Adaptive Stream Resource Management Using Kalman Filters

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

Edward Y. Chang
Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106
echang@ece.ucsb.edu

Yuan-Fang Wang
Computer Science
University of California
Santa Barbara, CA 93106
yfwang@cs.ucsb.edu

ABSTRACT

To answer user queries efficiently, a stream management system must handle continuous, high-volume, possibly noisy, and time-varying data streams. One major research area in stream management seeks to allocate resources (such as network bandwidth and memory) to query plans, either to minimize resource usage under a precision requirement, or to maximize precision of results under resource constraints. To date, many solutions have been proposed; however, most solutions are ad hoc with hard-coded heuristics to generate query plans. In contrast, we perceive stream resource management as fundamentally a filtering problem, in which the objective is to filter out as much data as possible to conserve resources, provided that the precision standards can be met. We select the Kalman Filter as a general and adaptive filtering solution for conserving resources. The Kalman Filter has the ability to adapt to various stream characteristics, sensor noise, and time variance. Furthermore, we realize a significant performance boost by switching from traditional methods of caching static data (which can soon become stale) to our method of caching dynamic procedures that can predict data reliably at the server without the clients' involvement. In this work we focus on minimization of communication overhead for both synthetic and real-world streams. Through examples and empirical studies, we demonstrate the flexibility and effectiveness of using the Kalman Filter as a solution for managing trade-offs between precision of results and resources in satisfying stream queries.

1. INTRODUCTION

In a Data Stream Management System (DSMS), *sensors* deliver continuous, high-volume, possibly noisy, and time-varying streams to a *central server* [5, 16, 21]. Efficient resource management is critical for a DSMS to achieve high-throughput performance. To conserve resources—network bandwidth, storage, and CPU—many recent papers [1, 2, 6, 13, 23, 30] propose methods to reduce the amount of data delivered to the server. If the server can answer queries within specified precision constraints, these methods do not enact data communication. Indeed, these methods have been shown effective for reducing network-bandwidth consumption, thereby also conserving the storage and processing loads at the server.

A major shortcoming of the existing solutions, however, is that they are often ad hoc, as explained in [2], and have been highly application-dependent. No unified solution has yet been developed for managing stream resources. In this paper, we treat stream resource management as fundamentally a filtering problem. An effective stream-filtering algorithm should filter out a maximum amount of data as long as the precision constraints are met at the server.

We introduce our *Dual Kalman Filter* (DKF) architecture as a general and adaptive solution to the stream-resource-management problem. We advocate the use of the Kalman Filter (KF) [18] for stream-filtering, since KF has been well studied and widely applied to many data filtering and smoothing problems. Specifically, we incorporate the Kalman Filter as the basic building block of a stream management system for the following two reasons:

- Traditional methods cache *static* data that easily become stale over time. This necessitates frequent and expensive synchronization between clients and servers through retransmission. Our method, on the other hand, caches filter parameters that enable *dynamic* and *accurate* system prediction on the server without clients' intervention.
- As will be fully explained in Section 3.2, the Kalman Filter can be easily customized to handle varying stream characteristics, sensor noise, and time variance to meet the requirements specified in [21]. The same filtering framework can be adapted to address a wide variety of stream resource management problems, providing a unified paradigm that is both powerful and versatile.

In this paper, due to the space limitations, we focus on showing the use of the KF to conserve network bandwidth (and hence the storage and processing resources at the server). To further emphasize the need to conserve network bandwidth, let us consider a typical wireless sensor monitoring system. In applications such as moving-object tracking, weather monitoring, and video surveillance, the power dissipation rate of a wireless sensor-node is an issue of primary

SIGMOD 2004, June 13-18, 2004, Paris, France.

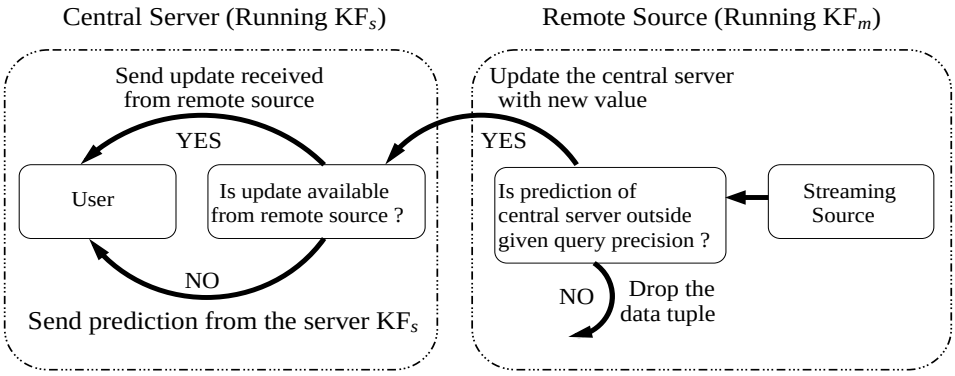

Figure 1: The DKF model

concern. It has been established [28, 35] that the majority of power dissipation occurs when transmitting bits over wireless networks, not when processing them. The ratio of energy spent in sending one bit over networks to that spent in executing one instruction is between 220 to 2,900 on various architectures [26, 27]. Thus, filtering data at sensors is beneficial not only for conserving bandwidth, but also for conserving power. Since the computational cost incurred by KF is insignificant in many practical sensing scenarios, KF is an attractive option as the filtering solution for resource conservation.

Figure 1 depicts the role of our proposed DKF model in a typical DSMS architecture. A user (on the left-hand side of the figure) issues a query to the server with some precision constraints. The server activates a KF, denoted as KF$_s$, and at the same time, the target sensor activates a mirror KF with the same parameters, denoted as KF$_m$. The dual filters KF$_s$ and KF$_m$ predict future data values. Only when the filter at the remote source, KF$_m$, fails to predict future data within the precision constraint (and thus KF$_s$ cannot provide an accurate prediction at the server) that the sensor sends updates to KF$_s$. Significant bandwidth conservation can be achieved if a reliable and accurate data prediction mechanism is employed. We propose the KF as such a mechanism for its simplicity, efficiency, and provable optimality under fairly general conditions.

1.1 What is Kalman Filter?

The Kalman Filter is a stochastic, recursive data filtering algorithm. It has been widely used in predicting a system's internal state based on the observation of its external behavior. For stream management applications, a stream is modeled as a generative process (a streaming model) controlled by the stream's internal parameters (state), which evolve over time. The state may or may not be directly observable, and hence, has to be inferred by observing the system's external behavior. The observed data stream serves as the external observation that is used to estimate the internal state of the stream's generative process. The state estimation process operates using recursive steps of prediction (propagating the internal state of the system) and correction (fine-tuning the prediction with external observation) [10, 18, 32].

A concrete example is the problem of estimating the state of a moving vehicle. The system state comprises the current location and velocity of the vehicle, and is represented by a vector $\boldsymbol{x_k}$, at a discrete time step k. The system evolves over time due to the driver's acceleration and braking actions, and road friction of a random. Hence, the time evolution of the system's state is governed by the equation

$$\boldsymbol{x_{k+1}} = \boldsymbol{A x_k} + \boldsymbol{B} a_k + \boldsymbol{w_k}$$

$$\begin{bmatrix} p_{k+1} \\ c_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_k \\ c_k \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} a_k + \boldsymbol{w_k} \quad (1)$$

where $\boldsymbol{w_k}$ is process noise, matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ relate the state at k to that at $k+1$, a_k is the time-varying acceleration, c_k is the velocity, and T is the time between step k and step $k+1$. Now, suppose at discrete time intervals we can measure the position p. Then our measurement at time k can be denoted as

$$z_k = p_k + \nu_k \quad (2)$$

where ν_k is the measurement noise inherent in all measurement processes.

Given hints on the system state through the state propagation and external observation mechanisms, the KF integrates all the information to arrive at the best estimate of the system state over time. The KF weighs all available information by taking into consideration the noise in external measurement and the uncertainty in state propagation. More specifically, let us assume that the state propagation uncertainty $\boldsymbol{w_k}$ is white Gaussian with a covariance matrix $\boldsymbol{Q}$. The measurement noise ν_k is white Gaussian with a covariance matrix $\boldsymbol{R}$, and it is not correlated with the noise in state propagation. The formulation of the KF algorithm provides us with the following statistical properties:

1. The expected value of the KF estimate is equal to the expected value of the state. That is, *on average*, the estimate of the state will equal the true state. Or the KF is an unbiased estimator.

2. Of all linear estimation algorithms, the KF algorithm minimizes the variance of the square of the estimation error. That is, *on average*, it gives the *smallest* possible variance in estimation error. Or the KF is the linear estimator that can deliver the most consistent estimation results.

Continuing with our example of tracking a vehicle as it moves in a two-dimensional space, the vehicle might be able to provide rapid updates of its position to the central server as it moves (e.g., if it is equipped with a GPS positioning system). An inherent limitation is that the exact object location cannot be updated continuously due to the limited

bandwidth and battery power of the remote devices. Thus, approximate answers to the queries of the vehicle's position are acceptable. A promising solution is to maintain a precision bound width δ at the remote source and update the server whenever the true value deviates more than δ units from the server value as proposed in [23, 25]. We use our dual Kalman Filter approach to accomplish this (system equations are discussed in detail in Section 4).

Again, let us use the schematic diagram in Figure 1 to explain. Suppose a user query comes with a precision constraint δ. Our system will activate a Kalman Filter KF_s at the central server and its mirror Kalman Filter KF_m at the remote site. The remote source keeps track of the server prediction at time step k (note that this does not require any extra memory except for the usual matrices of the KF) and filters out the data (does not forward it to the central server) if the prediction at the central server deviates by less than a margin of δ. Notice that KF_s, after receiving the first few measurements from the remote source, would have established a good estimate of the state vector (p_k, c_k). Based on that, the server can compute the rate of change of the X and Y locations using Eq 1, and would require fewer updates from the remote source. Updates are needed only when sudden acceleration or braking actions induce a large error in the state estimate, or when noise gradually corrupts the state prediction to such a degree that a refresh is necessary. For tracking and recreating a vehicle's locations, the server does not need to record any information other than the X and Y coordinates of the vehicle and the Kalman Filter matrices.

1.2 Contribution Summary

In addition to perceiving and formulating stream resource management as fundamentally a filtering problem, and proposing using the Kalman Filter as a general and adaptive solution, the specific contributions of this paper are as follows:

1. We present a comparative analysis of our model versus the existing techniques and discuss different applications where the Kalman Filter has been successfully incorporated (Section 2).

2. We present the mathematic formulation of the filter and discuss how the filter formulation can be easily customized for a large variety of problem formulations in stream management. (Section 3).

3. We propose our dual Kalman Filter model (in Section 3.1) and discuss how constraints (query precision and smoothing factors) provided by the user are used to install and set initial parameters for the Kalman Filters.

4. Through examples and empirical studies (in Sections 4 and 5), we show that employing the Kalman Filter can facilitate and support different query scenarios. In terms of bandwidth conservation, the Kalman Filter is at least as good as traditional approaches, if not better. More important, its generality and adaptivity show promise as a building block for stream applications that concern fusion and integration.

5. In Section 6 we advance a set of promising extensions to build upon this work.

2. RELATED WORK

Algorithms in data streams have received increased attention over the past two years. A comprehensive survey of the issues in data streams is presented in [5, 16]. Major research directions include conserving computational and communication resources [1, 23], optimal storage algorithms, stream mining [36], query optimizers [5], and query solvers [17]. Data streams have also been treated as time series, and ideas from control theory were borrowed for the purposes of approximation [11] and mining [36].

Resource management in data streams is of prime concern due to their unbounded, continuous, and time-varying data-arrival characteristics. Furthermore, memory and communication resources form bottlenecks in a stream management system's performance as they are affected adversely by the above stated properties of data streams. There has been a considerable amount of research work in the direction of memory management for processing data stream queries [3, 7, 4], where the objective is to answer a given query using a minimum amount of memory under the given constraint of precision or characterize the memory requirements for the query.

Research work proposed in [23] (STREAM project [2]) provides an insight into the distributed-sources data streaming problem where adaptive filters are installed at each remote source updating a central server. The goal is to achieve maximum precision given the constraint of network bandwidth. The server caches the latest data received from the sensor until a new value is received. The caching model, though effective, is not adaptive to the characteristics of the input stream attribute values. For example, if a streaming attribute value shows a continuous ascending/descending linear trend over a span of time, this model would generate a high number of updates. This is because new precision bounds are not adjusted adaptively (based on time-varying characteristics) each time a streaming attribute value comes out of its *precision bound*, thereby increasing the probability of future updates. We propose a *general* and *flexible* model in which the server would predict the future values based on the current trend in the input value. So if the streaming values exhibit a linear trend, the Kalman Filter, after a few updates, would predict values based on the slope of the linear curve. We characterize the streaming model using state equations in order to save more bandwidth by predicting its behavior.

The problem of reducing network traffic can also be realized in terms of *load shedding*, where data tuples are dropped from the system when the load (total volume of data) increases to an extent that it cannot be processed completely. This problem has been studied in [30] (AURORA project [1]). The load at each *box* in the network is maintained, and *drop operators* are installed if the load reaches a threshold. The load is reduced by altering the sampling technique, changing the sampling rate or modifying selectivity criterion. We note that *load shedding* is not the solution to effective resource management, as it drop chunks of data from the input stream independent of the stream data arrival characteristics. The problem that this system solves is different from ours, since it tries to stream maximum updates for a given network load whereas our solution aims to stream minimum updates so that precision constraint is satisfied at all times.

Another approach for reducing communication overhead

System	Proposed Solution	Pros of using Kalman Filter
STREAM	Adaptive precision bounds, approx. value cached at server,does not work for noisy data (no data smoothing), dynamic precision widths are cached at the server, best estimate for future is the last cached value at the main server	Prediction algorithm can be used reduce communication overhead even further, on-line data smoothing helps to provide query answers even for noisy data
AURORA	Static precision widths, resource management using dynamic sampling rates based on *loss/gain* ratios, bounds do not change with input characteristics of the stream	Prediction mechanism is based on input characteristics, output is *sensitive* to input values
COUGAR	Partial query processing in the wireless network to prevent unnecessary data being forwarded (load shedding). Does not use any approximation or caching scheme	Prediction scheme gives better results, reducing load adaptively rather than dropping chunks of data indiscriminately

Table 1: Summary of existing solutions and advantages of using the Kalman Filter

has been proposed in the Cornell University COUGAR project [35]. An in-network aggregation scheme is used to perform partial aggregation of results at some intermediary node in the sensor network that prevents unnecessary data from being forwarded further ahead. The Telegraph adaptive dataflow system [12] is more focused on common sharing of resources for different queries, than on the trade-off between performance and resources.

A comparative overview of our work with three major data stream projects is presented in Table 1. None of the compared approaches use a prediction scheme and they do not seem to gracefully degrade when the input data is noisy. Furthermore, they cannot exploit partial information about the stream arrival characteristics (if available) to boost their performance. In contrast, our general framework can be applied to any streaming application by simply modifying the state transition matrix used in the Kalman Filter formulation (details in sec 3).

Research work proposed in [19] models the data stream from a sensor as a time series and uses an estimation mechanism to predict the values of the time series ahead of time. Our work differs in proposing a general prediction framework and an adaptive solution, which is *truly online*.

3. THE KALMAN FILTER

The Kalman Filter was introduced in 1960 by R. E. Kalman [18] as a recursive solution to the discrete-data linear filtering problem. Since then, it has found applications in the fields of process control [14], multi-sensor data fusion [33], motion-tracking [32], network-time keeping [8], and neural information processing [34], to name a few. The traditional Kalman Filter is a linear algorithm that estimates the internal state of a system based on two mechanisms [32]:

- *Prediction/Estimation*
  This step is used to propagate the internal state of the system. At time step k, the filter predicts the value of the internal state vector at the next time step $k+1$.

- *Correction*
  This step is responsible for fine-tuning the prediction step under the influence of external observations. At time $k+1$ when an actual measurement is available, the filter corrects itself based on the prediction error. This correction is done by minimizing the error covariance.

The Kalman Filter comprises a set of mathematic equations that provide a recursive solution to the least-squares method. We now introduce the mathematic notations common to the Discrete Kalman Filter [10]. The system model is represented in the form of the following equations:

$$\boldsymbol{x_{k+1}} = \boldsymbol{\phi_k x_k} + \boldsymbol{w_k} \tag{3}$$

$$\boldsymbol{z_k} = \boldsymbol{H_k x_k} + \boldsymbol{\nu_k} \tag{4}$$

where

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

The covariance matrices for the $\boldsymbol{w_k}$ and $\boldsymbol{\nu_k}$ vectors are as follows:

$$\boldsymbol{E}[\boldsymbol{w_k w_i^T}] = \left\{ \begin{array}{cc} \boldsymbol{Q_k} & i = k \\ 0 & i \neq k \end{array} \right\}, \tag{5}$$

$$\boldsymbol{E}[\boldsymbol{\nu_k \nu_i^T}] = \left\{ \begin{array}{cc} \boldsymbol{R_k} & i = k \\ 0 & i \neq k \end{array} \right\}, \tag{6}$$

$$\boldsymbol{E}[\boldsymbol{w_k \nu_i^T}] = \quad 0 \quad \textit{for all } k \textit{ and } i. \tag{7}$$

The prediction $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ is based on a linear combination of previous prediction/estimation $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}$, and the weighted difference of the measurement and its estimate (called *innovation*). The value of the weight is adjusted with each measurement and is called the *Kalman Gain* $\boldsymbol{K_k}$. The prediction is calculated as follows:

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} + \boldsymbol{K_k}(\boldsymbol{z_k} - \boldsymbol{H_k}\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}). \tag{8}$$

The estimation error is defined as

$$\boldsymbol{e_k^-} = \boldsymbol{x_k} - \hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}, \tag{9}$$

and the *a priori* error covariance as

$$\boldsymbol{P_k^-} = \boldsymbol{E}[\boldsymbol{e_k^- e_k^{-T}}]. \tag{10}$$

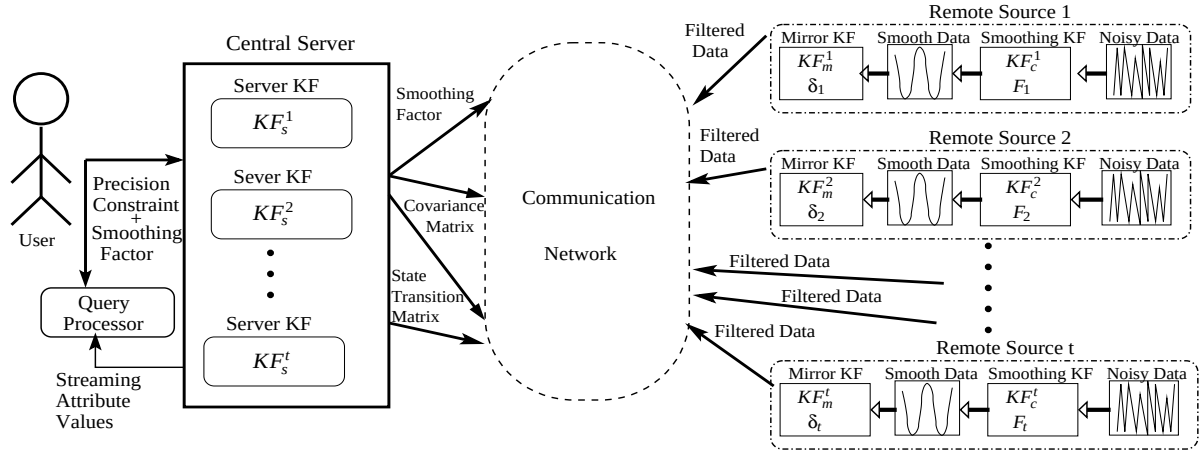

Figure 2: Architecture of DKF model

Applying the least squares method we get

$$\boldsymbol{K_k} = \boldsymbol{P_k^-}\boldsymbol{H_k^T}(\ \boldsymbol{H_k}\boldsymbol{P_k^-}\boldsymbol{H_k^T} + \boldsymbol{R_k})^{-1}, \tag{11}$$

and the *a posteriori* error covariance as

$$\boldsymbol{P_k} = (\boldsymbol{I} - \boldsymbol{K_k}\boldsymbol{H_k})\boldsymbol{P_k^-}. \tag{12}$$

Thus each prediction-correction step in the algorithm computes the value of $\boldsymbol{P_k}$ and $\boldsymbol{K_k}$, updates the system with these values, and obtains the next prediction $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ accordingly.

3.1 The Dual Kalman Filter Model

The architecture of our dual Kalman Filter model is depicted graphically in Figure 2, and the notations used are tabulated in Table 2.

The problem of adaptive setting of precision widths at individual sources has already been explored in [23]. The focus of this work is to adaptively predict the values at the main server with minimal information updates from the actual remote source. When a *continuous query* q_j with a precision constraint Δ_j is presented to the server on source object s_i, a Kalman Filter KF_s^i is installed at the main server. A mirror KF, KF_m^i, is activated at the remote source, which simulates the operation of KF_s^i. For the sake of simplicity, we assume that $\Delta_j = \delta_i$ and that we do not have queries with overlapping sources.

Symbol	*Significance*
t	Total number of sources
u	Total queries
s_i	Streaming source object i $(i = 1 \cdots t)$
q_j	Query j $(i = 1 \cdots u)$
Δ_j	Precision width of q_j $(i = 1 \cdots u)$
δ_i	Precision width at s_i $(i = 1 \cdots t)$
F_i	KF Smoothing factor (optional) at source s_i
KF_s^i	KF at central server predicting attribute value at s_i
KF_m^i	Mirror KF at s_i simulating KF_s^i
KF_c^i	KF at s_i used for data smoothing (optional)
v_i^k	Actual value of streaming attribute at s_i at time instant k
$\hat{v}_i^k$	Prediction of attribute value at s_i from KF_s^i at time instant k
$\hat{v}_i^{k-1} - v_i^k$	Innovation sequence from KF_s^i at s_i

Table 2: Symbols and their meanings

At the main server we have as many filters running as the number of remote sources t. Since KF is not very expensive computationally (typically in sensor systems the number of state variables and the rank of the matrices are low), we assume that having multiple Kalman Filters at the main server does not affect the performance significantly. Both filters, KF_m^i and KF_s^i, operate on the same parameters and precision width δ_i. Latest values from the remote source are updated to the main server only when the prediction error exceeds the given precision (i.e., when $\mid \hat{v}_i^{k-1} - v_i^k \mid > \delta_i$). The user can also provide an optional parameter F_i with the query. This parameter controls the extent to which the data is smoothed by KF_c^i, before the attribute is considered by KF_m^i. When the input streaming value is noisy and the *average* values are desired for query answering, F is useful in providing *fine-grain control* to the user, over the *sensitivity* of the result.

This approach saves communication overhead in the following way. Suppose we are tracking a moving object. If the object is moving in a particular direction, then it is likely to continue moving in the same direction in the near future. Thus after a few measurements the filter may not require any further measurement updates until the object performs some maneuvers to change the motion pattern, or the noise corrupts the estimation process significantly. However, this should occur infrequently if the state model is correct and the sensor measurements reliably predict the state.

Our approach can be useful in data streaming and provide the following advantages :

1. It is more effective than caching approximate results [23, 25] at the main server, as KF provides time-varying estimated values of the stream attribute at a lower communication cost.

2. Due to flexibility and generality of the filter we can model different processes (even non-linear models) accurately, applying the same framework to different applications.

3. F_i gives *fine-grain* control on the level of data smoothing and thus control over the *sensitivity* in the query result, which is more effective than the moving average approach.

4. From experimental results we show that even when

the state transition equation is not known, DKF can perform sufficiently well.

5. The innovation sequence ($\hat{v}_i^{k-1} - v_i^k$) helps in detecting outliers and adaptively adjust the sampling rate.

6. It is relatively simple to change the state equations dynamically, as they are parameters to the system.

3.2 Why Kalman Filter?

In casting stream resource management into a filtering problem, or more specifically a Kalman filtering problem, it is necessary to examine the applicability of the Kalman Filter under a wide variety of possible problem formulations. The Kalman Filter is a stochastic, recursive estimator, widely used in estimating the internal state of a system based on the observation of the system's external behavior. In essence, what a Kalman Filter does is to maintain and update the best estimate of the internal state by properly weighing and combining all available data (state prior and external measurements) to form an educated guess. It accomplishes this feat by propagating the state estimate forward in time (a *prediction* process), incorporating external measurements whenever they are made available to update the state estimate (a *correction* process), and repeating these steps recursively and continuously over time. Or the Kalman Filter operates a prediction-correction mechanism for state estimation based on the Bayesian principle.

Many variations are possible in formulating a resource management problem under the umbrella of the *prediction-correction* paradigm. What is unique about the Kalman Filter, as we will illustrate below, is that the Kalman Filter can be customized to form workable solutions for *all* these formulations. The adaptation involves both simplification (e.g., static Kalman Filter or recursive least squares) and generalization (e.g., extended Kalman Filter).

Variation in problem formulations can come from (1) the state, whether it is directly observable or not, (2) the state propagation equation, whether it is linear or not, (3) the external measurement equation, whether it is linear or not, (4) the external measurement, whether it comes with a confidence value or not, and finally (5) the noise processes, whether they are stationary or not. We will discuss these five system design parameters in greater detail below.

The special case in which the state is directly observable (case 1 above) does not pose any problem. This is the case when the resource to be managed is directly measurable (e.g., CPU or memory usage). In this case, the $\boldsymbol{H}$ matrix that relates the internal state to the external measurement is an identity matrix. In other cases, the Kalman Filter formulation applies unchanged.

The standard Kalman Filter assumes that both the state propagation equation and the external measurement equation are linear [20]. This is the case for the first example in our experiments (Section 4.1), where the position and velocity of a mobile platform moving in a plane are tracked. The internal state in this case comprises the current estimated position and velocity of the platform, and the external measurements are the observed positions of the platform (through dead reckoning, GPS positioning, or sensor feedback). When either the state propagation equation or the external measurement equation is non-linear (cases 2 and 3[1]), these equations are first linearized in the most recent estimate, which results in the extended Kalman Filter (EKF) formulation. While EKF loses many nice mathematical properties enjoyed by the standard Kalman Filter (e.g., provable convergence) [20], it nonetheless is very useful, easy to implement, and efficient at run time.

As the Kalman Filter is really a Bayesian estimator in disguise [15, 29], its basic operation is to properly weigh all available data to arrive at a best state estimate. The data can be at times corroborating or conflicting, and the Kalman Filter depends on the confidence measure associated with the data to break the tie if necessary. Such a confidence measure, in Bayesian terminology, is the error covariance matrix of the data measurements. The Kalman Filter weighs different measurements using the inverse of the covariance matrix (i.e., the larger the error, the smaller the weight).

Situations do arise when such a confidence measure may not be available (case 4). For instance, in the network monitoring example used in our experiments (Section 5.3), the network traffic data are measured exactly. In a mobile robotics application, an ultrasound sensor might return the distance to a nearby obstacle, without indicating how accurate the measure is. Under such a condition, unless an alternative way can be found to assign a meaningful confidence (or error) measure, the data are treated as absolutely correct (or as the ground truth). While the standard Kalman Filter formulation is still applicable, maintaining and updating the covariance matrices makes little sense. (As the measurements are treated as absolute with zero variance, they dominate the evidence fusion process.) State estimation then can be shown to reduce to a least-squares or, more generally, a weighted least-squares fitting problem, where the system state is chosen to best explain the external observation. In that sense, least squares can be considered a special case of Kalman filtering.

Finally for case 5, whether the noise processes are stationary affects the runtime performance. When the noise processes are stationary, e.g., when the position of a mobile platform is reported by a GPS system at regular internals with a fixed precision, the propagation of error covariance becomes completely predictable. This is because updating error covariance involves only the covariances of the state propagation and external measurement processes, not the actual sensor readings themselves. This process can then be performed off-line, resulting in the Riccati equation [20]. In many real-world applications, the error covariance is not stationary, e.g., an ultrasound distance reading is much less accurate if the sound wave is aimed at a wall obliquely, resulting in little energy bounced back to the receiver. In this case, the covariance update has to be carried out in real-time.

To summarize, the reason that we use the Kalman Filter in our research is that the same filtering framework can be customized to address all the situations mentioned above, providing a powerful and versatile filtering paradigm for resource management. In the following sections, we will discuss our formulation with respect to the three important system design parameters: whether the problem formulation is linear (if not, an extended Kalman Filter should be used), whether the external measurements are associated

[1] For example, if the moving platform can also rotate about itself, it is then necessary to incorporate the vehicle's orientation and angular velocity in the state description. The observed position and orientation of the vehicle will then depend on the state vector in a non-linear manner.

with an error value (if not, a least squares or a weighted least squares formulation should be employed), and whether the noise processes are stationary (if not, the covariance matrix is computed and updated in real-time).

4. MODELING KALMAN FILTER FOR DATA STREAMING APPLICATIONS

We now describe how to map a data stream problem to our proposed DKF model. Due to the space limitations we present experimental results to illustrate only a few advantages mentioned in Section 3. We tested our proposed model on three different data sets to demonstrate the advantages 1–4 mentioned in Section 3.1. Testing was done on both synthetic and real datasets. We also show how to construct the state transition matrices for different scenarios.

- *Example 1: Tracking a Moving Object.*
  This example shows the performance of the Kalman Filter when the streaming attribute value does not have high noise and continues a *trend* in arrival characteristics for longer periods of time. Figure 3 shows the synthetically generated data. The experiment involves tracking the position (in two dimensions) of an object moving on linear line segments of different slopes. We demonstrate the first and the fourth advantages mentioned in Section 3.1 through this example. We also show that linear KF model performs better than the precision caching model [23]. If we choose a less accurate model (the constant KF model) the performance is equivalent to that of [23].

- *Example 2: Monitoring Average Zonal Electric Load.*
  The dataset holds the average power load in a particular zone over a period of one month [22]. This example exhibits the performance of the Kalman Filter for complex models, demonstrating the second advantage in Section 3.1. The dataset attribute value used in this example shows a sinusoidal trend over time (see Figure 6). We present experimental results for linear as well as sinusoidal KF models in this example. We show gains in performance if we use a correct model, and also that the performance does not degrade much if we use a less accurate model (the constant KF model).

- *Example 3: Network Monitoring.*
  This experimental setting is useful for monitoring the HTTP traffic between an organization and the outside world [31]. This experiment is used to exemplify the third and the fourth advantages of using DKF. The experimental dataset shown in Figure 9 does not depict any visually-identifiable trend. Here we illustrate the use of the data smoothing feature of the Kalman Filter. We also show how the user can control the sensitivity in the final query answer using the smoothing factor F.

We now describe how the KF equations can be framed for each of the examples shown above.

4.1 Tracking a Moving Object

In most of the moving-object tracking systems, the future location of the object is modeled as a linear function of time. Thus, the trajectory of the moving object is a line segment in the space-time domain. We restrict the movement of the object in the two-dimensional space: thus the location of the object can be represented by a point $P(x_k, y_k)$ at time instant k. Since the object is allowed to move only along straight lines, we have four state variables:

- x_k the Xcoordinate
- y_k the Ycoordinate
- $\dot{x}_k$ rate of change of the Xcoordinate
- $\dot{y}_k$ rate of change of the Ycoordinate

The state transition equation then becomes:

$$x_k = \dot{x}_{k-1}\delta t + x_{k-1}$$
$$y_k = \dot{y}_{k-1}\delta t + y_{k-1}$$
$$\dot{x}_k = \dot{x}_{k-1}$$
$$\dot{y}_k = \dot{y}_{k-1}$$

where δt is the time interval between k and $k-1$.

The measurement from the sensors in most cases (e.g., using a GPS) is the location $P = (z_x,\ z_y)$, and thus the measurement matrix z is

$$[z_x\ z_y]^T$$

The same Kalman Filter formulation can be easily generalized or simplified to suit many different system models and external observations, giving the system tremendous flexibility in modeling different behaviors. For example, if we expect the trajectories to be jerky, then more state parameters, which keep track of increasingly higher-order behaviors, can be used. For example, we can have a state vector of the form $[P, \dot{P}, \ddot{P}, \dddot{P}]^T$ and a transition equation of the form $P_k = P_{k-1} + \dot{P}_{k-1}\delta t + \frac{1}{2}\ddot{P}_{k-1}\delta t^2 + \frac{1}{6}\dddot{P}_{k-1}\delta t^3$. If a shaft encoder is used, it is possible to obtain the velocity information to enrich the measurement as $[z_x, z_y, \dot{z}_x, \dot{z}_y]^T$.

We can also model this process with just *two* state variables, where we do not maintain the rate of change of the X and Y coordinates. Although this would be a less adequate model to map the system, we show in our results that it performs at least as well as the conventional caching method [25]. Using a constant state transition model we have

$$x_k = x_{k-1}$$
$$y_k = y_{k-1}.$$

We can now present the matrices involved in the construction of the Kalman Filter for this problem. The state and covariance matrices are of the same dimension, but the dimensionality depends on the dynamic components in the system. For simplicity we keep the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices as diagonal matrices with value 0.05 and of dimensions 4×4 and 2×2, respectively. The state vector $\boldsymbol{x_k}$ is represented as

$$\boldsymbol{x_k} = \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \end{bmatrix}. \tag{13}$$

The state transition matrix for a linear model is $\boldsymbol{\phi_k}$ is

$$\boldsymbol{\phi_k} = \begin{bmatrix} 1.0 & \delta t & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & \delta t \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{14}$$

Similarly the state transition matrix for a constant model is

$$\boldsymbol{\phi_k} = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}. \quad (15)$$

The state measurement matrix $\boldsymbol{H_k}$ is represented as follows:

$$\boldsymbol{H_k} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix}. \quad (16)$$

We assume that $\boldsymbol{\phi_k}$ remains constant here. Although a more realistic value of $\boldsymbol{\phi_k}$ would make the model more accurate, we use a constant value for simplification, and show the generality of the model we propose.

We assume $\boldsymbol{H_k}$ to be constant to maintain simplicity. As evident from the equations the object is expected to follow its old trajectory in the absence of any forcing function. Thus when there are no measurements, the filter predictions would be based on the latest trend (*slope*) that the object has been following.

4.2 Monitoring Average Zonal Electric Load

In this example we measure the electric power load of a particular zone, updated every hour [22]. A close look at the dataset (Figure 6) indicates that the measurements follow a sinusoidal trend, where the load reaches its peak value during the working hours and drops during the night and early morning hours. At time instant k, if the sinusoidal component is $\alpha sin(\omega t + \theta)$, then this process can be best modeled by using two state variables x_k and s_k as follows:

$$x_k = x_{k-1} + \gamma cos(\omega k + \theta) s_{k-1} s_k = s_{k-1}$$

where,

$$\begin{aligned} x_k &= value\ of\ the\ average\ power\ load \\ &= \alpha sin(\omega k + \theta) \\ s_k &= rate\ of\ change\ of\ sinusoidal\ component \\ &= \frac{d}{dt} x_k \ = \ \gamma cos(\omega k + \theta) \\ &\omega,\ \gamma,\ \theta\ and\ \alpha\ are\ system\ parameters \end{aligned}$$

Matrices for this model can be constructed in a fashion similar to that in Section 4.1

$$\boldsymbol{\phi_k} = \begin{bmatrix} 1.0 & \gamma cos(\omega k + \theta) \\ 0.0 & 1.0 \end{bmatrix} \quad (17)$$

and

$$\boldsymbol{H_k} = \begin{bmatrix} 1.0 & 0.0 \end{bmatrix}. \quad (18)$$

The state transition matrix for a linear model can be constructed similarly.

4.3 Network Monitoring

In this example we measure the HTTP traffic between an organization and the rest of the world [31]. We show a different application of DKF through this example. As shown in Figure 9 the data is extremely noisy revealing no visually-identifiable *trend*. In this situation it is difficult for any prediction algorithm to aid in conserving network communication overhead. We propose to *smooth* the data *online* before operating on it. This is done by using one extra Kalman Filter KF_c^i at the remote source. This filter smooths the data based on parameter F_i and feeds data to the other Kalman Filter (KF_m^i) at the remote source. KF_m^i considers the output from the smoothing filter as the measurement and operates normally as described in the above sections. The performance of DKF is tested on both linear and constant model described in the previous sections. Note that the state transition matrices remain the same for the constant model KF and the smoothing KF. This matrix contains just one element whose value is unity. The extent of smoothing is controlled by parameter F_i, which is the value of the element of a process noise covariance matrix.

5. EXPERIMENTAL RESULTS

The performance of all our proposed models was tested against that of a cached approximation scheme used in [23]. Under this scheme each remote source s_i has a precision width δ_i. A precision bound consists of a lower bound L and an upper bound H such that $H - L = W_i \leq \delta_i$. Each time a sensor measurement V falls outside the bound it is updated to the main server, and the new bounds (H_{new}, L_{new}) are adjusted such that $H_{new} = V + W_i/2$ and $L_{new} = V - W_i/2$. The latest precision bounds are also cached at the main server. We do not consider dynamic bound *growing* and *shrinking* in our results as in [23].

We present a comparative analysis of our approach based on two metrics, *percentage of updates* and *average error value*. *Percentage of updates* is the ratio of updates that are actually sent to the main server to the number of readings taken by the remote source. *Average error value* is the average error *within* the precision constraint encountered during the query. At each time step t_k, if v_k^{source} is the reading at the remote source and v_k^{server} is its cached or predicted value, the error ϵ_k is $\mid v_k^{source} - v_k^{server} \mid$. If the total number of readings made by the remote sensor is n, then *average error value* $= \sum_k \frac{\epsilon_k}{n}$.

We now present experimental results for examples introduced in Section 4 . All the experiments were conducted on a Pentium III processor with 256 MB of memory on 10/100 Mbps Local Area Network (LAN). The coding was done on JDK 1.2.4 using JAMA matrix package [9] for performing matrix operations. The data set used in Example 1 is synthetic whereas real-datasets were used in Examples 2 and 3.

5.1 Example 1: Moving Object Environment

We simulated a moving-object trajectory for the purpose of this experiment. Each moving object had two attributes namely *location* (in terms of X and Y coordinates) and *velocity* (in terms *speed* and *angle* of direction). We used Java's uniform random-number generator to generate different slopes of the velocity vector at random intervals of time. We generated different speeds of the object at random time intervals in a similar manner. Thus the object could randomly change its speed and heading, and then continues on that linear path for a randomly generated length of time. The maximum speed of the object was limited to 500 units, whereas the slope could arbitrarily change by any amount. We constructed a dataset shown in Figure 3, using the above model containing 4000 data points at a sampling rate of 100 ms.

We tested the performance of the Kalman Filter approach on two different state models :

- Constant KF model
  The system is modeled such that the latest updated

value is the best prediction for the future. This model is conceptually similar to the cached approximation value model [24]. The measurement consists of just the position of the object in the two dimensional space, i.e., X coordinate and Y coordinate.

- Linear KF model
  Under this model we take the rate of change of the position into consideration when predicting future values. The state equations were described in Section 4.1.

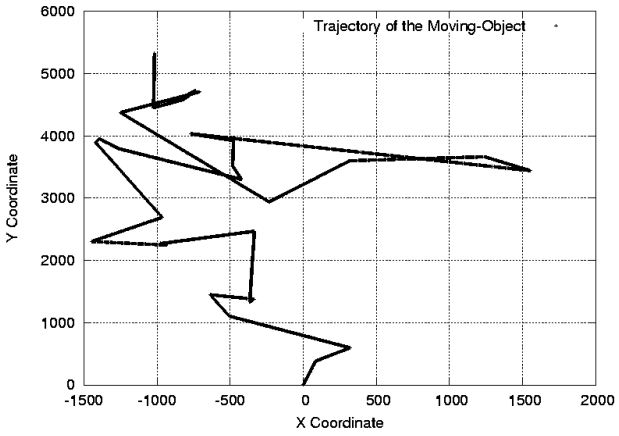

Figure 3: Moving-object dataset (Example 1)

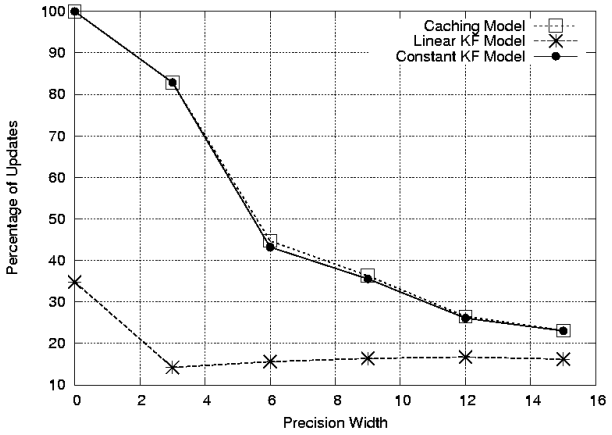

Figure 4: Number of updates received at the central server (Example 1)

Figure 4 shows comparative results of the two Kalman Filter models against the cached approximation scheme. Measurements are taken in the form of position $P(x, y)$. Given a precision constraint δ, point $P(x, y)$ is updated to the server if error in either X or Y value is greater than δ. In both the KF models, only position is recorded, there is no measurement of the rate of change of coordinate values. As evident from Figure 4, the percentage of updates using caching and constant KF model is the same. This is because the constant model is similar to the caching scheme, where the rate of change of values is not taken into consideration. However, if we use the linear KF model, we see that utilization of the communication source was cut down by approximately 75% at a moderate precision width of 3 units. As the precision width increases, the communication resource utilization drops, and all three models show comparable performance. We also observe that the DKF performs at least as well as the caching scheme even in a worst-case scenario (constant KF model). However, if some information about the stream arrival characteristics is available (linear KF model), it can provide a boost to the performance of the system.

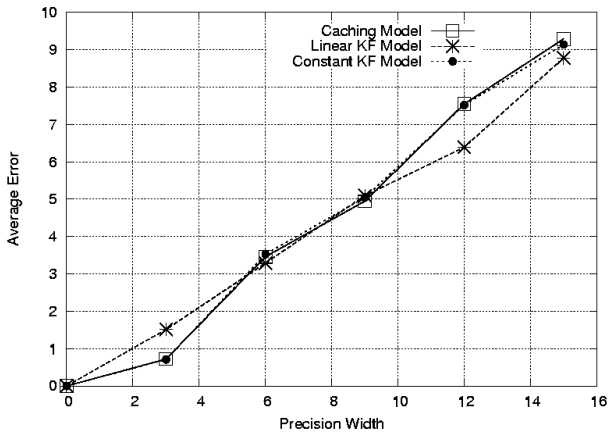

Figure 5: Average error produced by different KF models (Example 1)

Figure 5 shows the comparative results for average error values. The curves exhibited by the constant DKF and the caching scheme are similar; however, the performance of the linear DKF is slightly worse for low precision values but better for higher precision ones. The errors are measured as sum of errors in both the X and Y coordinates. Thus if Δx_k is the error in the X coordinate and Δy_k is the error in Y coordinate at time instant k, the total error is $\mid \Delta x_k \mid + \mid \Delta y_k \mid$. Thus we observe that even after saving significant communication overhead, DKF does not degrade the performance of the system adversely.

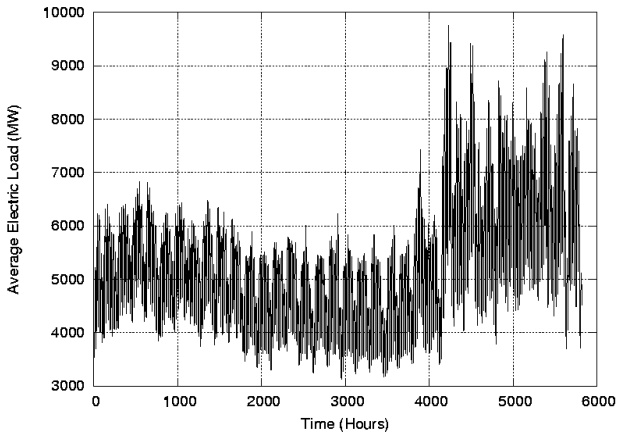

Figure 6: Electric power load dataset (Example 2)

5.2 Example 2: Monitoring Power Load

This dataset consists of electric hourly power load from a zone for a month and contains 5831 data points. The dataset is shown in Figure 6. A closer look at the data shows a sinusoidal trend. Since such stream characteristics can only be deduced after the stream has been analyzed by the system, an accurate model may not always be available.

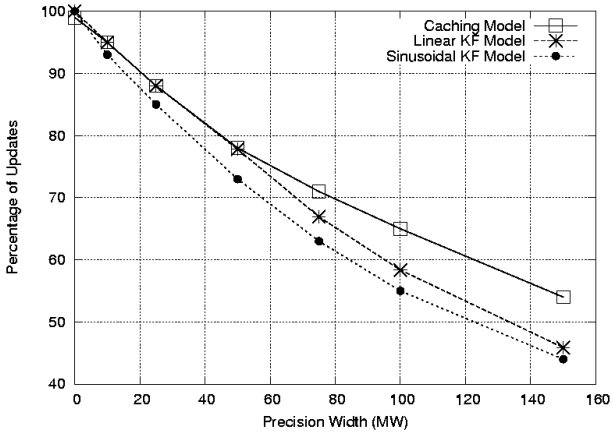

Figure 7: Number of updates received at the central server (Example 2)

Hence, we test the dataset on two KF models, one that assumes sinusoidal components and the other that works on a simple linear KF model. Using notations from Section 4.2, we used the following parameters in the sinusoidal model: $\omega = 18/\pi$, $\theta = \pi$ and $k = 3500$. We tested the sinusoidal model on different parameters, but no significant degradation in performance was observed, and in almost all cases the sinusoidal KF model outperformed the caching model, thus exhibiting the robustness of DKF. Figure 7 shows comparative results for the two KF models used for DKF. For such a complex data arrival characteristic, using a correct KF model gives performance boost of almost 10

Figure 8 shows the average error values. It is seen that performance is relatively comparable for low-precision values. However, for higher precisions caching the model gives slightly better results. At higher precision widths the average errors increase for DKF models, but performance on communication resource conservation improves.

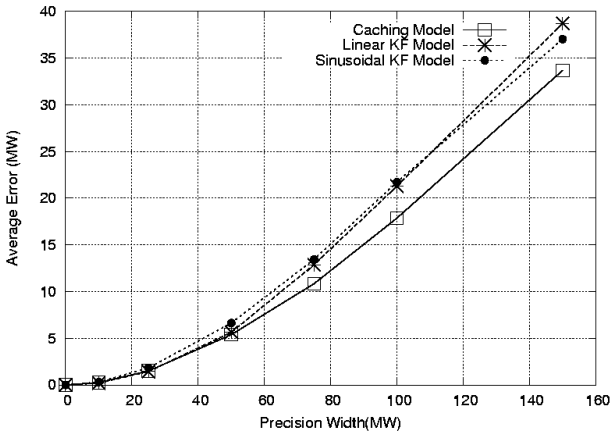

Figure 8: Average error produced by different KF models (Example 2)

5.3 Example 3: Network Monitoring

Here we show another application of KF in processing noisy data streams. This dataset was obtained from [31], and was later processed to hold the number of HTTP packets between Digital Equipment Corporation (DEC) and the rest of the world sampled at an interval of 10 time-stamp units. As evident from Figure 9 the data shows little visible trend and appears like a collection of noisy measurements. In such situations it is often acceptable to provide query results based on moving average values. However moving averages do not follow the trend in the original data closely. Even a series of spikes after a few steady measurements will not alter the moving average value significantly. Thus using this method does not provide *fine-grain* control over the query precision. KF provides this fine-grain control based on a parameter F passed to the filter along with other initialization parameters. The value of F is the covariance assumed in the process model. The advantage of using KF is that it does not require any extra memory, yet provides with a true online solution.

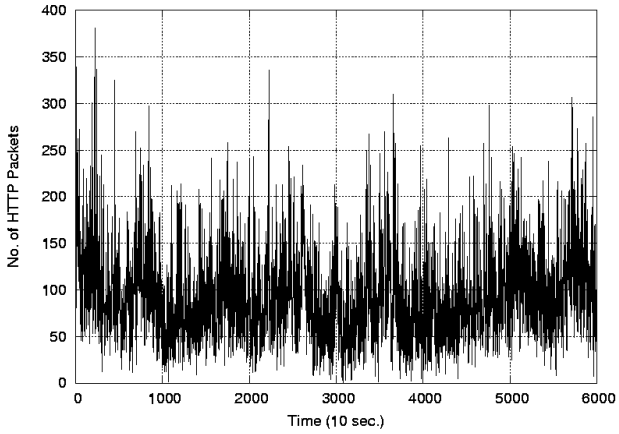

Figure 9: Network monitoring dataset (Example 3)

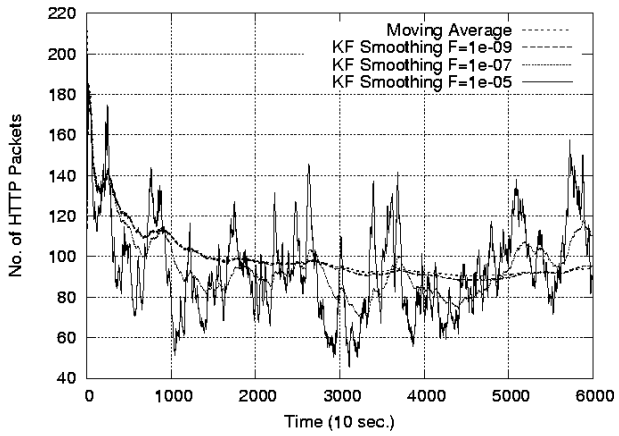

Figure 10: Comparative results for KF smoothing against moving average approach

The adherence of the smoothed data with the real data using KF is shown in Figure 10. It can be seen that using sufficiently low value of F (i.e., $F = 10^{-9}$) the smoothed data values match those produced using a moving average approach.

The performance of DKF for different values of precision for $F = 10^{-7}$ is shown in Figure 11. Since the data have been smoothed, the reduction in communication overhead is better using a linear KF model.

The effect of parameter F on the performance for a given precision is shown in Figure 12. Lowering F improves the performance as the variation in the data value decreases.

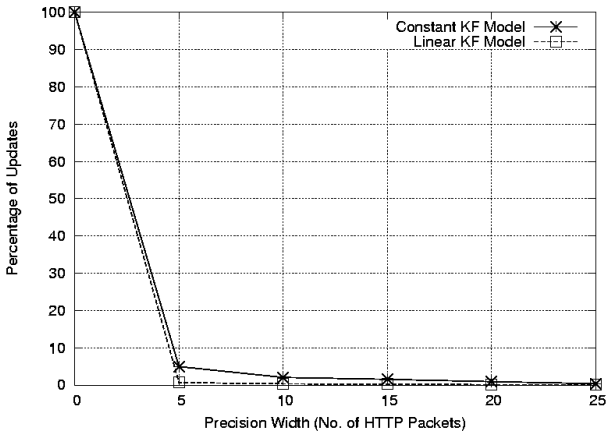

Figure 11: Performance of DKF on smoothed data with $F = 10^{-7}$ (Example 3)

Thus, we observe that using F we have a more flexible model

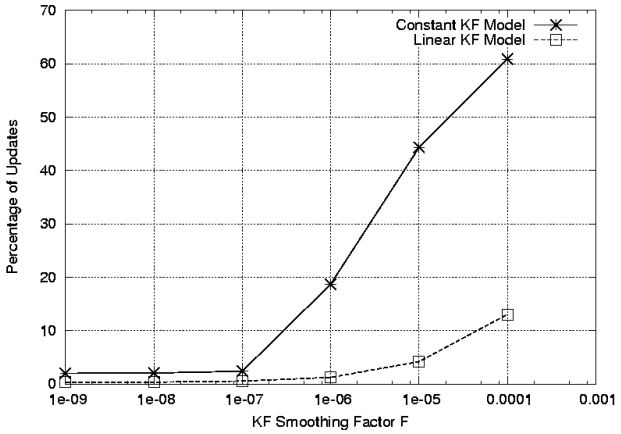

Figure 12: Performance of DKF for precision width $\delta = 10$ (Example 3)

at hand, which can be made to provide results similar to those of moving averages if fine granularity is not desirable.

6. CONCLUSIONS AND FUTURE WORK

We have proposed a novel solution based on a dual Kalman Filter approach to save communication resources in a multiple data streams environment. The framework is presented as a general model, which can be used in a large variety of data-streaming applications. Experimental results reveal that the KF approach gives improved performance in saving communication resources. The system is shown to be robust: it performs well even when the application cannot be modeled accurately. For future work we would like to investigate the following issues and incorporate their solutions into our existing model:

- Developing an end-to-end system, having a general framework for application to streaming systems.
- Investigating updating the state transition matrices online as the streaming data trend changes.
- Developing models for non-linear systems.
- Tuning system parameters for multiple queries with multiple attributes.
- Developing solutions for adaptively adjusting the sampling rate based on the innovation sequence.
- Studying the robustness of the KF when the statistics (for example, the covariance matrix) of the noise are not known, and
- Investigating applications of the Kalman Filter for storing stream (*summaries/synopsis*) under the constraint of specified reconstruction error tolerance.

Acknowledgements

We would like to thank the support of two NSF grants NSF Career IIS-0133802 and NSF ITR IIS-0219885. We would also like to thank Professor Jennifer Widom, Professor Rajeev Motwani, and members of the STREAM group for their suggestions to improve the related work section.

7. REFERENCES

[1] D. Abadiand, D. Carneyand, U. Cetintemel, M. Cherniack, C. Convey, C. Erwin, E. Galvez, M. Hatoun, J. Hwang, A. Maskey, A. Rasin, A. Singer, M. Stonebraker, N. Tatbul, Y. Xing, R.Yan, and S. Zdonik. Aurora: A data stream management system (demonstration). In *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data*, San Diego, CA, USA, June 2003.

[2] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, R. Motwani, I. Nishizawa, U. Srivastava, D. Thomas, R. Varma, and J. Widom. STREAM: The stanford stream data manager. *IEEE Data Engineering Bulletin*, 26:19–26, March 2003.

[3] A. Arasu, B. Babcock, S. Babu, J. McAlister, and J. Widom. Characterizing memory requirements for queries over continuous data streams. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 221–232, Madison, WI, USA, June 2002.

[4] B. Babcock, S. Babu, M. Datar, R. Motwani, and D. Thomas. Operator scheduling in data stream systems. Technical report, Stanford University, CA, USA, October 2003.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–16, Madison, WI, USA, June 2002.

[6] B. Babcock, M. Datar, and R. Motwani. Load shedding for aggregation queries over data streams. In *Proceedings of the ICDE Intl. Conf. on Data Engineering*, Boston, MA, USA, March 2004.

[7] S. Babu, U. Srivastava, and J. Widom. Exploiting k-constraints to reduce memory overhead in continuous queries over data streams. Technical report, Stanford Univesity, CA, USA, November 2003.

[8] A. Bletsas. Evaluation of Kalman filtering for network time keeping. In *Proceedings of PerCom IEEE Intl. Conf. on Pervasive Computing and Communications*, pages 289–296, Fort Worth,TX, USA, March 2003.

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

[10] R. G. Brown. *Introduction to Random Signal Analysis and Kalman Filtering*. Wiley, New York, NY, USA, 1983.

[11] A. Bulut and A. K. Singh. SWAT: Hierarchical stream summarization in large networks. In *Proceedings of the ICDE Intl. Conf. on Data Engineering*, pages 303–314, Bangalore, India, March 2003.

[12] S. Chandrasekaran. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *Proceedings of the CIDR Conf. on Innovative Data Systems Research*, Asilomar, CA, USA, January 2003.

[13] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of ACM SIGMOD Intl. Conf. on Management of Data*, pages 551–562, San Diego, CA, USA, June 2003.

[14] R. Clarke, J. Waddington, and J. N. Wallace. The application of Kalman filtering to the load/pressure control of coal-fired boilers. In *IEE Colloquium on KAlman Filters: Introduction, Applications and Future Developments*, volume 27, pages 2/1–2/6, London,UK, Feburary 1989.

[15] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, 2nd Edition*. Wiley, New York, NY, USA, 2001.

[16] L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Record*, 32(2):5–14, June 2003.

[17] Z. G. Ives, D. Florescu, M. Friedman, A. Levy, and D. S. Weld. An adaptive query execution system for data integration. In *Proc.of ACM SIGMOD Intl. Conf. on Management of Data*, pages 299–310, Philadelphia, PA, USA, June 1999.

[18] 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, March 1960.

[19] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantess. In *Proceedings of the ICDE Intl. Conf. on Data Engineering*, pages 429–420, Bangalore, India, March 5–8 2003.

[20] P. S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, New York, NY, USA, 1979.

[21] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query processing, resource management, and approximation in a data stream management system. In *Proceedings of the CIDR Conf. on Innovative Data Systems Research*, Asilomar, California, USA, January 2003.

[22] Basic generation services data room, http://www.bgs-auction.com/bgs.dataroom.asp. Newark, NJ, 2003.

[23] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 563–574, San Diego, CA, USA, June 2003.

[24] C. Olston, B. Loo, and J. Widom. Adaptive precision setting for cached approximate values. In *Processdings of ACM SIGMOD Intl. Conf. on Management of Data*, pages 355–366, Santa Barbara, CA, USA, May 2001.

[25] C. Olston, B. T. Loo, and J. Widom. Adaptive precision setting for cached approximate values. In *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 355–366, Santa Barbara, CA, USA, May 2001.

[26] C. Pereira, S. Gupta, K. Niyogi, I. Lazaridis, S. Mehrotra, and R. Gupta. Energy efficient communication for reliability and quality aware sensor networks. Technical report, Univeristy of California at Irvine and University of California at San Diego, April 2003.

[27] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.

[28] T. Simunic, H. Vikalo, P. Glynn, and G. D. Micheli. Energy efficient design of portable wireless systems. In *Proceedings of ISLPED IEEE Symposium on Low Power Electronics and Design*, pages 49–54, Rapallo, Italy, July 2000.

[29] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, MA, USA, 1986.

[30] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *Processdings of VLDB Intl. Conf. on Very Large Data Bases*, pages 309–320, Berlin, Germany, September 2003.

[31] The internet traffic archive, http://ita.ee.lbl.gov. Lawrence Berkeley National Laboratory, USA, April 2000.

[32] G. Welch and G. Bishop. An introduction to the Kalman filter. In *ACM SIGGRAPH Intl. Conf. on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA, August 2001.

[33] G. Wu, Y. Wu, L. Jiao, Y.-F. Wang, and E. Y. Chang. Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. In *Proceedings of the ACM Intl. Conf. on Multimedia*, pages 528–538, Berkeley, CA, USA, 2003. ACM Press.

[34] W. Wu, M. J. Black, E. B. Y. Gao, M. Serruya, A. Shaikhouni, and J. P. Donoghue. Neural decoding of cursor motion using a Kalman filter. In *Neural Information Processing Systems: Natural and Synthetic*, pages 133–140, Vancouver, British Columbia, Canada, December 2002.

[35] Y. Yao and J. Gehrke. Query processing for sensor networks. In *Proceedings of the CIDR Conf. on Innovative Data Systems Research*, Asilomar, CA, USA, January 2003.

[36] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *Proceedings of the ICDE Intl. Conf. on Data Engineering*, pages 13–22, San Diego, CA, USA, March 2000.