution configuration. Cloudlets [6] 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.

Our prior work, GeoServ [7] mainly focuses on how to store in and retrieve sensor data from external storage systems, where the location-awareness is the main consideration on its data management over an overlay-based P2P routing. Thus, GeoServ is a general purpose urban sensing P2P storage with no consideration of performance modeling, energy saving and optimization, and computation offloading. As an ongoing effort of building urban sensing applications, we propose VeSense, a model-based "energy-efficient" system of sensor information sharing and computation offloading in urban environments. Unlike GeoServ, VeSense focuses on the performance/energy modeling and computational offloading by formulating cost functions (computation and communication cost). It also provides a way to optimize program execution time and energy for a given mobile application since the battery constraint is one of the biggest challenges in mobile phones. We adopted GeoServ as a sensor data and computing resource management scheme which can be nicely integrated in our performance and energy optimization framework. Another contribution to VeSense is to formulate a suite of optimization formulations for mobile devices, which was never considered in GeoServ: a program partitioning program, network resource allocation problem, network selection problem. Therefore, VeSense much looks like a modern mobile cloud computing platform, focusing on enabling performance- and energy-efficient urban sensing applications.

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.

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001

2. Related work

Urban sensing: Internet-based approaches for *generic* sensor data sharing have a simple multi-tier structure. In ArchRock and SensorBase [2], 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. In SensorBase, back-end servers (called republishers) further process sensor data to enable sensor data searching. SensorMap [3] 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 [4], each organization maintains database servers for its own sensors, and a global naming service is provided for information access; GSN allows users to query local and remote sensor data sources. VeSense differs from these approaches in that it focuses on large-scale participatory sensing and facilitates location-sensitive information sharing via a scalable structured P2P overlay.

DHT-based overlay: Structured overlay networks (or DHTs) such as Chord, CAN, and Pastry (and file systems based on DHTs such as CFS and PAST [8]) provide efficient, scalable, multicast methods of locating and storing resources over the overlay network. However, since these systems use *consistent hashing* to map node ID and keyword to key space (i.e., DHT only provides exact match queries), it is non-trivial to support complex queries such as range queries. Chawathe et al. [9] proposed a Prefix Hash Table (PHT) that is a trie-like data structure to provide a range query on top of the DHT layer. PIER [10], a distributed query engine based on DHTs provides rich declarative SQL queries such as equi-join. The major disadvantage of this layered approach is that an extra data structure must be maintained over the DHT.

Mobile cloud computing: The approaches from MAUI [5] and Zhang et al. [11] seem promising because their model incorporates a cost model for deciding best execution configuration, and they can be also adapted dynamically according to real-time conditions. The approach in [12] is similar to above, but it lacks of dynamic adaptation of the computation between mobile devices and cloud services. Cloudlets [6] allows high abstraction and personalization of the computing environment by using VMs, but lack from fine-grained execution adaptation. [13] enables high horizontal scaling of the available ad-hoc mobile nodes, but with high communication overhead. These works are similar to VeSense when it comes to dynamic adaptation and scalable architecture. Existing work mostly focused on saving energy consumption on mobile devices using *serial* offloading; in contrast, our work provides a way to *parallel* offloading based on analytical cost models to minimize the energy. The parallel offloading is made by non-locking-based asynchronous remote procedure calls that can save lots of time and energy for some mobile applications as we demonstrate in Section 6. Note that some application can only support serial offloading due to the nature of procedural execution. They are different because VeSense does not require heavy modification in VMs but require light-weight instrumentation of source code of an application with a small amount of auxiliary code (less than 100 lines). Unlike existing work, VeSense can support multi-languages and cross-platforms with the help of Web technologies. Like other works, VeSense automatically partitions a program into both a local and remote execution part without human intervention. The capability of integration with the existing Web services such as Facebook, YouTube, and Twitter allows programmers to create more dynamic urban sensing applications.

Resource allocation in heterogeneous RATs: A resource allocation problem in a single RAT has been well studied. Several approaches including convex optimization [14], stochastic optimization [15], dynamic programming [16], and game theoretic approach [17], have been applied to such resource allocation problems. One of the frontiers studying heterogeneous RATs is [18]. Although Piao et al. [19] studied load balancing and resource allocation strategies in this direction, they did not consider the traffic characteristics and the quality of radio. We consider a utility maximization strategy which is similar to [20,21].

Network selection in heterogeneous RATs: Approaches to a network selection problem can be divided into three categories: network-centric, user-centric, and collaborative approach. The network-centric approach includes game theoretic formulation [22] and utility-driven approach [23], in order to select network that will optimize the network operator's profit. In the user-centric approach, the network is determined by using utility, cost, profit function, or MADM methods [24]. In collaborative approaches, the selection of the access network is accomplished by considering the profits of both users and the network operators. This approach is similar to the user-centric one in that they apply utility functions [25], and MADM methods [26].

3. System model

Going back to the network architecture of smartphone-based vehicular sensing in Fig. 1, we start this section by studying the current status of vehicular sensing applications. 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.

3.1. 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.

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001

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 $\mathbf{y} = [y_1, \ldots, y_i, \ldots, y_\theta]$, where y_i denotes it response variable for a single observation $i \in \hat{\Theta}$ and a Φ predictor vector is denoted by $x_i = [x_i^\phi, \ldots, x_i^\phi]$. The corresponding set of regression coefficients is expressed by a vector $\Gamma = [\gamma_0, \ldots, \gamma_\phi]$. Thus, a linear function of predictors Φ is given by,

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

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 $\mathrm{Var}(\epsilon_i) = \sigma^2$.

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.

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

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,

$$\hat{\epsilon} = y_i - \hat{\gamma}_1 - \sum_{j=1}^{\Phi} \hat{\gamma}_j x_i^j. \tag{3}$$

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

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

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 [27].

$$\hat{y}_i = E\left[\gamma_0 + \sum_{j=1}^{\Phi} \gamma_j x_i^j + \epsilon_i \right] = \gamma_0 + \sum_{j=1}^{\Phi} \gamma_j x_i^j. \tag{5}$$

The Eq. (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 Eq. (6).

$$\hat{P}_i^j = \eta \cdot \hat{y}_i^j, \tag{6}$$

where η can be experimentally obtained.

3.2. 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 [21,28], 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 focus on interference constrained RATs.

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001

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

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

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,

$$D_{m,b} = a\log_2(1 + c\nu_{m,b}). \tag{8}$$

We use positive constants a and c as system parameters such as bandwidth, modulation, and bit-error rates.

$$\tilde{D}_{m,b} = a\log_2\left(1 + c\frac{p_{m,b}}{\frac{\delta q_{m,b}\bar{P}_b + \sum_{b'\neq b} q_{m,b'}P_{b'} + \omega}{q_{m,b}} - \delta p_{m,b}}\right) \tag{9}$$

$$= a\log_2\left(a + c\frac{p_{m,b}}{\pi_{m,b} - \delta p_{m,b}}\right) \approx \left(\frac{\alpha}{\pi_{m,b}}\right)p_{m,b} \tag{10}$$

$$= \bar{D}_{m,b}p_{m,b}. \tag{11}$$

Note that the feasible data rate $D_{m,b}$ is not convex. In order to formulate a convex optimization problem [21], we further assume all base stations have a fixed transmission power, thus we can approximate a mobile user's data rate $\bar{D}_{m,b}$ in Eq. (11).

4. 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.

4.1. 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,

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

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'. The first part of minimization in Eq. (5) indicates the total cost of computation incurred from the program execution. The second part indicates the total amount of communication cost when offloading some of computation to the cloud. The first constraint limits that the offloading cost must be less than the local execution cost, thereby mobile devices save their energy. In practice, the communication cost as well as the cloud resource normally incurs the corresponding charge or usage fees. Once may address this issue by limiting his monetary budget as a constraint. For simplicity, we keep our optimization problem as simple as possible in this work. Furthermore, the calculation of communication cost incurred due to code offload is given by,

$$T_{v'} = n_{v'} \times D_{m,b}, \tag{13}$$

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001

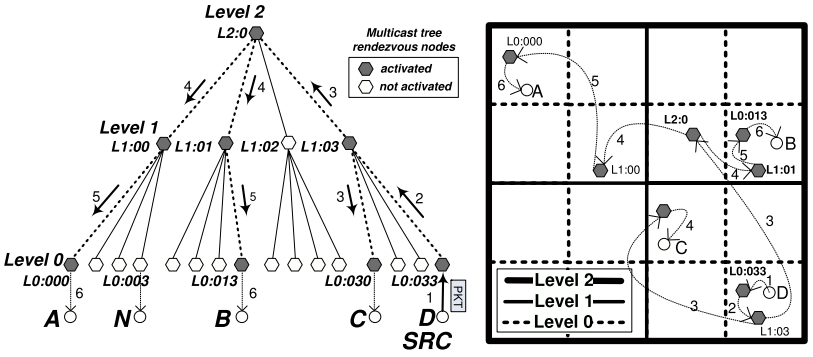

Fig. 2. Subscription-based multicast example: D (source) and A, B, C.

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 Eq. (8). We formulate further problems for how to assign the data rate to each mobile client in Section 4.2. The problem formulated in Eq. (13) consists of a concave objective over linear constraints, and it becomes convex. Therefore, there are various convex optimization algorithms to solve it from [29].

4.2. Network resource allocation problem

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

$$U = \sum_m \sum_b D_{m,b}. \tag{14}$$

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,

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

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

$$\begin{aligned} &\max\ U, \\ &\text{s.t.} \sum_m D_{m,b} \leq \sum_m \bar{D}_{m,b}, \\ &\sum_b D_{m,b} \geq D_{\min,b}, \\ &D_{m,b} \geq 0, \end{aligned} \tag{16}$$

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 U of all mobile users in all base stations as in Eq. (15). Note that Eq. (16) consists of a concave objective over linear constraints and thus is convex. That means there exists various algorithms to solve the problem [29].

5. Cloud-based vehicular sensing architecture

VeSense is a two-tier sensor networking platform that exploits the P2P-based Cloud servers similar to GeoServ [7]. Since most sensor data is generated on the roads (and most 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.

5.1. Location-aware sensor data retrieval service

We illustrate the Hilbert space filling curve, review routing semantics, present a detailed routing mechanism and its improvement techniques (e.g., delay and load balancing) and prove that the Hilbert curve based approach preserves content (geographic) locality.

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001

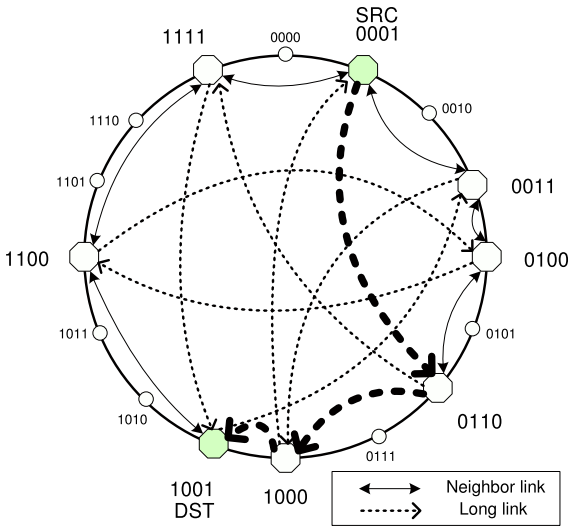

Fig. 3. Illustration of unicast routing: each node has neighbor links and one long link to a random location. Source located at 0001 sends a packet to the destination node located at 1001. It uses a long link to 0110 followed by neighbor links to 1000 and 1001 sequentially (thick dotted lines).

Routing semantics: 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. For example, assuming that the size of the contiguous US is approximated as 3000 km $\times$ 3000 km, it can be represented using $2^{13} \times 2^{13}$ fixed grids where R is given as 1 km. Given this 2D grid space, we use the Hilbert space filling curve [30], a linear mapping function where successive points are nearest neighbors in the 2D grid, the basic mapping is replicated in four quadrants. The lower left quadrant is rotated clockwise 90°, the lower right quadrant is rotated anti-clockwise 90°, and the sense (i.e., direction of traversal) of both lower quadrants is reversed. The two upper quadrants have no rotation and no change of sense. Thanks to the recursive construction above, the linear ID along the curve for any given grid point (x, y) can be easily calculated. The linear coordinate is augmented with two bits at a time for each recursion; i.e., the most significant bits (MSBs) of the x and y give the 2 MSBs of the resulting linear coordinate, along with the rotation and sense to be applied to the rest of the computation.

Routing semantics: The Hilbert curve enables GeoTable to map a 2D grid coordinate (x, y) to a D-bit numeric address on the Hilbert curve. Location-aware applications running on top of VeSense (or mobile users) can access sensor data generated from a remote region which can be a grid point, or multiple contiguous grid points denoted using a line/curve segment or a generic polygon formed by a set of line segments; e.g., apps want to fetch GPS readings originated from a set of road segments to calculate the average speeds in that area. Depending on how many overlay nodes are deployed and the size of a queried region, the region could be covered by a single overlay node, or by multiple overlay nodes. Thus, this routing strategy can be treated as *geocasting* (which is widely used in wireless mobile ad hoc networks) because destination nodes are implicitly set by specifying a target region—query packets are delivered to a group of overlay nodes that cover the region.

Geocasting to a single grid point: Since there is only a single overlay node that covers a given grid point, this can be seen as geographic *unicast* routing of a query packet. The unicast routing exactly follows the routing policy Symphony DHT [31] that uses Kleinberg's Small World phenomenon. For completeness, we present Symphony DHT. In Symphony, a node joins the network by picking up a random ID (equivalent to a numeric grid ID on the Hilbert curve). Every node maintains two short links to one's 1-hop neighbors and $k \geq 1$ long distance links. Long distance links are constructed as follows. Consider a node whose ID is n and is responsible for the range $[\ell, r]$. Let I denote the space of D-bit Hilbert curve, $[0, 2^D)$. For each link, a node draws a number $x \in I$ based on the harmonic probability distribution function: $p_n(x) = 1/(n \log x)$ if $x \in [2^D/n, 2^D)$. Kleinberg showed that such a construction allows us to greedily route packets to a random node (i.e., in each hop, follow a long link that is closest to the destination) in $O(\log^2 n)$ hops on average [31]. Fig. 3 shows an example. Readers can find the details of join/leave functions in [31].

Geocasting to multiple grid points: 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 loses 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.

Suppose we have two segments, namely $\{[0001 - 0010], [1101 - 1110]\}$. Given this, geocasting is straightforward. For a given ordered list, a packet is first routed to the head of the first segment (e.g., 0001) using the aforementioned unicast routing scheme. By following the neighbor links, the first segment is scanned. Since an overlay node typically covers a span of key space, this is simply local scanning. After this, the query packet will be forwarded to the head of the next segment and another scan will be performed. This process repeats until we cover all the segments in the list.

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001

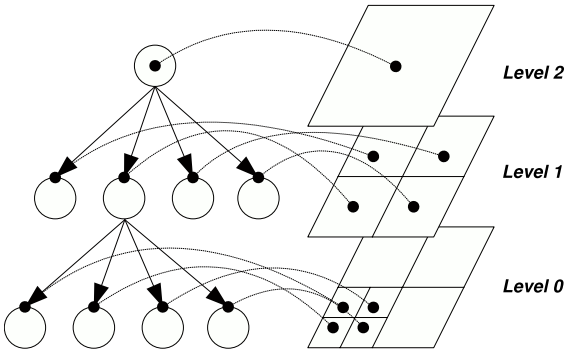

Fig. 4. Illustration of a hierarchical geographic location service.

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 [7]. GeoTable uses Mercury's load balancing mechanism to preserve locality of content retrieval [32].

5.2. 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. In this section, we present the support for *subscription queries* of multiple users who are interested in data updates on a target region: e.g. traffic information on the commute route. 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 [7].

GeoPS overview: Given that majority of data consumers of location-sensitive data will be located near the area where the data are generated (e.g., traffic information on the commute route), the key design issue is to build a multicast tree that exploits the geographic locality of the group members.[2] Our approach called GeoPS is inspired by hierarchical geographic location services (HGLS) in mobile ad hoc networks such as GLS [33] and HIGH-GRADE [34] where the entire area is recursively divided into a hierarchy of smaller grids, and mobile users' current locations are efficiently tracked under the geographic hierarchy. The key idea of GeoPS is to build a multicast tree over this geographic hierarchy and to use our geocasting algorithm over the tree to preserve geographic locality.[3] This is a major departure from existing DHT-based multicast solutions (e.g., Bayeux, Scribe) that *destroy locality* using consistent hashing and randomly distribute geographically correlated subscribers across the entire key space.

Review of HGLS: In mobile ad hoc networks, a location service keeps track of mobile nodes' current locations and lets mobile nodes to query the current location of an arbitrary node (e.g., to use it for geographic routing). In HGLS, a geographic hierarchy tree is constructed by recursively dividing the entire area into a hierarchy of smaller grids. Fig. 4 shows an example where the root of a tree covers the entire network area (level 2), and each of its children covers a sub-region whose size is one fourth of the network area (level 1). For each level i, nodes have a pair of common hash functions $h_{i,x}(id)$ and $h_{i,y}(id)$ that map a node ID to a geographic coordinate (x, y) at level i. For a given node whose ID is ℓ, one node located around the location $(h_{i,x}(\ell), h_{i,y}(\ell))$ is chosen as node ℓ's *location server* at level i. The node ℓ publishes its current location to the leaf region (level 0 area where the node is currently located), and all its upper level location servers along the *single path* of the geographic hierarchy tree are initialized as rendezvous points. Note that up-to-date location information is stored locally (at level 0 servers where the node is currently located), and rendezvous points are updated only when the node crosses the level boundary.

Given this, any node can send a location query for the node ℓ as follows. The query is first routed to location servers around $(h_{0,x}(\ell), h_{0,y}(\ell))$ in the level 0 area where the querying node is located. If the level 0 location servers do not have the information, the query is routed to the level 1 location servers for node ℓ that are located around $(h_{1,x}(\ell), h_{1,y}(\ell))$. The process is repeated until it finds the location servers at level i that have the path information (i.e., rendezvous point). The query then traverses down the hierarchy to find the exact location available at the level 0 location servers. In Fig. 2, node A's current location is stored in node L0:000, and we have two rendezvous points at Level 1 (L1:00) and Level 2 (L2:0). Node D can find node A's location as follows. It queries node D's Level 0 server (L0:033), but it fails to find the information. It tries

[2] Note that if all subscribers are originated from a single region, we can easily implement the service using geocasting. In this section, we focus on more general scenarios where subscribers are from a set of non-contiguous regions.

[3] We can easily prove that multicast routing is localized as we use our geocasting algorithm presented in the previous section (using similar proofs used in Theorem 1 and in Yu et al. [34]).

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001

Level 1 server (L1:03), fails, and finally finds a rendezvous point at Level 2 (L2:0). By following the links along the rendezvous points, we can find node A's current location at node A's Level 0 server (L0:000).

Multicast tree construction: In GeoPS, 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 Fig. 2, when mobile user A joins, the subscription message is installed at L0:000, L1:00, and L2:0 sequentially. We repeat the same process when user B, C, D join, and Fig. 2 shows the resulting multicast tree (dark gray nodes have the subscription entry). Now, when a new mobile user N joins, its subscription message will be installed at L0:003, and it will then be forwarded to L1:00. This level 1 node finds that there is an existing subscription entry set by mobile user A, and the subscription message stops propagating.

The leave process is similar to the join process. When a mobile node gracefully leaves the system, it sends a leave message to upper levels to remove the subscription information. In each level, if there is no more subscription entry for a given group, the message is sent to the upper levels sequentially. For mobility handing, we follow the same scheme detailed in GeoServ [7].

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 Fig. 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.

Data update publish: A source can send a message along the tree starting from the leaf node (Level 0) and traversing toward the upper levels. When there is a matching subscription in an intermediate node, it sends the message to each child in the subscription entry from which the packet starts traversing down the tree. Fig. 2 shows an example. We have four members (mobile clients): A, B, C, and D. Source D sends the packet to L1:03 (step 1). L1:03 sends it to both L0:030 and L2:0 (steps 2 and 3). After this, L2:0 sends it to L1:00 and L1:01 (step 4). L1:00 and L1:01 send it to L0:000 and L0:013 respectively (step 5). They deliver the packet to A and B (step 6).

Minimum depth configuration: In practice, the number of overlay nodes is much less than the total number of grids (i.e., entire key space). Thus, the lowest depth should be configured as L_{M-K} rather than naught (where M is the maximum level, and K is the depth of a multicast tree) such that there is at least one overlay node in that region; otherwise, we are storing redundant rendezvous points (in sub-trees below the lowest level) to the same overlay node. Note that we can configure the depth in a distributed manner by utilizing the node distribution histogram and network size estimates from GeoTable's load balancing. Our prior study in [7] proves that we have $K = O(\log n)$ under uniform node distribution.

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 Fig. 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 Fig. 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.

6. 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

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001

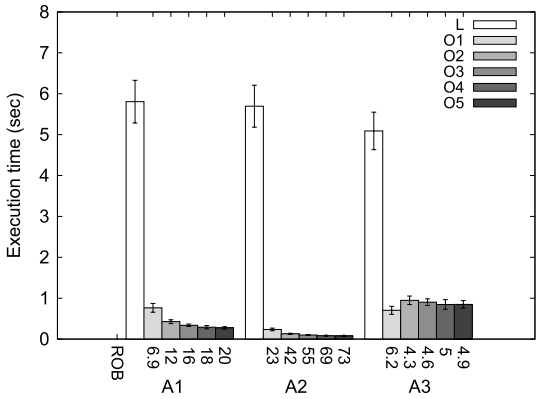

Fig. 5. The average execution time and relative offloading benefit (ROB) of three vehicular sensing applications with various offloading scenarios are measured and presented with 95% CIs.

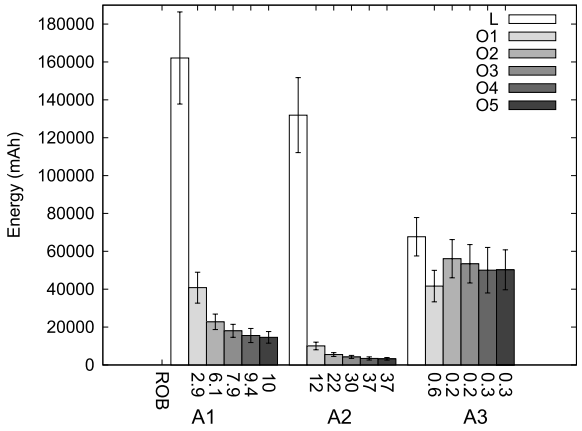

Fig. 6. The average energy consumption and relative offloading benefit (ROB) of three vehicular sensing applications with various offloading scenarios are measured and presented with 95% CIs.

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 [35]. For simplicity, interface constrained RATs such UMTS are only considered. Unless specified, network parameters are given form [28].

Fig. 5 compares the execution time between the mobile and the cloud by applying to three different vehicular sensing applications: A1–A3, while Fig. 6 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 [36]. 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 Fig. 2. Fig. 7 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.

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001

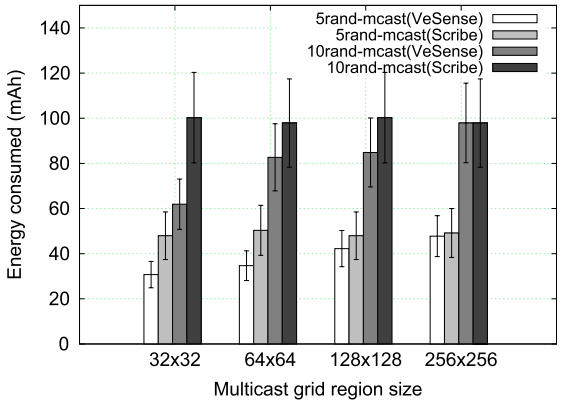

Fig. 7. Subscription-based multicast routing comparison in energy consumption: VeSense vs. Scribe.

7. 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. For the future work, we plan to study energy issues in resource allocation of cloud services. Particularly, the cooling down machines in data center has been a major research topic. By modeling air flow in data center, we may generate a better power management and job scheduling strategy.

References

[1] USA to Add 80 Million New Smartphone Users by 2011. http://twittown.com/mobile/mobile-blog/usa-add-80-million-new-smartphone-users-2011.
[2] S. Reddy, G. Chen, B. Fulkerson, S.J. Kim, U. Park, N. Yau, J. Cho, Sensor-Internet share and search, in: DSI, 2007, pp. 11–16.
[3] S. Nath, J. Liu, F. Zhao, SensorMap for wide-area sensor webs, IEEE Comput. Mag. 40 (7) (2007) 90–93.
[4] P.B. Gibbons, B. Karp, Y. Ke, S. Nath, IrisNet: an architecture for a worldwide sensor web, IEEE Pervasive Comput. 2 (4) (2003) 22–33.
[5] E. Cuervoy, A. Balasubramanianz, D.-K. Cho, A. Wolmanx, S. Saroiux, MAUI: making smartphones last longer with offload, in: MobiSys, 2010, pp. 49–62.
[6] M. Satyanarayanan, P. Bahl, R. Cáceres, N. Davies, The case for VM-based cloudlets in mobile computing, IEEE Pervasive Comput. 8 (4) (2009) 14–23.
[7] J.H. Ahnn, U. Lee, H.J. Moon, GeoServ: a distributed urban sensing platform, in: CCGRID, 2011, pp. 164–173.
[8] A. Rowstron, P. Druschel, Storage management and caching in PAST, a large-scale persistent peer-to-peer storage utility, in: SOSP, 2001, pp. 188–201.
[9] Y. Chawathe, S. Ramabhadran, S. Ratnasamy, A. LaMarca, A case study in building layered DHT applications (PHT), in: SIGCOMM, 2005, pp. 97–108.
[10] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, PEIR, the personal environmental impact report, as a platform for participatory sensing systems research, in: MobiSys, 2009, pp. 55–68.
[11] X. Zhang, S. Jeong, A. Kunjithapatham, Simon Gibbs, Towards an elastic application model for augmenting computing capabilities of mobile platforms, in: Mobileware, 2010, pp. 161–174.
[12] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, G. Alonso, Calling the cloud: enabling mobile phones as interfaces to cloud applications, in: Middleware, 2009, pp. 83–102.
[13] G. Huerta-Canepa, D. Lee, A virtual cloud computing provider for mobile devices, in: MCS, 2010, pp. 1–5.
[14] D.P. Bertsekas, Nonlinear Programming, second ed., Athena Scientific, Belmont, Massachusetts, 1995.
[15] H. Kushner, G. Yin, Stochastic Approximation and Recursive Algorithms and Applications, second ed., Springer, 2003.
[16] D.P. Bertsekas, Dynamic Programming and Optimal Control, Vol. I, Athena Scientific, Massatchusetts, 1995.
[17] D. Fudenberg, J. Tirole, Game Theory, MIT Press, 1991.
[18] 3GPP system architecture evolution: report on technical options and conclusions (release 7), Tech. Report TR 23.882, Third Generation Partnership Project, 2008.
[19] G. Piao, K. David, I. Karla, R. Sigle, Performance of distributed MxRRM, in: IEEE PIMRC, 2006, pp. 1–5.
[20] M.J. Neely, E. Modiano, Chih-Ping Li, Fairness and optimal stochastic control for heterogeneous networks, IEEE/ACM Trans. Netw. 16 (2) (2008) 396–409.
[21] I. Blau, G. Wunder, I. Karla, R. Sigle, Decentralized Utility Maximization in Heterogeneous Multicell Scenario, in: PIMRC, 2008, pp. 1–6.
[22] K. Piamrat, A. Ksentini, J.-M. Bonnin, C. Viho, Radio resource management in emerging heterogeneous wireless networks, Int. J. Comput. Netw. 34 (9) (2010) 1066–1076.
[23] Q.-T. Nguyen-Vuong, N. Agoulime, Y. Chamri-Doudane, A user-centric and context-aware solution to interface management and access network selection in heterogeneous wireless environments, Int. J. Comput. Netw. 52 (18) (2008) 3358–3372.
[24] O. Ormond, G.M. Muntean, J. Murphy, Economic model for cost effective network selection strategy in service oriented heterogeneous wireless network environment, in: IEEE NOMS, 2006, pp. 1–4.
[25] W. Luo, E. Bodanese, Radio access network selection in a heterogeneous communication environment, in: WCNC, 2009, pp. 1–6.
[26] B. Nacef, N. Montavont, A generic end-host mechanism for path selection and flow distribution, in: PIMRC, 2008, pp. 1–5.
[27] F. Harrell, Regression Modeling Strategies, Springer, 2001. 2.
[28] J. Buhler, G. Wunder, An optimization framework for heterogeneous access management, in: WCNC, 2009, pp. 1–6.
[29] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, NY, USA, 2004.
[30] H.V. Jagadish, Linear clustering of objects with multiple attributes, SIGMOD, 1990, pp. 332–342.
[31] G.S. Manku, M. Bawa, P. Raghavan, Symphony: distributed hashing in a small world, in: USITS, vol. 10, 2003, p. 10.
[32] A.R. Bharambe, M. Agrawal, S. Seshan, Mercury: supporting scalable multi-attribute range queries, in: SIGCOMM, 2004, pp. 353–366.
[33] J. Li, J. Jannotti, D.S.J. De Couto, D.R. Karger, R. Morris, A scalable location service for geographic ad hoc routing, in: MobiCom, 2000, pp. 120–130.
[34] Y. Yu, G.-H. Lu, Z.-L. Zhang, Enhancing location service scalability with HIGH-GRADE, in: MASS, 2004, pp. 164–173.
[35] Pavement Interactive Core: Roughness. http://pavementinteractive.org/index.php?title=Roughness.
[36] M. Castro, P. Druschel, A.-M. Kermarrec, A. Rowstron, Scribe: a large-scale and decentralised application-level multicast, IEEE J. Sel. Areas Commun. 20 (8) (2002) 1489–1499.

Please cite this article in press as: J.H. Ahnn, M. Potkonjak, VeSense: High-performance and energy-efficient vehicular sensing platform, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.05.001