# IMPROVING THE EFFICIENCY OF QUERY RESPONSE AND UNAVAILABILITY ISSUES IN A SENSOR NETWORK OF MOBILE DEVICES

D N Kartheek, G Amarnath, P Venkateswarlu Reddy, S Srujan

***Abstract***

Sensors are now-a-days finding wide applicability in many kinds of electronic devices. Mobile phones turn out to be a best way to deploy them to provide wide coverage and keep the network live for an extended period. The limitation for embedding sensor devices in mobile phones, the mobility issue was successfully dealt to an extent in [1] by using virtual sensors. Also, certain areas of interest need not be frequently sampled as the sampled data does not show a significant change. This was used as an advantage to reduce the end-end latencies by minimizing expensive data collection from sensors. It uses the techniques to cache aggregate results of samples and one-pass sampling to utilize cached data.

We propose that the end-end latency can be further reduced and the availability of the nodes increased by taking advantage of the virtual sensor layer and employing caching in the same layer. One example can be an air pureness monitoring device embedded in cell phones. The results are computed by simulating an adequate number of samples and would give the efficiency of the technique along with the latency.

***Keywords-*** caching, query latency, sampling, mobile phones, virtual sensor node.

## 1 INTRODUCTION

We have known for a long time now that mobile phones can be used as sensor devices. These devices have inbuilt audio, video sensors and also other sensing devices can be connected to the device using Bluetooth. The miniaturization of sensing devices may enable them to be embedded within the mobile phones.

Mobile phones have primarily two sensors: a camera and a microphone. It would be an efficient use of this sensor infrastructure if we could build a sensor network that exploits the deployed base of millions of mobile phones worldwide. Here, mobility has been a primary concern in all such networks which render the network inefficient and useless at times. For example, sampling a desired region with a given device becomes difficult due to the uncontrolled nature of device motion. [1] employs the concept to make use of a data centric abstraction to deal with this problem. Introducing a layer of static virtual nodes corresponding to the sampled data locations obviates the need for treating the physical devices as our sensor nodes. Queries can be directed to the virtual sensors, which accumulate the data samples from the physical devices.

Again, Query processing is an expensive task, particularly if we wish to have live data as we need to sample data a number of times. [2] uses COLR-Tree that, in part deals with using cached data when a node is unavailable. COLR-Tree uses two techniques to optimize end-to-end latencies of user's queries by minimizing expensive data collection from sensors. First, it uses a new technique for caching aggregate results computed over sensor samples with different expiry times. Second, it incorporates an efficient one-pass sampling algorithm with its range lookup to utilize cached data to compensate for occasional unavailability of sensors.

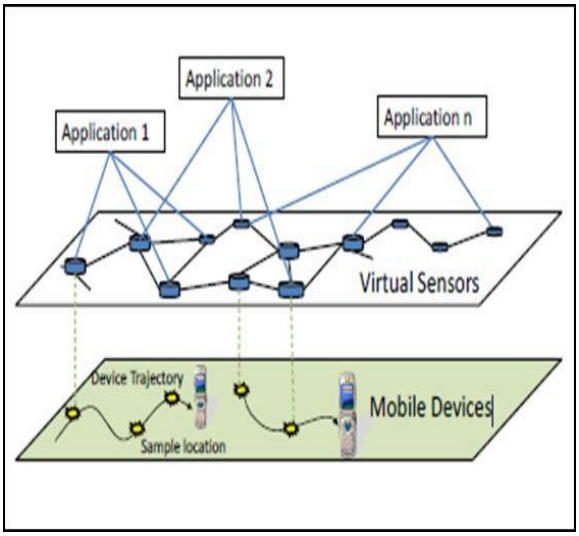
We propose to make use of caching to effectively deal with sensor unavailability in our implementation of the sensor network that we build using mobile phones as sensor nodes while simultaneously dealing with the mobility problem by using a layer of virtual sensor nodes. This technique can be employed over the deployed base of mobile phones in a city which have embedded atmospheric air monitoring sensors that measure the carbon compound content in the air and relay it to the base. Based on the readings, the concerned can have certain neutralizing elements released into the air in the air. This is particularly useful in large cities with huge traffic that spike the carbon levels. We consider a temperature sensor system to implement the idea and provide the results.

The advantages in using this mechanism to measure temperature are highly advantageous as 1) the temperature at numerous points in a part of a continent can be known and irregular temperature patterns can be studied. 2) Given temperature rise in the heart of the city is largely due to toxic heated gases from the vehicles, we can make a pattern of the traffic and thereby assist drivers to choose the best routes to their destinations by sending this pattern over to their GPS devices on request. 3) It is efficient to make use of the existing systems, the mobile devices rather than deploying new sensor systems that may increase the cost of implementation. This system can be made available on a web portal that can host data generated by millions of sensors and lets users query live data directly on the map.

The following section gives an insight into the proposed architecture and provides details of the nodes and different layers. The third section deals with each layer in detail and describes how the entire system operates. The implementation section
describes the constraints imposed on each functional component and furnishes the results. Conclusion has points for future work and improvements to the current system.

## II. SYSTEM ARCHITECTURE

In this system, sensor nodes are accessed not directly by their node ID's or network addresses. As in [1], we employ a data abstraction layer that effectively obviates the need for an overhead of storing network addresses or node ID's. That is, a layer of virtual sensor nodes is superimposed over the actual physical network and the data obtained from the virtual sensor corresponds to the data collected by it from the physical devices in its supposed coverage area. These virtual nodes, implemented in the network infrastructure are deemed static. The applications are the layer directly above the virtual sensor layer and direct their queries to the virtual sensors instead of the physical devices.




The virtual sensor nodes are supposed to hold the cache for the respective regions. The caches are updated based on requirement and is decided by the software.

## III. DESCRIPTION

The architecture described above strives to increase the availability of the physical devices and also to minimize the query latency. The first objective is achieved is by using the static virtual sensor layer and utilizing data from cache. Query latency can be reduced by limiting repeated expensive data gathering and using the already sampled data.

The samples collected from the devices have a location stamp on them and hence are identified to be under a particular sensor node. This concept maintains that the location of a virtual sensor node is static and all data gathered will be grouped under the respective sensor node.

As in [1], all applications query the virtual sensor layer directly instead of querying the physical devices that greatly reduces the query latency compared to traditional query mechanisms. Also the need for maintaining the node ID's has been eliminated. This follows as it suffices to have a minimum number of sensor devices to provide with the samples required.

To deal with the problems of unavailability and query latency, we introduce a cache corresponding to each virtual sensor. This cache keeps track of the samples gathered from the physical devices along with the time stamps. When a virtual node has no data due to unavailability of physical devices, the system is still live and working with the cached data stored. This works with the same principle as in [2], by allowing a time out after which the data is considered stale. Similarly, the mechanism of slot-cache that makes use of timestamps to group efficiently the data collected and respond to the queries based on freshness of the data is inherited from [2].

COLR-Tree [2] needs to aggregate data in different spatial granularities. The assumption was that the locations of sensors do not change often, allowing COLR-Tree to be built bottom-up, in batch mode, by iteratively computing sensor clusters with a k-means algorithm [4] to construct a hierarchy. We periodically reconstruct the COLR-Tree index to reflect any change in sensor locations. Our proposal does not require any usage or reconstruction of the COLR-Tree as we do not operate directly on mobile physical nodes. But these are used merely as tools for information. This raw data is stored and processed when requested if it still is not stale. In case that it is, we will have the only option of going with collecting data at that time, which would definitely increase the query latency. A possible work around would be to populate the cache more frequently to prevent the above situation. The frequency should be greater than the frequency at which the queries flow in or in other way, the interval between successive sampling events should be less than the timeout. The frequent sampling would consume the battery in the sensor devices quickly as sampling involves sensing the data formatting it and relaying it to the base. This would fail us in increasing the lifetime of the devices and ultimately the network. So, as each sample has a separate timeout, we retain the values from samples that are within the timeout interval but expunge the timed out values and replace them with fresh values.

The use of caching will see a significant drop in bandwidth utilized as many queries can be replied with the data in the cache. The cache and timeout concepts obviate the need for sampling the physical devices for each query also saving power for the devices and increasing life time of the network. The physical devices time stamp the data that is considered stale after a predetermined time out, also set by the device. We propose to include a mechanism that tracks the incidence of queries to a particular virtual node and in case it is higher, then a method of automatic sampling with an interval of $\Delta t$, where t is the time before the sampled data is rendered stale. This is implemented at the network infrastructure same as the virtual layer. It records the number of queries per unit time and compares it with the timeout of the aggregate data in the node. A suitable decision is implemented to be carried out either to sample the region or to proceed without any action. The choice is made not to sample the region if the frequency of queries is much greater than the sampling frequency of the network. Again, in the case that a decision to sample has been made, an enhancement to traditional sampling method can be that depending on the query frequency and the time left before the data is rendered useless, we can schedule the sampling to occur in intervals of multiples of timeout 't'. In the case that a query comes in before a sample is taken, the traditional process of responding by collecting the data after the query is received is followed. This form of automated querying ensures the freshness of the data always at least at the busy nodes. Thus, accuracy and integrity of query response is maintained.

## IV. IMPLEMENTATION

The network has been simulated and the following rules are employed in simulating. We considered four rectangular cells each with a virtual sensor at its center. All the nodes in a network cell transmit the sampled data to the base. The base is in fact the virtual node implemented in the network infrastructure.

We have mobile devices spread over all the four rectangular cells with finite boundaries. The mobiles can change position at any time. Initially, the mobiles are assumed to have been sampled and data stored in the cache. Although much better applications exist, just for the ease of storing and processing, the data sampled is assumed to be temperature. The temperature samples are ranged in between 35 and 39.

Data sampled is refreshed, i.e. the cache is updated every few seconds and the energy of nodes is checked to mark the failed nodes. Each sensor node is implemented as a data structure with position and energy attributes. When sampled, the location and sampled data are stored in the cache along with the time at which the data is sampled. When queried, the data cached is utilized in sending a response, provided a minimum number of 'fresh' samples are available in the cache. The parameter 'fresh' is tagged to a sample if it was updated to the cache no earlier than the timeout. The timeout here is fixed as 3 seconds. For each transmission of data by the sensor, the energy is reduced by a certain predetermined amount.

One-tenth of the total mobile nodes are programmed to move by 0 to 3 units in space, along two axes. Correspondingly, their position information is updated and in case of cell crossover, its previous virtual node entry is deleted. No entry is made in the new cell virtual node cache.

When a query is made for a location in a particular cell, its cache is first checked for the minimum number of samples. If the desired number is not available, mobile nodes are sampled to make the number of samples in the cache equal to eight and then the response is sent using the updated cache.

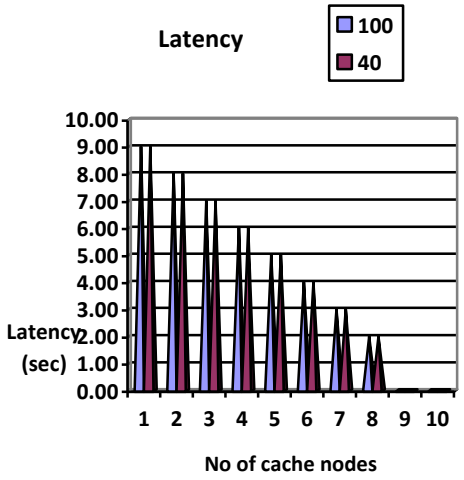
In comparing the query latency, we have compared the best and worst case, the best case being when all the required samples in the cache are fresh and the worst case being when no node in the cache is fresh. The latter case requires the nodes to be sampled again which might incur significant delay which includes the transmission delay for cellular communication from mobile to base station. This delay at its maximum is one second. We have taken into account this maximum.

The simulation takes the following steps.

Algorithm:

1) The cache is first filled up by sampling the nodes along with the time.
2) The simulation time is started.
3) The timer is repeatedly checked for the time out and cache is updated.
4) Position of ten percent of the total nodes is changed and the update function is called to reflect these changes.
5) The above steps are put in a loop which iterates if the simulation time is yet to be reached.
6) The simulation ends when this condition is met.

The experiment is carried out in Turbo C 3 using 100 nodes and 40 nodes taking time out of 3 seconds. The best case times are zero. The worst case time as discussed above is the maximum number (in worst case, cache is empty) of nodes to be sampled multiplied by the maximum cellular latency, assuming each node is sampled sequentially. We ran these experiments on a laptop machine with an Intel Pentium M Processor 735, 1.24GB RAM and a Seagate ST94813A.




The above figure shows the latency of the query in its worst case, tested using both 100 and 40 nodes. Over 50 samples are collected for each configuration and the readings show the latency largely depends on the cellular communication latency, i.e. maximum one second. A majority of the samples gave latency in between 2 and 5 seconds with 9 as the maximum. The cellular latency is taken randomly as either 0 or 1 for a sample. We are in the process of gathering more samples to show the increase in availability. The energy consumed during transmission is taken into account and the same amount is reduced for the node each time it is sampled. The increase in network lifetime is also desired to be shown with larger set of readings.

## V. CONCLUSION

We proposed a number of improvements that we claim would compensate for the unavailability of mobile devices. This is done by resorting to methods that assume a virtual node in a virtual sensor layer amidst existing nodes. Again, methods to improve query response efficiency have been stated. Future work includes proposals to reduce latency of the queries by improving over the stated methods. Ways to reduce bandwidth utilization can also be pursued by methods which take advantage of the cellular technologies and their mode of data transfer.

## VI. REFERENCES

[1] Aman Kansal, Feng Zhao. Location and Mobility in a sensor network of mobile phones, ACM SIGMM 17th International workshop on Network and Operating Systems Support for Digital Audio & Video (NOSSDAV '07), MSR-TR-2007-26.

[2] Yanif Ahmad, Suman Nath; 2008; COLR-Tree: Communication-Efficient Spatio-Temporal Indexing for a Sensor Data Web Portal; ICDE 2008. IEEE 24th International Conference on Data Engineering; Pages: 784-793.

[3] "Microsoft's plan to map the world in real time," MIT Technology Review, http://www.technologyreview.com/read\article.aspx?id= 16781&ch=infotech, May 2006.

[4] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," ACM Comput. Surv., vol. 31, no. 3, pp. 264–323, 1999.