Adaptive Sampling for Sensor Networks

Ankur Jain

Computer Science
University of California, Santa Barbara
Santa Barbara CA 93106
ankurj@cs.ucsb.edu

Edward Y. Chang

Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara CA 93106
echang@ece.ucsb.edu

Abstract

A distributed data-stream architecture finds application in sensor networks for monitoring environment and activities. In such a network, large numbers of sensors deliver continuous data to a central server. The rate at which the data is sampled at each sensor affects the communication resource and the computational load at the central server. In this paper, we propose a novel adaptive sampling technique where the sampling rate at each sensor adapts to the streaming-data characteristics. Our approach employs a Kalman-Filter (KF)-based estimation technique wherein the sensor can use the KF estimation error to adaptively adjust its sampling rate within a given range, autonomously. When the desired sampling rate violates the range, a new sampling rate is requested from the server. The server allocates new sampling rates under the constraint of available resources such that KF estimation error over all the active streaming sensors is minimized. Through empirical studies, we demonstrate the flexibility and effectiveness of our model.

1 Introduction

As sensor networks grow in size, bandwidth allocation becomes increasingly critical. A sensor network needs to allocate its bandwidth to maximize total information gain. A desirable bandwidth allocation scheme should distribute the given bandwidth such that it is sensitive to streaming data characteristics, query precision, available resources (communication, power, CPU), and sensor priority (data from some sensors might be more important than others) [9, 2]. We can further motivate this research using the following two examples.

- Wireless sensor-networks are being used for habitat monitoring applications. In [11], sensors registering light, temperature, and sound are deployed in burrows of Storm Petrels (a seabird) for monitoring purposes. During the day time, the burrows are expected to be empty, and thus we can have a low sampling rate. However, if some unusual measurements are recorded at some burrows (say abrupt increase in sound levels), it would be desirable to collect samples from them more frequently than the other burrows.

- In video surveillance applications like [6], multiple cameras are mounted at key locations to monitor activities of vehicles and people in a parking lot. If a camera shows a vehicle exhibiting unexpected behavior (random swirling, speeding), the camera's sampling rate should be increased by decreasing the sampling rates of the other cameras that are not observing abnormal behavior.

A naïve solution to the above-mentioned problems is over-sampling [12]. However this comes at increased cost of resources, namely:

- *CPU* — The CPU at the central server might have to process unnecessary data from numerous sources, but this would not affect the result significantly.

- *Network Bandwidth* — The communication channel would be transmitting unnecessary data. Moreover, in cases of low bandwidth networks the option of over-sampling might not be available at all.

- *Power Usage* — Power conservation is critical for wireless sensors. Over-sampling leads to increased power consumption of a sensor's measuring devices, radio transmitter, and processing unit.

Proceedings of the First Workshop on Data Management for Sensor Networks (DMSN 2004), Toronto, Canada, August 30th, 2004.
http://db.cs.pitt.edu/dmsn04/

There has been a significant amount of work in the sensor-network resource management. The key aspect that differentiates this work from the prior efforts lies in data collection (sensing). We adjust sampling rates (sensing rates) at sensors to adapt to data characteristics. Traditional methods (e.g., load-shedding [17] and adaptive precision setting [13]) collect data at a peak sampling rate and then determine whether collected data should be dropped to conserve resources. Even though the filtering and load-shedding approaches can reduce bandwidth consumption in the transmission phase, excessive sampling rates incur high cost in data collection and processing (to determine what data to drop) at the sensors. The adaptive sampling scheme proposed in this paper adjusts the data collection rate according to data characteristics. Therefore, resources are conserved and better utilized working only on data relevant to the queries.

Our *general* and *adaptive* sampling approach adjusts the sampling interval SI (the time interval between two consecutive samples) collectively. At the sensors, the SI is adjusted depending on the streaming data characteristics. The remote source is allowed to modify the sampling interval independently within a specified *Sampling Interval Range* (SIR). If the desired modification in the SI is more than that allowed by the SIR, a new sampling interval is requested from the server. At each sensor, we use the Kalman Filter estimator to predict the future values of a stream based on those seen so far. Large prediction errors signify unexpected behavior of the streaming data or an interesting event. The sampling interval is adjusted based on the prediction error. At the server, new sampling intervals are allocated to the requesting sensors based on available bandwidth, network contention, and streaming source priority.

We consider a simple network model to conduct the experiments, where all the streaming sources connect to a single network channel. The server continuously monitors the usage of this network channel and allocates new bandwidth based on its availability. These kinds of networks are prevalent in video surveillance, object tracking and process control (automated meter reading, building automation). Extending our current architecture to multi-hop sensor networks is a part of future research.

The main contributions of our work can be summarized as below:

- We propose a model which, is *adaptive* to adjust the sampling rate based on the input data characteristics and *general* to map to linear (as well as non-linear) problems without many major modifications.
- Our method utilizes the given bandwidth judiciously such that more important sources get more bandwidth by reducing the bandwidth of less important ones.
- Our method allows the *capability* at the remote site to adjust the sampling rate (to a certain extent) independently without the central server mediation to improve response time.
- Finally, we propose an *optimal estimation* scheme (Kalman Filter) that can be used on the sensor side to assess data arrival characteristics.

2 Related Work

The resource management problem in data streaming has been studied mainly from the perspective of data filtering [5, 13]. It has been shown that using adaptive precision bounds [13], unusual *trends* in the streaming data can be captured (the data is updated to the server only when it falls out of an adaptive precision bound) at low communication costs. However, due to uniform sampling, the approach does not have the capability to utilize a given bandwidth to maximize the information gain.

The adaptive sampling approach proposed in [10] considers only the network channel contention while adjusting the sampling rate. The sensors check for the network channel contention before putting the data on it and reduce the sampling rate if the contention and data-tuple drop rate is high. This reduces the overall load on the network channel and achieves a better delivery rate at the server. The proposed approach does not utilize the network channel judiciously, and it uses adaptive sampling only when the network channel becomes congested and requires load-shedding.

The use of adaptive sampling and bandwidth management in sensor networks has been very well motivated in [12, 3, 14, 9]. However a scalable method applicable in a distributed environment is still not available.

As we have discussed in Section 1, the problem of adaptive sampling is not the same as that of load-shedding [17]. First, to the best of our knowledge, none of the load-shedding techniques have yet used prediction/estimation models. Second, while load-shedding modules are activated only when the load on the system increases beyond what it can handle, adaptive sampling modules are executed during the lifetime of a stream. In the event of network congestion, the load-shedding module would reduce the data *transmission* rate of the sensor by randomly dropping tuples, whereas an adaptive sampling technique would reduce the data *collection* rate in such a way that higher priority data receive a higher proportion of the available bandwidth.

3 The Kalman Filter

The Kalman Filter was introduced in 1960 by R. E. Kalman [7] as a recursive solution to the discrete-data linear filtering problem. Since then, it has found application in the fields of data smoothing, process es-

timation, and object tracking, to name a few. The traditional Kalman Filter is a linear algorithm that estimates the internal state of a system based on a prediction/correction paradigm. Below, we provide a brief overview of the Kalman Filter's mathematical formulation, for more details refer [18].

The Kalman Filter comprises a set of mathematic equations that provide a recursive solution to the least-squares method. The system model is represented in the form of the following equations:

$$\boldsymbol{x_{k+1}} = \boldsymbol{\phi_k x_k} + \boldsymbol{w_k} \tag{1}$$

$$\boldsymbol{z_k} = \boldsymbol{H_k x_k} + \boldsymbol{\nu_k} \tag{2}$$

where

$\boldsymbol{x_k} =$ *state vector of the process*
$\boldsymbol{\phi_k} =$ *state transition matrix relating* x_k *to* x_{k+1}
$\boldsymbol{w_k} =$ *process model noise*
$\boldsymbol{z_k} =$ *measurement vector*
$\boldsymbol{H_k} =$ *matrix relating system state and measurement vector*
$\boldsymbol{\nu_k} =$ *measurement noise*
$k =$ *discrete time index*

The prediction $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ is based on a linear combination of previous prediction/estimation and the weighted prediction error. This error is called *innovation* $\boldsymbol{\psi_k}$, which is calculated as follows:

$$\boldsymbol{\psi_k} = \boldsymbol{z_k} - \boldsymbol{H_k}\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}. \tag{3}$$

The value of the weight is called *Kalman Gain* $\boldsymbol{K_k}$ which is adjusted with each measurement. The prediction is calculated as follows:

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} + \boldsymbol{K_k}\boldsymbol{\psi_k}. \tag{4}$$

Applying the least-square method we get

$$\boldsymbol{K_k} = \boldsymbol{P_k^-}\boldsymbol{H_k^T}(\ \boldsymbol{H_k}\boldsymbol{P_k^-}\boldsymbol{H_k^T} + \boldsymbol{R_k})^{-1}. \tag{5}$$

$$\boldsymbol{P_k} = (\boldsymbol{I} - \boldsymbol{K_k}\boldsymbol{H_k})\boldsymbol{P_k^-}. \tag{6}$$

where, $\boldsymbol{P_k}$ and $\boldsymbol{R_k}$ are the error covariance and measurement noise covariance matrices respectively (the superscript denoting the *a priori* state of the matrices).

The advantage of using the Kalman Filter is that it gives satisfactory results even when we cannot model the process accurately (i.e., when the values of matrices $\boldsymbol{\nu_k}$ and $\boldsymbol{\phi_k}$ are unknown) and that the innovation sequence can be used to evaluate the performance of the estimation process.

There is a wide spectrum of filtering solutions available which work on the estimation/correction paradigm and can be substituted for Kalman Filter in our proposed architecture. However, we support the use of Kalman Filter as it can be easily customized to provide good results on a wide range of streaming sensor data and produce *unbiased* estimates even when the incoming data have high variance. Biased algorithms (like Exponential Weighted Moving Average, EWMA) might not be the best choice when incoming data has high variance. Error estimates can be further improved using more sophisticated solutions like Particle Filter [8] or condensation (conditional density propagation) [4] as they work on non-Gaussian noise processes and multi-modal state propagation. Such algorithms are likely to provide better results as real-life data are not Gaussian, however this performance upgrade comes at increased cost of computational resources. Most of the sensing devices have limited computational capacity and selecting the best filtering solution is subject to the availability of the resources. The advantage of using Kalman Filter here is that the computational complexity can be easily manipulated by adjusting the number of state variables in the state propagation equation.

4 Our Approach

We now present our adaptive sampling approach in a distributed stream environment. We consider an environment where numerous sensors continuously stream updates to a central server. For example, a system of sensors that continuously measure the location of a moving object in two dimensions (one sensor for each object). Our adaptive approach would distribute the available bandwidth automatically in such a way that sensors monitoring objects showing increased activity have shorter time intervals between successive (low sampling interval) measurements whereas those with reduced activity have longer time intervals (high sampling interval). This way, the trajectory generated at the server by interpolating the measurements from the sensors would be closer to the original trajectory than that obtained by performing uniform sampling.

To maintain simplicity, we do not assume the presence of any data filtering or load-shedding modules in our discussion. Thus, the data-sampling interval is the same as the data-transmission interval of the sensor. We interpret the sampling interval as the number of time units between two successive measurements.

There are two main modules in the system, one on the sensor side and the other on the server side. Due to the space limitations, we describe each of them only briefly. To simplify our discussion, we assume in this paper that the tuple size over all the sources is the same, and hence the bandwidth consumption is directly proportional to the sampling interval at the streaming sources.

4.1 Source Side Module

Let SI_i denote the current sampling interval at source S_i (i is the i^{th} source) which, is the number of time units between two consecutive measurements. Let SIR_i denote the range within which, the sampling interval can be adjusted by the source without any server mediation and SI_i^{last} denote the latest value of sampling interval received from the server. (We currently assume static SIR_i's.) Let $SI_i^{desired}$ denote the desired sampling interval based on the KF prediction error. Sensor S_i need not contact the server for additional bandwidth provided that

$$(SI_i^{last} - SIR_i/2) \leq SI_i^{desired} \leq (SI_i^{last} + SIR_i/2). \quad (7)$$

If $SI_i^{desired}$ satisfies Equation 7 then SI_i takes the value of $SI_i^{desired}$. This scheme helps the source to capture unexpected data trends immediately as the server grants over the network could be delayed due to network congestion or unavailability of resources. Each data tuple sampled by S_i is forwarded to a Kalman Filter KF_i which, provides with the innovation $\boldsymbol{\psi_t^i}$ value (Section 3). The estimation error δ_i at any instant t is then calculated as:

$$\delta_t^i = sqrt(trace(\boldsymbol{\psi_t^i}(\boldsymbol{z_t^i})^{-1})^2). \quad (8)$$

We multiply the innovation (error in prediction) by the inverse of the measurement matrix to get the fractional error ($\boldsymbol{\psi_t^i}$ and $\boldsymbol{z_t^i}$ are column matrices). We take the square of the matrix to eliminate any negative values. Finally the square root of the trace gives the fractional error over all the variables in the measurement matrix.

S_i maintains a sliding window of size W_i that holds the last W_i values of the estimation error. If n_j^i is the j^{th} element of the sliding window at S_i (n_1^i being the latest element), total error Δ_i over the sliding window is calculated as:

$$\Delta_i = \frac{\sum_{j=1}^{j=W_i} n_j^i/j}{\sum_{j=1}^{j=W_i} 1/j}. \quad (9)$$

Equation 9 ensures that the newer values in the window have higher weight.

User parameters λ_i and θ_i control the dynamics of SI_i. Each time Δ_i is calculated, a new sampling interval SI_i^{new} is generated as follows:

$$SI_i^{new} = SI_i + \theta_i * (1 - e^{f_i}). \quad (10)$$

where $f_i = \frac{\Delta_i - \lambda_i}{\lambda_i}$. Equation 10 ensures sharp fall and gradual rise in the sampling interval due to the exponential factor that helps improving the response time of the system. If SI_i^{new} satisfies Equation 7, then the sampling interval is assigned this new value; otherwise, a new sampling interval is requested from the server. The source requests the change is the sampling interval ΔSI_i such that

$$\Delta SI_i = SI_i^{new} + SIR_i/2. \quad (11)$$

In addition the source also sends the fractional error f_i for each request of decrease in the sampling interval.

4.2 Central Server module

We now discuss the sampling rate allocation policy at the central server. The allocation algorithm is executed each time a request for decrease in sampling interval (increase in the sampling rate) is received from a streaming source. The server maintains a variable R_{avail} that holds the amount of communication resource available at any time. When a source reports about an increase in its sampling interval, the server immediately adds the proportional amount of resource units to R_{avail} and sends an acknowledgment to the source. Any request for a decrease in a sampling interval is added to a job-queue that is processed continuously by a separate thread.

Each job J_p in the job-queue has 5 attributes which, are described below:

1. *Fractional error* f_p is received from the source when it sends a request.
2. *Request* Req_p is the units of resource requested.
3. *History* h_p is the age of the request in the job-queue. Its value is incremented by unity each time the job-queue is processed.
4. *Grant* g_i is the fraction by which, the Req_p has been satisfied so far.
5. *Query Weight* w_p the weight of the streaming source from the query evaluator.

Assuming that the error f_p is reduced to zero if resource request J_p is satisfied completely, we can formulate a linear optimization problem, minimizing the total error over all the jobs. If J_p is allocated A_p units of resources, then the residual error after satisfying the job is proportional to $(1 - A_p/Req_p)$. Jobs having higher f_p, h_p, and w_p are given more priority than others, whereas the priority varies inversely with g_p. We normalize each attribute by dividing it by the sum of its value in all the jobs in the job-queue. Thus the objective function can be formulated as:

$$\min_{A_p} \left(\frac{f_p}{\sum f_p} * \frac{h_p}{\sum h_p} * \frac{w_p}{\sum w_p} * \frac{\sum g_p}{g_p} * \left(1 - \frac{A_p}{Req_p}\right)\right) \quad (12)$$

$$\text{s.t.} \begin{cases} \sum A_p \leq & R_{avail} \\ 0 < A_p \leq & Req_p \end{cases} \quad (13)$$

Constraints in Equation 13 ensure that the sum of the allocated resources is less than that of the total available and that each grant is less than its request. Once

the optimization problem is solved, the resource units are distributed to the requesting sources and the job-queue attributes are updated accordingly.

5 Results

In this section we present the preliminary results of our distributed adaptive sampling system. We performed the experiments on data produced by the oporto realistic spatio-temporal data generator [15]. We recorded the trajectories (in 2 dimensions) of 12 shoals produced by the generator for $3,000$ time units. Oporto produces data with uniform distribution and some of the trajectories were more complex than the others.

We implemented our system and conducted the experiments on a Pentium III processor workstation with 256MB of RAM on a 10/100 Mbps LAN. The coding was done on JDK 1.2.4, using JAMA [1] matrix package for matrix operations and OR-Objects [16] package to solve the LP problem.

We initialized different streaming sources with different trajectories but the same initialization parameters using a linear KF model [5]. All the sources had to wait until the sliding window was full. We ran the simulation until one of the sources had read all the $3,000$ records. The tuples received at the server with their timestamps were then used to create the complete trajectory using linear interpolation for both X and Y coordinates. We evaluated the performance of our system based on an *effective resource utilization* (ERU) metric ξ which, is calculated as

$$\xi = \eta * m \tag{14}$$

where m is the fraction of messages exchanged between the source and the server, to the total number of tuples read by the source, where η is the mean fractional error between the actual trajectory and that generated by interpolation. While calculating m we considered the number of tuples forwarded by the source, messages for bandwidth allocation and acknowledgment messages from the server to the source. In all the experiments θ_i=2, the initial sampling interval was five tuples and none of the sources were allowed to skip more than 12 tuples in the adaptive sampling module. We studied the affect of the number of sources, sliding window size W_i and λ_i on the ERU. Results shown in Figures 1, 2, and 3 were obtained using $W_i = 5$ and $\lambda_i = 0.6$.

Figure 1 shows the mean fraction of messages forwarded to the main server against the number of sources. In this figure m is low for a small number of sources, but as the number of sources increases, it rises and stabilizes around 0.12. In all the cases the number of messages is less than or equal to that sent using uniform sampling.

Figure 2 shows that the fractional error using adaptive sampling is always less than that using uniform sampling, except when the number of sources is one. This is because we initialize the experiments with same

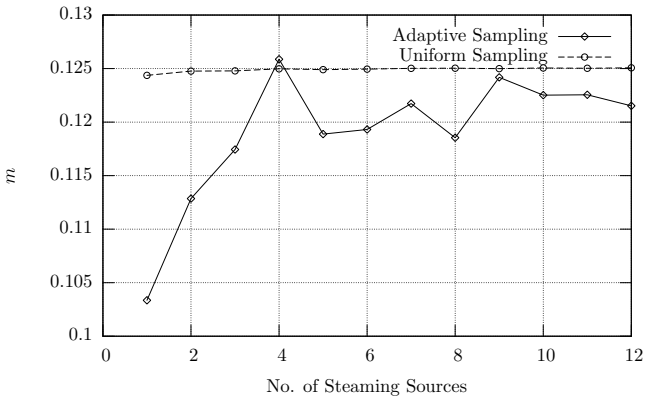

Figure 1: m on varying # of streaming sources

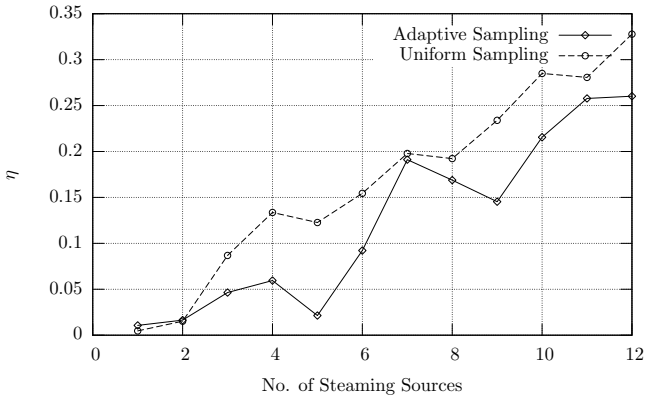

Figure 2: η on varying # of streaming sources

sampling interval for both uniform and adaptive methods. Thus if the number of sources is one, then we cannot beat the uniform sampling method.

Figure 3 shows change in ERU on varying the number of sources. The trend is similar to that in Figure 2. We observe that our approach outperforms the uniform sampling method even when the number of sources is high. There are some unusual results when the number of sources is five and seven. This is because the trajectories for these sources of input data may be unusually simple/complex.

Figure 4 shows the affect of parameter λ_i on the ERU. Resource utilization is high for very low values of λ_i because although the error rate would be low, the number of messages would be very high. We observed lower ERU when λ_i varied around 0.4 and 1.2. This is because at lower values of λ_i the error is low and thus ERU is low, on slightly higher values, although the error is high, the value of m drops down significantly enough to reduce the resource utilization below that of uniform sampling. However at further increasing the value of λ_i, η_i starts to dominate and the ERU starts to increase.

The effect of varying the sliding window size is shown in Figure 5. It is observed that at low values

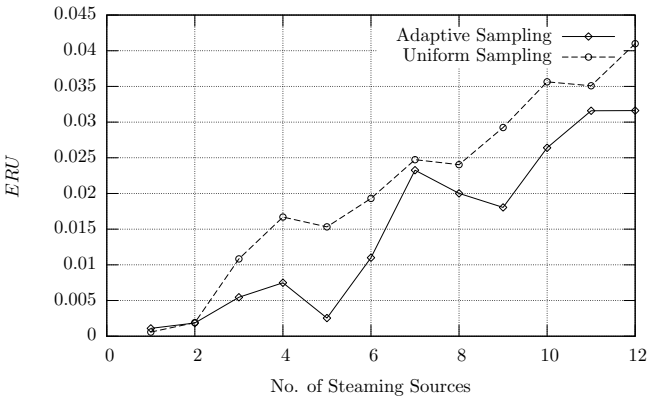

Figure 3: ERU on varying # of streaming sources

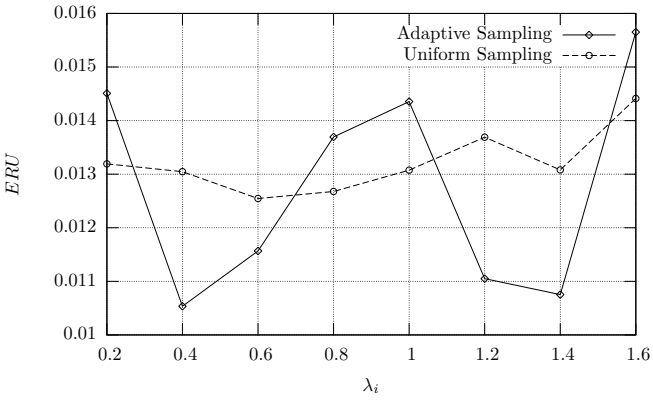

Figure 4: ERU on varying λ_i

of W_i give better ERU. As the window size increases, the ERU approaches constant value.

6 Conclusions and Future Work

In this paper we have proposed an adaptive sampling technique based on a Kalman Filter estimation of error as an alternative to commonly used uniform sampling techniques. We motivated the need for adaptive sampling techniques in a sensor network environment, where network bandwidth is a valuable resource. Adaptive sampling was shown to be desirable not only to conserve resources but also to improve the overall quality of results (minimize the fractional error between the actual and the interpolated results).

We discussed some of the preliminary results in Section 5 to show the effectiveness of our approach. We observed that when we choose the input parameters judiciously, our system can provide performance upgrade as much as three to four times as compared to uniform sampling (Figure 3). We have also shown the effect of different input parameters on the system performance which, suggests that further research needs to be conducted to enable us to choose optimal parameters for the system.

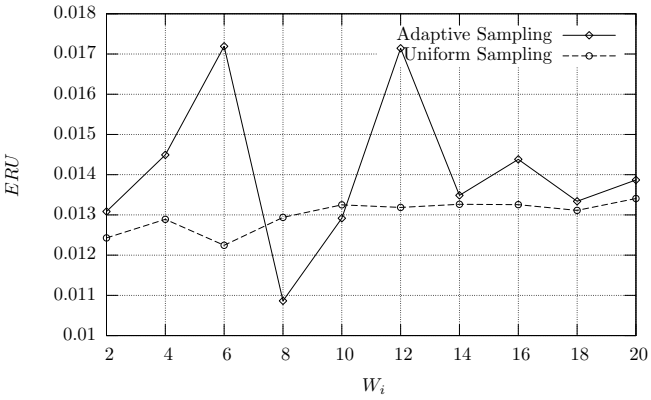

Figure 5: ERU on varying W_i

Our preliminary results are encouraging but further research is indicated in the following directions:

- Extending the current architecture to multi-hop sensor networks.
- Choosing appropriate window size.
- Developing efficient techniques to compute the error over the sliding window. In some cases exponential decay methods might provide better results.
- Developing efficient algorithms to reduce the request/acknowledge message overhead between the server and the sources. (Currently the message overhead is high.)
- Developing algorithms to incorporate adaptive SIRs in the current system.
- Testing the system performance on more real life data sets.

References

[1] R. F. Boisvert, B. Miller, R. Pozo, K. Remington, J. Hicklin, C. Moler, and P. Webb. Jama : A java matrix package.

[2] P. Bonnet, J. E. Gehrke, and P. Seshadri. Towards sensor database systems. In *Second Intl. Conf. on Mobile Data Management*, Hong Kong, January 2001.

[3] B. Hull, K. Jamieson, and H. Balakrishnan. Bandwidth managment in wireless sensor networks. In *Intl. Conf. on Embedded Networked Sensor Systems*, Los Angeles, California, USA, November 2003.

[4] M. Isard and A. Blake. CONDENSATION – conditional density propagation for visual tracking. *International Journal Computer Vision*, 1998.

[5] A. Jain, E. Chang, and Y. F. Wang. Adaptive stream resource management using kalman filters. In *Proceedings of the 2004 ACM SIGMOD Intl. Conf. on Management of Data*, 2004.

[6] L. Jiao, Y. Wu, G. Wu, E. Y. Chang, and Y. F. Wang. The anatomy of a multi-camera security serveillance system. *ACM Multimedia System*, 2004.

[7] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[8] M. K.Pitt and N. Shephard. Filtering via simulation: auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, June 1999.

[9] I. Lazaridis, Q. Han, X. Yu, S. Mehrotra, N. Venkatasubramanian, D. V. Kalashnikov, and W. Yang. QUASAR: Quality aware sensing architecture. *ACM SIGMOD Record*, 33(1):26–31, Mar. 2004.

[10] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD Intl. Conf. on Management of Data*, pages 491–502. ACM Press, 2003.

[11] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM Intl. Workshop on Wireless Sensor Networks and Applications,WSNA*, Atlanta, Georgia, USA, September 28 2002.

[12] A. D. Marbini and L. E. Sacks. Adaptive sampling mechanisms in sensor networks. In *London Communications Symposium*, London, UK, 2003.

[13] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proc.of ACM SIGMOD Intl. Conf. on Management of Data*, San Diego, California, USA, June 2003.

[14] J.-Y. Pan and S. S. amd Christos Faloutsos. Fastcars: Fast, correlation-aware sampling for network data mining. In *GLOBECOM 2002 - IEEE Global Telecommunications Conf.*, pages 2167–2171, Taipei, Taiwan, November 2002.

[15] J.-M. Saglio and J. Moreira. Oporto: A realistic scenario generator for moving objects. *GeoInformatica*, 5(1):71–93, 2001.

[16] D. Systems. OpsResearch: OR-Objects. http://www.opsresearch.com.

[17] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in data streams. In *29th Intl. Conf. on Very Large Data Bases (VLDB)*, pages 309–320, Berlin, Germany, September 2003.

[18] G. Welch and G. Bishop. *Introduction to Kalman Filter*. http://www.cs.unc.edu/∼welch/kalman, 2002.