Mobile Stream Sampling under Time Constraints

Ioannis Boutsis
Department of Informatics
Athens University of Economics and Business
Athens, Greece
mpoutsis@aueb.gr

Vana Kalogeraki
Department of Informatics
Athens University of Economics and Business
Athens, Greece
vana@aueb.gr

***Abstract*—The proliferation of mobile networking and the increasing capabilities of smartphone devices in the recent years have resulted in transforming mobile smartphone devices into ubiquitous sensing platforms. In this new class of "Community-based Participatory Sensing" systems, users actively participate in the data collection and sharing for the benefit of the community, in a wide range of application areas from entertainment, to transportation, to environmental monitoring. These approaches, however, generate large amounts of transient data streams, leading to real-time computational challenges. In this paper we propose sampling algorithms on streams of mobile data generated by ubiquitous sensing devices that need to be processed under time constraints. In our approach users participate in the system by sensing and sharing streams of data. The system then uses a sampling mechanism to select a subset of data streams that preserves the characteristics of the stream data and provides the highest "information gain" to the system, given the real-time, budget and resource constraints. Detailed experimental results illustrate that our approach is practical, efficient and depicts good performance.**

I. Introduction

The proliferation of location-aware smartphone devices (iPhones, GPS, PDAs, etc.) in the recent years have made it possible to monitor and track the movement of humans in order to capture and understand social dynamics or identify events of interest for the community. For example, in traffic monitoring systems such as MetroSense [1] and VTrack [2], users equipped with smartphones outfitted with a wide array of sensing capabilities such as GPS, WiFi, microphones, cameras and accelerometers provide real-time traffic data information in the form of travel times or vehicle trajectories. By combining traffic data streams from different users, important information and events of interest can be extracted *i.e.,* traffic congestion; these can be used in a variety of ways, for example, informing drivers and making suggestions for alternative routes to reach a destination faster. Similar examples can be found in earthquake warning detection systems [3] and location-based services such as personalized weather information and identifying areas of good WiFi connectivity [4].

These approaches, however, invite a wide variety of users to participate, and can thus generate large amounts of transient data streams. We argue that in such a system, not all data streams are equally important. The "true value" of the user data depends on various factors, including the characteristics of the sensing data, the user context (*e.g.,* geographic location), available resources (*e.g.,* communication, power), the density of the area where data is extracted, the elapsed time from the previous measurements, etc. Thus, a fundamental question is how to choose a suitable representative *sample* from the stream data that preserves the characteristics of the data, while saving in processing and communication costs when obtaining the data and satisfy any application constraints.

The problem of sampling is challenging due to the following aspects that should be taken into consideration: (i) Efficiency: we need a method to efficiently represent the stream data that preserves the properties of the spatio-temporal information, with minimum errors; (ii) Real-time processing: the sampling technique must be able to process mobile stream data under real-time and resource constraints; (iii) Complexity: the technique needs to be able to cope with very large data streams generated from mobile phones. The amount of data that the system is capable of processing is constrained by two major factors: First, the resource availability across the distributed system that will collect and process the stream data. Second, the cost for the application to receive the data. There is a tradeoff between the approximation quality, and time and space complexity. Furthermore, we expect that users may wish to receive some compensation to participate in such a system, so we need to reward them for the data they provide. Hence, the system has to consider the number of data streams that can be supported by considering both the available resources and the corresponding budget paid to the users. These constraints make the sampling process even more challenging, since we have to select the most representative subset of data streams, that would not exceed the system's resource constraints, considering the respective cost as well. Although we have witnessed commercial participatory systems where users provide their data with no compensation, *e.g.,* to use the service such as the traffic data in Google Maps [5], we expect that this would not be the case in other non-popular services where participation is needed. For instance, users would not willingly participate in an environmental monitoring application where their local data are sensed and sent for analysis or in applications where human effort is needed such as taking a picture of polluted places. However, the accuracy of the results in such applications depend on user participation. Hence, we consider that users may receive a monetary reward for the services they provide and we expect that the monetary compensation provided could attract many users for participatory systems.

In this paper we address the problem of sampling streams of mobile data generated by ubiquitous sensing devices under time constraints. We consider this as an important capability for participatory sensing applications which are rapidly growing in popularity. Specifically, in our approach users participate in the community by sensing and sharing streams of data.

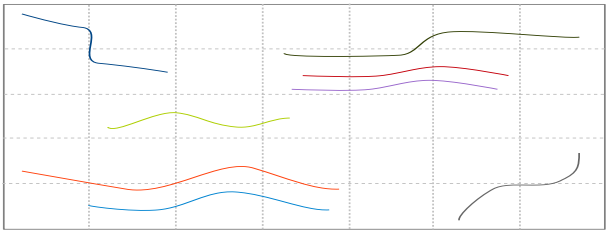

Fig. 1. Trajectories of the mobiles.

They define their offers and they receive a monetary reward for the stream data they provide. The offers reflect the quality of the data provided along with the corresponding cost for providing this data. The system aims to select the offers that preserve the characteristics of the stream data while providing the highest "information gain" to the system, given the real-time resource and budget constraints. The problem of selection can be reduced to the well-known Knapsack problem and thus we introduce greedy sampling mechanisms to select the subset of data streams. We perform a study of different sampling mechanisms that belong to the family of Budget Feasible Mechanisms to examine the effect of information quality on the data samples. The sampling technique works cooperatively with our rate allocation mechanism. The goal of the rate allocation mechanism is to determine the maximum amount of stream data that can be processed in the system based on the timeliness constraints (*i.e.,* deadlines) of the applications and the system resource availability. We use an adaptive rate allocation technique that uses measurements of elapsed times, application projected latencies and measurements of resource availability to dynamically determine the rate allocation for the stream processing applications to meet timeliness and rate demands. We have implemented a real-time traffic monitoring application in our system using the Berkeley Mobile Millennium dataset [6] and present detailed experimental results on PlanetLab. Our experimental results illustrate that our approach is practical, efficient and depicts good performance.

II. System Model

We assume a spatial decomposition of the geographic area into a number of non-overlapping regions, that enables the system to perform localized processing based on the stream data at each region. The organization of the area into regions can be done with respect to the size of the network, possibly defining several tiers at different levels of granularity, ranging from small local areas at the lowest tier, to the entire network area at the highest tier; this allows the system to collect measurements from all mobile phones in a scalable manner[7].

Let S denote the mobile sensor network of m mobile devices that monitor and report measurements from the observed phenomena. Each mobile device generates *data streams*; each data stream consists of sequences of individual chunks of data, referred as *Application Data Units* (ADUs). Data streams represent short messages that are triggered locally at the phone using sensing devices present on mobile phones such as microphone, camera, GPS, accelerometer and motion sensors; the size of a data unit depends on the type of the application.

Each mobile device i moving in the Euclidean space is characterized by the following set of information $L_i(x_i, y_i, \text{ADU}_i)(t)$, $i \in 1, ..., m$, for every time instance t, where the pair (x_i, y_i) corresponds to the position of the device (latitude, longitude), and ADU_i is the data stream generated. Examples of data streams are: <latitude, longitude, video data, timestamp> (for surveillance monitoring applications) or <latitude, longitude, accelerometer data, timestamp> (for earthquake monitoring). We consider that data streams are triggered concurrently and independently from different smartphones, as they are sensed or observed by the application components. Data streams from multiple mobile sensors are streamed into the distributed stream processing system for further processing based on the application logic. Although smartphones are powerful enough to do some local processing instead of sending the raw data streams, for this paper, without any loss of generality, we consider that only raw data are sent.

Mobile users express their willingness to participate and share their sensed data, by presenting their offers. Each user i defines its $\text{offer}_i(t)$ as a pair of values: (i) $\text{QoI}_i(t)$, and (ii) $\text{cost}_i(t)$ for its produced $\text{ADU}_i(t)$ at time t. The offer expresses the willingness of the user to have its sensed data shared with the community and reflects the importance of the specific stream data generated by the mobile node i to the application, along with the cost to the system to receive that stream. Both values should be transmitted when a mobile node wishes to participate in the sampling process. The $\text{QoI}_i(t)$ value is computed based on application-defined functions, as defined in Section III. The cost can be expressed in terms of a payment that the user wishes to receive and could be (i) a fixed price payment provided by the system, (ii) provider-defined (*e.g.,* the user may be charged for the data he/she uploads in the system by its wireless provider), (iii) user-defined (*e.g.,* users may also want to charge an amount for the energy consumption).

In our system we assume that all users are cooperative, that there are no malicious users and that users submit their "true" $\text{QoI}_i(t)$ information and get compensated for the information they provide.

Each application q is characterized by a Deadline_q, a relative metric that represents the end-to-end time constraint required for the application q to process a number of ADUs. Hence, the rate of the application R_q is defined as the amount of data units that the system processes within its Deadline_q. Multiple sampling operations can be carried out with different deadlines.

III. Our Approach

In this section we define our optimization problem, discuss our sampling techniques and our rate allocation technique.

A. Optimization Problem

Problem *Consider regions $R_1, R_2, .., R_k$ of a given geographical space where M mobile nodes are located. Each mobile node i produces streams of events (ADUs) over time t with an information quality $QoI_i(t)$; this reflects the relative utility (importance) of the stream data provided by the mobile phone i to the system, and a cost $cost_i(t)$. The system selects multiple data streams based on its $Budget$, so that the information quality provided to the system is maximized. The selection of the streams should also consider the resource constraints,*

imposed by the system resources, on the Rate$_q$ that the system can support so that the application q can meet its relative Deadline$_q$, determined by the rate allocation mechanism.

Thus, our goal is to determine those ADUs, that maximize the information benefit, subject to the constraints. This is expressed as follows:

$$\text{Maximize} \sum_{i \in M} \text{Information}_i(t)$$
$$\text{subject to} \sum_{i \in M} \text{cost}_i(t) \leq Budget$$
$$\sum_{i \in M} \text{ADU}_i(t)/\text{Deadline}_q \leq \text{Rate}_q$$

Our sampling technique aims to solve this constrained maximization problem at run-time, in order to maximize the system's information profit by selecting the most important streams, subject to the system's constraints. The factor $\sum_{i \in M} \text{Information}_i(t)$ is general and refers to the information benefit from the data streams that were selected. However, the maximization problem can provide different solutions. One possible solution would be to set $\sum_{i \in M} \text{Information}_i(t)$ as $\sum_{i \in M} \text{QoI}_i(t)$ or another solution could aim to maximize the number of ADUs, but with a lower individual $\text{QoI}_i(t)$ and $\text{cost}_i(t)$. Hence, we propose different strategies to solve our maximization problem and compare their performance.

The fundamental idea of our approach is to segment the trajectory of each mobile device into piece-wise linear parts, based on the spatial characteristics of the trajectory. This segmentation is static and can be determined based on the regions defined in the geographic area. The intuition is that when there are multiple segments within a given geographical region, the system can assign different costs per segment based on the availability of the segments. Thus, we can express it as an optimization problem which trades off the quality of the information, with resource cost and time complexity. As we aim at sampling trajectories, another intuitive representation would be to segment the trajectories along the temporal dimension. However, we argue that such a representation would result in higher complexity as each trajectory would be modeled with possibly different number of time periods of varying duration and we would then have to compare the different segments in order to identify the corresponding geographical regions.

We implement a sampling component (discussed in section III.B) at each mobile node responsible for obtaining the paired offers and decide which data streams will be sampled. Moreover, this component works in concert with our rate allocation mechanism (discussed in section III.C) which determines the maximum amount of stream data, that the system can efficiently process, depending on resource availability. Suppose there are m_i mobile nodes in region R_j. The goal of the sampling mechanism is to select a subset of data streams s_i, (where $s_i < m_i$) to be processed.

Sampling Error. We compute the sampling error to evaluate our approach, similar to [8], as follows. In order to estimate the average value of the m_i data streams generated in a specific time period in region R_j, we sample $\bar{s_i}$ ($\bar{s_i} < m_i$) records from it ($x'_1, x'_2, .., x'_{\bar{s_i}}$). The average of the sampled records $y'_i (y'_i = \sum_{j=1}^{\bar{s_i}} x'_j/\bar{s_i})$ can be considered as the estimate of the exact average (y_i) of all the m_i records. Hence, we define the sampling error metric Δ_i as the expected absolute difference between the estimate average and the exact average:
Sampling Error: ${\Delta_i}^2 = E[(y'_i - y_i)^2]$

Moreover, we define the percentage variation of the estimate average compared to the exact average:
Sampling Error Percentage: ${\Delta_i}^2 = E[(\frac{(y'_i - y_i)}{y_i})^2]$

Our goal is to maximize the $\sum_{i \in M} \text{Information}_i(t)$ rather than minimizing the error metric, although these two formulations are not always proportional. Consider for example that $\text{Information}_i(t)$ depends on a utility function where each spatial region has a specific weight. We can infer that maximizing the $\sum_{i \in M} \text{Information}_i(t)$ would decrease the Sampling Error in some regions with high weights, while increasing it in others and thus the ${\Delta_i}^2$ would not be absolutely minimized. However, the application's target will be met and the error would be relatively minimized with respect to the defined objectives.

Information Quality. We assume that each stream of data has a "value" expressed as a function of time. Two fundamental questions arise when defining the utility of the information that a data stream has for the system. First, not all stream data have the same utility (*i.e.,* importance). Typically, some data streams may have higher utility than others and this relative utility may change dynamically at run-time. Second, the relative utility of the stream data might not be directly related to the time deadline with which the stream data needs to be delivered to the system. Our goal is to consider these two parameters of information utility and application time deadline in concert, when determining the appropriate data streams to be sampled.

Utility functions. We use utility functions (that we call *QoI functions*) to express the benefit of providing a stream of data to the application. Although our proposed QoI functions are mainly linear functions over time, in the general form the utility functions might have different forms and shapes to be able to meet the demands of each application [9] [10]. These functions form the foundation of our sampling approach that aims to maximize the information benefit of the system. In our previous work we have shown that the selection based on the QoI functions can improve the accuracy of the results compared to a uniform selection [11]. Moreover, as will be presented in the experimental section, combining several utility functions to compute $\text{QoI}_i(t)$, further increases the accuracy.

In the experimental evaluation we investigate the use of different QoI functions, that, in their general form are expressed as: $\text{QoI}_i(t) = \text{QoI}_i(t-1) + \sum \text{CurrentValues}_i(t)$.

In every run of the sampling mechanism, the $\text{QoI}_i(t)$ value for a mobile node i is adjusted based on the $\text{CurrentValues}_i(t)$, that represents the additional benefit in the specific time interval, computed by the utility functions. These functions are predefined, application-specific, based on the application logic and are used by each mobile device independently at run-time, to compute its $\text{QoI}_i(t)$ value at each time instance t. Since the $\text{QoI}_i(t)$ value is adjusted during the runs of the sampling mechanism, based on the application-specific functions, the chances that a mobile node is selected for sampling increase as its $\text{QoI}_i(t)$ value increases.

We present examples of different utility functions in Figures 2, 3 and 4. Figure 2 represents a utility function where

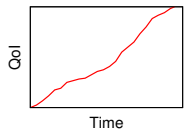

Fig. 2. Density.

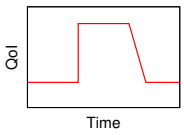

Fig. 3. Transition.

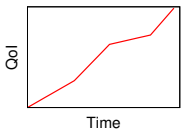

Fig. 4. Region.

the QoI value increases over time. The step increase of this utility depends on the current state and the measured values. Figure 3 represents the case of a utility function where its value increases when a data stream occurs and then decreases as the data stream becomes distant in time. This is the case, where for example, the utility function represents the event of congestion in a traffic monitoring application; receiving this data stream has high importance for the system, while when the event is reported by multiple individuals its value decreases. Finally, figure 4 is similar to 2, however it has a different slope. This utility function represents, for example, the case where a mobile moves through geographic regions. While it moves in the same region, the incremental step remains the same, however changing region results in a different step increase.

B. Sampling

Our sampling technique is based on a family of mechanisms called **Budget Feasible Mechanisms**, initiated by Singer [12]. The basic idea is to select the optimal offers, that provide the highest information utility, where each offer has a $\text{QoI}_i(t)$ and a respective $\text{cost}_i(t)$, without exceeding the total $Budget$ or resources. It has been shown that determining the optimal offers, where each offers has a utility with a respective cost and the total Budget cannot be exceeded, is an NP-hard problem, as it can be reduced to the well-known Knapsack problem. The Knapsack problem states that given a Budget B and a set of items $N = \{1, .., n\}$ each with a cost c_e and a utility u_e, find a subset of items S that maximizes $\sum_{i \in S} u_i$ under the budget constraint. Hence, our problem is similar to the Knapsack problem, although we have additional constraints that need to be satisfied. The family of Budget Feasible Mechanisms, can be solved with a number of approximation mechanisms, that need to depict specific behavior [13]. Thus, these mechanisms need to be: (i) *Truthful*: sellers are assumed to report the true cost for their offers, (ii) *Computational Efficient*: the algorithms must be computed in polynomial time, (iii) *Budget Feasible*: the mechanism's rule for determining offers should not exceed the budget, and (iv) *Approximation*: the determined subset, from the set of bids, must yield the highest possible value for the buyer. However, in our approach we provide additional real-time constraints, to ensure that the amount of the offers would not exceed the available resources of the system. We import these constraints without affecting the complexity or the four behavioral characteristics that the mechanism depicts.

In our approach all nodes that wish to participate in the sampling submit their offers ($\text{QoI}_i(t)$, $\text{cost}_i(t)$) to the system. Our adaptive rate allocation technique informs the sampling components about the amount of ADUs that the system can process in the next time window (Rate_q) to meet the application Deadline_q. Thus, the sampling component will be able to select the nodes with the highest offers, based on the chosen sampling strategy, and will inform them to transmit their ADUs. At the end of the sampling process, the smallest QoI value from the set of the selected offers (that we call $Threshold$) is identified. This value is returned to the nodes that have not been selected for sampling, so that in the next run they can choose not to submit their offers, if their $\text{QoI}_i(t)$ value is less than the $Threshold$. As we will show in the experimental evaluation, this simple boundary, results to an important efficiency improvement.

Our approach aims to receive data streams that provide the maximum information profit. However, each ADU transmitted has a corresponding $\text{cost}_i(t)$. Thus, we define the *Information_Gain* value for each data stream generated at time t by mobile phone i, as: $\text{Information\_Gain}_i(t) = \frac{\text{QoI}_i(t)}{\text{cost}_i(t)}$

The Information_Gain is defined to be proportional to $\text{QoI}_i(t)$ and depends on the $\text{cost}_i(t)$ value. Essentially, it provides a metric to evaluate the data units based on their "value for money".

Algorithm Operation. Below we summarize the steps of our algorithm:

- At each time unit t the mobile nodes i calculate their (application-specific) $\text{QoI}_i(t)$ based on local knowledge and information obtained by the application logic.
- Each mobile node i defines its $\text{offer}_i(t)$: the $\text{QoI}_i(t)$ value and the respective $\text{cost}_i(t)$ that depicts the profit he/she estimates for the data units provided.
- If the $\text{QoI}_i(t)$ value is greater than the $Threshold$, then the mobile node makes an offer on the sampling component.
- The sampling component is aware of the system's ADU processing ability (Rate_q) for the given time window and $Budget$, and selects as winners the set of k mobile phones with the highest offers, that fulfill these constraints, *according to the strategy used for sampling* of the data units (discussed next).
- The winners are informed to send their $\text{ADU}_i(t)$, get paid with the corresponding $\text{cost}_i(t)$ and set their $\text{QoI}_i(t-1)$ to zero.
- All the other mobile users that participated in the sampling process receive the minimum winning price (smallest $\text{QoI}_i(t)$ value from the set of the winning offers), in order to set this value as their $Threshold$ in the next run.

Sampling Strategies. We investigate three sampling strategies. First, we propose a basic mechanism where the cost of each ADU equals a fixed price called ISAM. Then we present an extension to that mechanism which is the more realistic algorithms ISAM+ and ISAM++ where the cost depends on user-defined prices. All the mechanisms rely on the Rate Allocation scheme (given in section III.C) in order to define the number of winners at runtime. Since an optimal solution is costly to be provided in real-time, as explained above, we provide greedy algorithms that aim to provide a good approximation. In all the following strategies, we define as A the total set of offers submitted for the selection of the sample by the users.

ISAM is the base algorithm: the mobile nodes selected from the sampling component, receive the same, predefined value as their payment. This mechanism is inspired by the First-Price Sealed-bid Auction, where bidders submit one bid in a concealed fashion and the highest submitted bids win the auction. Thus, in this algorithm we try to maximize the $Information\_Gain_i(t)$ when the $\text{cost}_i(t)$ is a **fixed value** for each offer, determined by the system and denoted as K. Since the system pays the selected nodes with the same amount of money, regardless of their QoI profit, ISAM is going to choose these mobile nodes that provide the highest $\sum_{i \in M} \text{QoI}_i(t)$ value. Thus, the maximization problem becomes:

$$\begin{aligned} \text{Maximize} & \sum_{i \in M} \text{QoI}_i(t) \\ \text{subject to} & \sum_{i \in M} \text{cost}_i(t) \leq Budget \\ & \sum_{i \in M} \text{ADU}_i(t)/\text{Deadline}_q \leq \text{Rate}_q \end{aligned}$$

This Sampling mechanism, works as follows:

1) *Order all offers in set A s.t. $QoI_1(t) \geq QoI_2(t) .. \geq QoI_{|A|}(t)$*
2) *Let Winner set $S = \emptyset$ and $k = 1$;*
3) *Set $cost_i(t) = K, \forall i \in A$*
4) *While $\big(k \leq |A|$ and $\frac{|S|+1}{Deadline_q} \leq Rate_q$ && $cost_k(t) \leq Budget * (QoI_k(t)/\sum_{i \in S \cup \{k\}} QoI_i(t))\big)\{$ $S \leftarrow S \cup \{k\}$; $k \leftarrow k + 1$; $\}$*
5) *Return set S;*

Worst-Case Complexity: Our approach needs to sort the bids which takes $O(n \log n)$, meaning $O(|A| \log |A|)$. The while-clause will repeat at most $|A|$ times, and the assignment statements inside the while-clause cost $O(1)$, so it costs $O(|A|)$. Thus, the worst-case complexity of our algorithm is $O(|A| \log |A|)$.

ISAM, is our basic algorithm. However, in a real system, different users may have different needs(i.e. different 3G cost). Thus, the following two algorithms where users can define their payments seem as a more realistic approach.

ISAM+ is an extension to ISAM where the users of the mobile nodes define the $\text{cost}_i(t)$ of their data streams from their devices. ISAM+ is also a greedy algorithm. The goal of ISAM+ is to maximize the amount of ADUs subset to the system application rates and budget constrains, with respect to their QoI values. Thus, it aims to select the subset of ADUs that depicts the highest total QoI, among the feasible subsets that provide the maximum possible amount of ADUs, based on the rate constraints. We use this algorithm to explore whether the selection of more ADUs with probably lower QoI values can compete with the other algorithms that select ADUs with high QoI values no matter if the total amount is smaller. The maximization problem now becomes:

$$\begin{aligned} \text{Maximize} & \sum_{i \in M} \text{ADU}_i(t) \\ \text{subject to} & \sum_{i \in M} \text{cost}_i(t) \leq Budget \\ & \sum_{i \in M} \text{ADU}_i(t)/\text{Deadline}_q \leq \text{Rate}_q \end{aligned}$$

The algorithm states:

1) *Order all offers in set A s.t. $QoI_1(t) \geq QoI_2(t) .. \geq QoI_{|A|}(t)$*
2) *Let Winner set $S = \emptyset$ and $k = 1$;*
3) *While $\big(k \leq |A|$ && $\frac{|S|+1}{Deadline_q} \leq Rate_q$ $\big)\{$ $S \leftarrow S \cup \{k\}$, $k \leftarrow k + 1$ $\}$*
4) *While $\big(\sum_{i \in S} cost_i(t) \geq Budget$ $\big)$ $\{$ Substitute $offer_i(t)$ with the highest $cost_i(t)$ from S with the following unallocated $ADU_x(t)$ from A, or with null if such an offer does not exist;$\}$*
5) *Return set S;*

Worst-Case Complexity: Similarly to ISAM, it costs $O(|A| \log |A|)$ to sort the offers. It costs at most $O(|A|)$ to iterate through the first while, since the assignments only cost $O(1)$. Moreover, the second while-clause will be executed at most $|A|$ times, if we remove all offers. Substituting the offer with the highest $\text{cost}_i(t)$ costs $O(1)$ if an index is maintained for the costs. Hence, the second while-clause costs $O(|A|)$ and thus the worst-case complexity of ISAM+ is $O(|A| \log |A|)$.

ISAM++ is an extension to ISAM like ISAM+ with the exception that ISAM++ tries to maximize the "value for money" of the data streams when sampling. Again, users can define the $\text{cost}_i(t)$ of the data streams from their devices. However, in ISAM+ when a user sets an extremely high cost compared to the other users, it is more likely to be rejected, no matter if the data stream has an increased QoI value. On the other hand, ISAM++ tries to receive as many ADUs as possible that reflect an increased QoI value compared to their cost. Thus, the maximization problem becomes:

$$\begin{aligned} \text{Maximize} & \sum_{i \in M} \text{Information\_Gain}_i(t) \\ \text{subject to} & \sum_{i \in M} \text{cost}_i(t) \leq Budget \\ & \sum_{i \in M} \text{ADU}_i(t)/\text{Deadline}_q \leq \text{Rate}_q \end{aligned}$$

ISAM++ works as follows:

1) *Order all offers in set A s.t. $Information\_Gain_1(t) \geq Information\_Gain_2(t) .. \geq Information\_Gain_{|A|}(t)$*
2) *Let Winner set $S = \emptyset$ and $k = 1$;*
3) *While $\big(k \leq |A|$ && $\frac{|S|+1}{Deadline_q} \leq Rate_q$ && $cost_k(t) \leq Budget * (QoI_k(t)/\sum_{i \in S \cup \{k\}} QoI_i(t))$ $\big)$ $\{$ $S \leftarrow S \cup \{k\}$; $k \leftarrow k + 1$; $\}$*
4) *Return set S;*

Worst-Case Complexity: The complexity can be computed similar to ISAM as $O(|A| \log |A|)$.

C. Rate Allocation

In our previous work we have studied the problem of determining the rate allocation of a distributed stream processing application at run-time, in a distributed manner [14]. In this paper, we use this work to determine the incoming rate of application q (Rate_q) subject to the application Deadline_q constraint and the resource availability in the system. We give a short description of how this is achieved below:

Let C be the set of application components that can be deployed in the system, and R_{c_i} be the rate that each component c_i will get. The rate of the application Rate_q can be considered as the input rate of the source component. Every other component has a rate R_{c_i} that depends on the selectivity among components. The selectivity represents the relation between the input and the output rates of a component. Every application q is represented as a graph of component invocations that executes on an input data stream and needs to finish within Deadline_q. The objective is to maximize the rates for the components invoked by the applications in the system so that the QoS requirements of the applications are met and the resource constraints are satisfied. This is described as: $maximize \sum_{c_i \in C} R_{c_i}$

In order to ensure that the application executes within its deadline, the sum of the computation times of all the components invoked by application q and the corresponding communication times need to be smaller than the application deadline. This can be expressed as follows: $max_{path}(\sum_{c_i \in q} \text{Comp}_{c_i}(R_{c_i}) + \sum_{c_i \in q} \text{Comm}_{c_i->c_{i+1}}(R_{c_i})) \leq \text{Deadline}_q$

where max_{path} is used in the case that the application is represented as a graph with more than one paths. In the above equation, $\text{Comp}_{c_i}(R_{c_i})$ represents the average computation time required for component c_i to execute at rate R_{c_i}, obtained through profiling techniques with low overhead, and $\text{Comm}_{c_i->c_{i+1}}(R_{c_i})$ represents the corresponding communication time among components c_i and c_{i+1}.

All components on a processor in the distributed stream processing system are competing for available CPU resources. This constraint states that the sum of the rates allocated to each component multiplied by the CPU share required to process each ADU must be smaller than the fraction of available resources. For each node we denote: $\sum_{c_i \in n} R_{c_i} * CPU_{c_i} \leq 1$

We focus our attention on the processing resource (CPU capacity), as this is the sparsest resource in stream processing intensive environments. CPU_{c_i} denotes the cpu share required for component c_i to process one ADU in the period of the Deadline. The selectivity sel_{c_i} of a component c_i represents the average ratio of the number of output data units to the number of input data units of c_i and depends on the service run by the component. Then the flow conservation constraint is represented as: $R_{c_i+1} = sel_{c_i} * R_{c_i}$

Discussion. The constrained maximization problem can be solved using linear programming techniques. In our implementation the problem is solved in a distributed manner and thus each component defines its own rate. The rate of the application's source component that refers to the input rate of the application Rate_q would then be provided to the sampling components to perform the sampling.

IV. Experimental Evaluation

A. Experimental Setup

We have implemented our techniques in Java6 with approximately 5700 lines of code and tested it on PlanetLab.

Traffic Monitoring Application: The experimental evaluation scenario was a traffic monitoring application where the goal is to identify congested areas in the Interstate 880 in California in real-time. We used Berkeley's Mobile Millennium Dataset [6], which includes real time traffic data taken from GPS-enabled phones, to determine congested areas using streams of traffic data provided by individual mobile phone users. That dataset consists of data taken from 77 cars for the period between 10:00 and 18:00 on the Feb 8th 2008. Each ADU in the dataset is associated with the following information: <time, latitude, longitude, speed and carID>. The results presented are the average over 5 runs.

The application scenario was implemented with 4 main components: (1) A **selection component** that receives ADUs and computes each car's geographic region and trajectory. (2) A **projection component** that projects the latest ADUs, based on geographic region and trajectory information. (3) A component that estimates the **average speed** for the projected ADUs. (4) A component that defines the congested areas.

The experimental evaluation focuses on the following parameters: (i) **Error metric**, (ii) **Sampling Size**, (iii) **Energy and Cost Efficiency**, (iv) **Total sent ADUs**, (v) **Payment** and (vi) **Threshold impact**.

The Region weights are defined with values from 1 to 10 and the weights are distributed so that the values increase in accordance with the Region number. Thus, Region5 receives a higher weight than Region3. The Density weight is defined by subtracting the number of cars in the current region from the maximum expected cars which we have set to 10, based on our dataset. Finally, the Transition weight is set to 10 when a mobile reaches a new region. Our formulation is generic so that different regions can have different values for the Transition weight to indicate the relative importance of the regions. When referring to regions we define them as a region-trajectory entity. Hence, Region2-SouthWest is a different region than Region2-NorthWest, since these two regions can be opposite lanes to a highway, so they should be treated separately. The default average speed for the sample regions is set to 27.91, which is the total average speed from all regions in the dataset.

B. Experimental Results for several QoI functions

In the first set of experiments we evaluate the different QoI functions' behavior under the same resource conditions. In these experiments the base algorithm (ISAM) is used so that the evaluation can be independent of the ADU cost, which is a fixed price for all ADUs. Moreover, we define a rate allocation of 5 ADUs per second which refers approximately to a sample size of 25% over the available ADUs.

We present 4 QoI functions: (1) **Random QoI** is our Baseline function, where the selection is made randomly by the scheduler, based on the QoS constraints without using informational criteria, (2) **Region QoI** is the function where each mobile receives a specific weight based on the region it

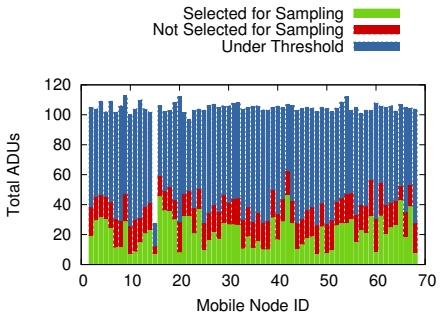

Fig. 5. Region QoI

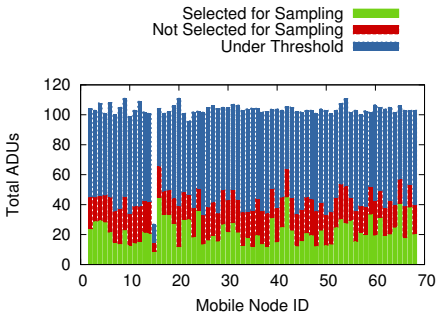

Fig. 6. Region + Density QoI

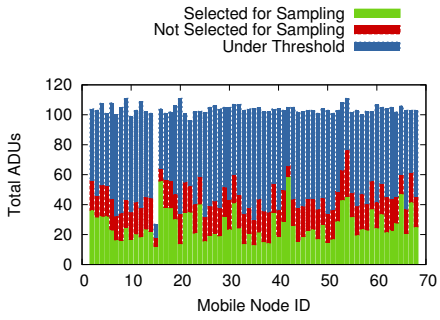

Fig. 7. Region + Density + Transition QoI

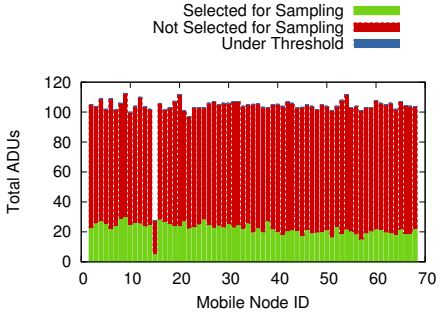

Fig. 8. Random QoI

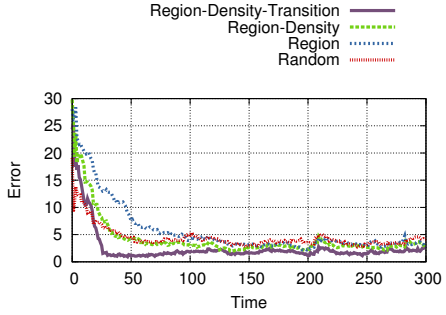

Fig. 9. Error Over Time.

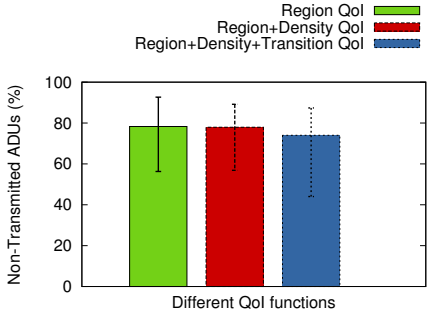

Fig. 10. Cost Efficiency (%)

belongs and has the form of the utility function of figure 4. (3) **Region+Density QoI** where both region and density (has the form of figure 2) weights are computed for the mobile node and (4) **Region+Density+Transition QoI** where the 3rd QoI function is extended with a weight whenever a mobile node changes region (has the form of figure 3). In this set of experiments we aim to reduce the Error in relation to the region's importance to demonstrate how our technique approaches the application logic for the sampling.

Figures 5, 6, 7 and 8 illustrate the number of ADUs that were selected for sampling, rejected or were under threshold for each mobile node. Figure 5 illustrates the behavior of the Region QoI function. As can be observed, all the mobile nodes participate in the sampling process and since they get paid to provide their data streams, we expect that they will stay connected to the system. One important observation is that the ADUs that were selected for the sampling are much fewer compared to the total number of produced ADUs, and specifically it corresponds to an average of 22% from the total ADUs. We should also remark that the majority of the produced ADUs were below the threshold (62% of the total ADUs). The fact that these ADUs were withdrawn, infers a tremendous improvement in the cost, both in terms of energy and money, since they would have been rejected from the sampling mechanism anyway.

Figure 6 presents the density weight combined with the region QoI. As can be seen from the figure, the behavior of the mechanism is similar to the previous experiment. Again, we can observe that all the nodes contribute their data stream samples to the system. Moreover, the percentage of the selected data streams compared to the total produced ones was again 22%. Furthermore, 58% of the total ADUs were below threshold.

Similarly, in figure 7 we present the third function that combines the density, region and transition weights. All nodes contribute their data streams as in the previous figures. However, the selection of the data streams was slightly increased and the average percentage of the selected data streams was 24%. The percentage of the ADUs that were below the threshold in this function was 57%.

Figure 8 illustrates the results for the random QoI function. The average percentage of the data streams selected for sampling was 22%. As it is denoted from this function, there was about the same amount of ADUs selected compared to the previous functions. Thus, we infer that the threshold did not force ADUs to be excluded from the sampling. The random QoI technique does not provide an infrastructure to preserve metrics, such as threshold since every ADU may be chosen for processing in any run of the algorithm. Hence, all of the produced ADUs are sent to the sampling mechanism causing energy and cost overheads. We also note that although it seems that all of the mobiles provide the same quantity of ADUs, this is not the case for every instance of the experiment. As mentioned, these results represent the average values among all experiments, so the randomized selection seems to converge in one value. However, in each separate experiment the mobiles had great variations in providing ADUs. In the contrary, our QoI functions gave similar results for the same instance of the experiment since there was no random factor.

Another point to mention is that, as can be seen in all the above figures, mobile node 14 provides fewer data streams compared to those provided by all the other mobile phones. We searched further why this happens. The reason we see this behavior is that it only starts producing its data streams at the end of the experiment, so the number of data streams produced by this device is much fewer.

Figure 9 illustrates the average error for all the regions over time. At the beginning all the approaches depict low accuracy,

since they have not processed ADUs from enough regions, due to sampling. However, after the first 100 seconds it is observed that the random QoI has the lowest accuracy. The region QoI is close enough to the random one, due to the decreased accuracy in low weight regions. The region+density QoI has a small improvement to the results of the region one and finally the combined QoI has an obvious improvement over all approaches.

One of our goals was to improve the energy and cost efficiency of the mobile nodes. By using our infrastructure the mobile node should only send two numbers to the sampling node (QoI, cost) or nothing at all, if the node is below threshold. Figure 10 presents the average percentage of ADUs that were not sent to the sampling component for each QoI function. We use the error bars to depict the highest and the lowest percentage that we received from the mobile nodes. We have computed that approximately 76.7% of the ADUs for each mobile were not transmitted and this percentage at most, depends on the threshold. Thus, there is a great efficiency improvement on both energy and money for the mobiles. In the case of the random QoI function the mobile nodes would transmit all their ADUs in every run, as it is possible to have them processed and thus the cost efficiency is 0% at all times.

C. Experimental Results for QoI on several sample sizes

In this set of experiments we evaluate the previous QoIs by executing the experiments on several sampling sizes. Thus, we evaluate the Error Factor over different sampling sizes that range from 5% to 70%, which refer to rate allocation that ranges from 1ADU/sec up to 15ADUs/sec.

Figure 11 illustrates the average Error Percentage, which was presented earlier over all regions and runs, for each sampling size. As can be inferred from that figure the random QoI has obviously the worst outcome in all sampling sizes. The region QoI follows and the density+region one suggests a slight improvement to that. However, again there is a great improvement of the whole system accuracy with the combined QoI, especially for the smaller sample sizes where the error is expected to be high.

In figure 12 we present the Error factors in km/h, for the lower weighted regions. As can be observed, due to our metrics the random QoI has an improved accuracy compared to others that give a higher priority to the high weighted regions. However, this is only up to 40% since there should be enough data processed in that percentage to overcome the region weights. Moreover, we observe that the QoI that is based on region has the worst accuracy since it constantly tries to satisfy the needs of the other regions. The other two QoIs result on a better behavior, since they use more metrics that involve all regions and again the combined QoI function has the highest accuracy among them.

Figure 13 presents the same data for the higher weighted regions. As expected the QoI functions that use the region weight as a metric, result on increased accuracy than the random case. However, the interesting part is that they have an almost identical behavior. Thus, it can be inferred that the information gain from the additional metrics, for the lower weighted regions, have not reduced the accuracy to the higher importance ones.

In order to clarify the difference of the metrics, one would imagine that since both the region+density function and the combined one have the same average sampling error in both figures 12 and 13 at sample size 5%, they should also have the same Error Percentage. However, this factor provides a representation of the error based on the actual expected price and not only to the distance of the values. Although a distance of 2km/h to the sampled result compared to the full dataset one, can be a small difference if the average speed is approximately 60km/h, it would be a great deal for that factor if the actual average speed was 5km/h.

D. Experimental Results among different Sampling Strategies

In this set of experiments we evaluate the behavior of the different sampling mechanisms we proposed. We conduct several experiments using different rate allocations, to observe the behavior of the algorithms. We have used the combined QoI function for all three algorithms and every experiment suggest a budget of 1.00$ in every run. In the basic algorithm ISAM we expect that the organization, that defines the fixed price, would suggest a price that can be handled in all rate allocation instances. Thus, this fixed price is 0.07$ for each ADU. In the other two algorithms, the prices are randomly chosen for each mobile phone and range among 0.01$ to 0.30$ to better observe the behavior when the budget becomes a bottleneck to the available resources. ISAM can be inferred as the ideal behavior since it can choose those ADUs with the highest QoI, because there is no budget bottleneck for the available resources. However, the other two algorithms seem to be more realistic.

Figure 14 illustrates the Error of each mechanism in different rate allocation resources. It is obvious that until rate reaches 5, the cost is not a bottleneck and all three have an identical behavior. From that point, ISAM+ depicts the worst behavior. In order to process as many ADUs as can be supported by the rate allocation mechanism, ISAM+ has to reject ADUs when their prices are high, no matter if those ADUs can provide a higher information gain. Moreover, we can observe that as rates continue to increase, the problem becomes worse and the Error begins to rise a little instead of decreasing in order to process as many low cost ADUs as the system's capacity can handle, over high-cost ADUs that may contain higher QoI value. On the contrary, ISAM++ that uses ADUs with the highest Information Gain has an impressive behavior, which is close enough to the ideal ISAM.

In figure 15 we can observe the total payment from the sampling mechanism to the mobile nodes. ISAM has a linear behavior due to the fixed price. ISAM+ pays more than ISAM++, since ISAM++ is not trying to make a full utilization of the budget. However, both techniques start to converge as the budget becomes insufficient.

Figure 16 presents the number of ADUs processed at each rate and the corresponding sample size. Again, ISAM has a linear behavior since there is no budget concern in that mechanism. ISAM+ tries to maximize the processed ADUs based on the rate allocation, thus it processes more ADUs compared to ISAM++, which tries to maximize the Informational Gain instead. However, the total number of processed ADUs depend on their cost, so both techniques seem to be bounded from

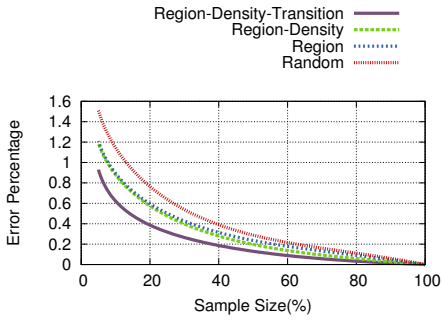

Fig. 11. Error Percentage for QoIs.

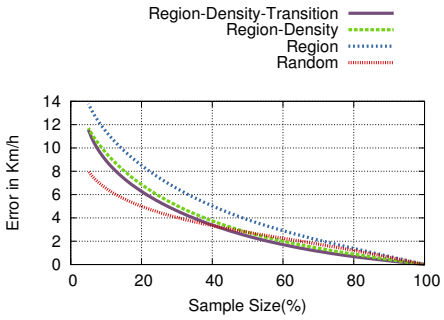

Fig. 12. Error on lower importance Regions.

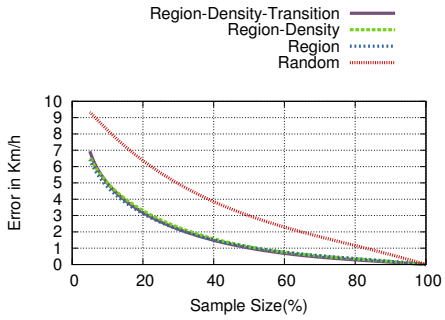

Fig. 13. Error on higher importance Regions.

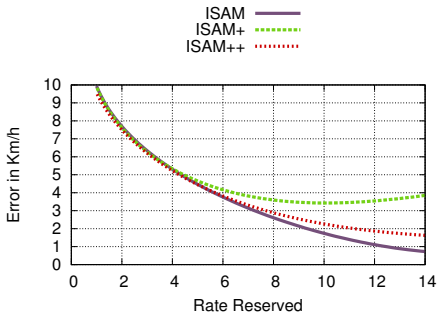

Fig. 14. Error for Algorithms on different rates.

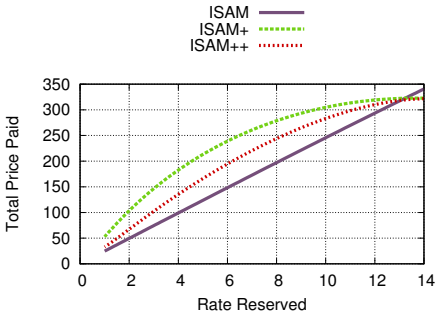

Fig. 15. Total Paid Price for Algorithms on different rates.

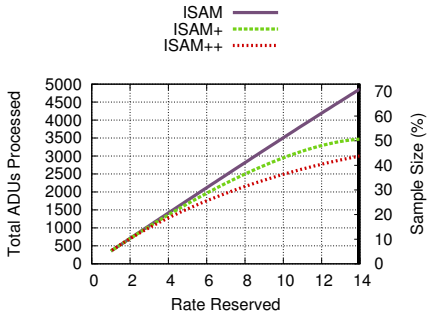

Fig. 16. Total ADUs processed for Algorithms on different rates.

the budget as sample size increases. Moreover, we illustrate the sampling size percentage compared to the rates reserved which has an identical figure to the processed ADUs for the same rates and budget.

E. Discussion

In this section we evaluated our proposed mechanisms. We have shown that using QoI functions increases the accuracy of the results, based on the application targets, and we have observed that combining QoI metrics, improves the informational profit. On the other hand, the random QoI function resulted in the worst behavior since it does not consider neither the application logic nor the importance of the data streams when sampling.

We have also illustrated that our sampling mechanisms improve the cost efficiency of the mobile nodes and we remark that the evaluation between the sampling algorithms, where users can choose their payments, shows that ISAM++ outperforms ISAM+ since it is more accurate, processes fewer ADUs and needs to pay less money to the mobile nodes.

V. Related Work

Participatory sensing systems have recently become increasingly popular for processing data sensed on mobile devices to identify events of interest [1]. Several of the current approaches focus on traffic monitoring systems [2], [15]. For example, CarTel [15] works with continuous queries where users set the rate that they want to receive the ADUs from a specific geographical area, without considering the issue of data redundancy or the information quality of the ADUs and the system resource availability. VTrack [2] uses map matching based on a Hidden Markov Model scheme, with a way to interpolate sparse data to identify the most probable road segments driven by the user and to attribute travel times to those segments. Authors in [16] propose techniques to provide incentives to the users of the participatory sensing systems. They consider two types: a platform-centric mechanism, that uses a Stackelberg game to maximize the utility of the platform, and a user-centric auction-based incentive mechanism. Since their scheme does not consider the quality of the information, we can employ these mechanisms to attract more users into the participatory sensing system.

The research in the area involving the problem of sampling is very rich and several approaches have been proposed (including previous work by the authors). In this section we review the related work for mobile sensor sampling. In [8], the authors perform region sampling in sensor networks. They aim to bound the energy consumption while minimizing the approximation error and use statistics to predict the optimal sampling plan, while we aim to maximize the informational benefit. Al-Kateb *et al* in [17] propose an algorithm to extend the reservoir sampling, that selects a uniform random sample of a given size from an input stream of an unknown size, with an adaptive-size reservoir. Our technique, driven by the application logic, outperforms uniform random samples. Arai *et al* in [18] propose a technique for sampling in aggregation queries for a peer-to-peer database as an infrastructure which is opposed to the stream processing architectural logic. In [19] they suggest an approach that samples a small fraction of the sensor data and utilize the correlation model to estimate the non-sampled readings to decide for the best set of sensors to sample. This selection is greedily computed by selecting the sensor that maximizes the estimation confidence of the correlation model, but results on large computational overhead and energy consumption, as the sensor data and their correlations evolve. Stratified Sampling [20] is another well-known method for efficient sampling from a population, where

the members of the population are combined to constitute homogeneous subgroups(stratums). Random sampling is then performed from each stratum independently and the global result is a weighted combination of all the partial stratum results. Authors in [21] propose a sampling technique that generates samples over sliding windows based on the proportion and the population of the attributes in a window. In [22] they propose a mobile sensing framework, called OptiMos. OptiMos aims to identify the optimal sampling for the sensor readings in each segment, where the selected readings can guarantee reasonably high sensor coverage with limited sampling rate. Our sampling technique differs from all these techniques since it considers real-time constraints as well as the information quality and cost of the individual data streams.

Finally, techniques that consider the Quality of Information have also been proposed. Wu *et al* in [23] suggest an approach where the quality of data of queries compared to their processing cost is considered by the scheduler of the database to satisfy the user queries. Their definition for quality of data, however, does not represent the system's information profit, but the quality profit for the query. In [24] the proposed technique uses QoI to identify and select sensors to provide the most "relevant" sensory data to a users needs, considering a cost per provider. However, they define QoI as the spatiotemporal coverage of the sensor, without considering any timeliness constraints on the data collected. Authors in [25] propose a model for resource adaptation in stream-mining algorithms that takes quality into account. However, they do not consider the quality of the individual data streams but only the quality of the result for each task.

In our previous work [11] we introduced the architecture of our Participatory Sensing System, where we investigated the impact of different QoI functions for providing streams of events, in the context of traffic monitoring applications. The goal of this paper is to investigate different sampling strategies, to maximize the information profit, subject to the real-time and budget constraints, that focus on: (1) efficiency, (2) constraints satisfaction, (3) information gain, (4) monetary cost.

VI. Conclusions

In this paper we present our approach for mobile stream sampling under time constraints. We propose algorithms that determine a suitable sample of the data streams based on the information quality that the individual devices provide in the system and the corresponding costs. The advantage of our technique over the classical sampling methods, is that it considers application real-time constraints and approaches the application logic with the utility functions. Detailed experimental results illustrate that our approach is practical, efficient and depicts good performance.

Acknowledgment

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) - Research Funding Program:Thalis-DISFER, Aristeia-MMD, Aristeia-INCEPTION Investing in knowledge society through the European Social Fund, the FP7 INSIGHT project and the ERC IDEAS NGHCS project.

References

[1] S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, G. seop Ahn, and A. T. Campbell, "Metrosense project: People-centric sensing at scale," in *SenSys*, Boulder, Colorado, USA, Oct-Nov 2006.

[2] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones," in *SenSys*, Berkeley, California, USA, November 2009, pp. 85–98.

[3] M. Olson, A. H. Liu, M. Faulkner, and K. M. Chandy, "Rapid detection of rare geospatial events: earthquake warning applications," in *DEBS*, New York, USA, July 2011, pp. 89–100.

[4] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikinen, V. Tuulos, S. Foley, and C. Yu, "Data clustering on a network of mobile smartphones," in *SAINT*, Munich, Germany, July 2011.

[5] "Google maps." [Online]. Available: https://maps.google.com/

[6] J. Herrera, "Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment," in *Transport. Res. Part C*, 2009.

[7] M. Halkidi, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Resilient and energy efficient tracking in sensor networks," *Int. J. Wire. Mob. Comput.*, vol. 1, no. 2, pp. 87–100, Feb 2006.

[8] S. Lin, B. Arai, D. Gunopulos, and G. Das, "Region sampling: Continuous adaptive sampling on sensor networks," in *ICDE*. Cancún, México: IEEE, April 2008, pp. 794–803.

[9] P. Li, B. Ravindran, and E. D. Jensen, "Adaptive time-critical resource management using time/utility functions: Past, present, and future," in *COMPSAC Workshops*, Hong Kong, China, September 2004, pp. 12–13.

[10] B. Ravindran, E. D. Jensen, and P. Li, "On recent advances in time/utility function real-time scheduling and resource management," in *ISORC*, Seattle, WA, USA, May 2005.

[11] I. Boutsis and V. Kalogeraki, "Short paper: Dynamic qos-aware event sampling for community-based participatory sensing systems," in *DEBS*, Berlin, Germany, July 2012, pp. 153–156.

[12] Y. Singer, "Budget feasible mechanisms," in *FOCS*, Las Vegas, Nevada, October 2010.

[13] N. Chen, N. Gravin, and P. Lu, "On the approximability of budget feasible mechanisms," in *SODA*, San Francisco, California, Jan 2011.

[14] I. Boutsis and V. Kalogeraki, "Radar: Adaptive rate allocation in distributed stream processing systems under bursty workloads," in *SRDS*, Irvine, California, October 2012.

[15] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "Cartel: a distributed mobile sensor computing system," in *SenSys*, Boulder, Colorado, USA, October 2006, pp. 125–138.

[16] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in *Mobicom*, Istanbul, Turkey, August 2012.

[17] M. Al-Kateb, B. S. Lee, and X. S. Wang, "Adaptive-size reservoir sampling over data streams," in *SSDBM*, Banff, Canada, July 2007, p. 22.

[18] B. Arai, G. Das, D. Gunopulos, and V. Kalogeraki, "Approximating aggregation queries in peer-to-peer networks," in *ICDE*, Atlanta, GA, April 2006.

[19] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *VLDB*, Toronto, Canada, Aug-Sep 2004.

[20] W. G. Cochran, *Sampling Techniques, 3rd Edition*. John Wiley, 1977.

[21] K.-T. Chuang, H.-L. Chen, and M.-S. Chen, "Feature-preserved sampling over streaming data," *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 4, pp. 15:1–15:45, Jan. 2009.

[22] Z. Yan, J. Eberle, and K. Aberer, "Optimos: Optimal sensing for mobile sensors," in *MDM*, Bengaluru, India, July 2012.

[23] H. Wu, Q. Luo, J. Li, and A. Labrinidis, "Quality aware query scheduling in wireless sensor networks," in *DMSN*, Lyon, France, August 2009.

[24] G. Tychogiorgos and C. Bisdikian, "Selecting relevant sensor providers for meeting "your" quality information needs," in *MDM*, Luleå, Sweden, June 2011.

[25] C. Junghans, M. Karnstedt, and M. Gertz, "Quality-driven resource-adaptive data stream mining?" *SIGKDD Explor. Newsl.*, vol. 13, no. 1, pp. 72–82, Aug. 2011.