Probabilistic Data Aggregation In Distributed Networks

Ling Huang
Ben Zhao
Anthony D. Joseph
John D. Kubiatowicz

Electrical Engineering and Computer Sciences
University of California at Berkeley

Probabilistic Data Aggregation
In Distributed Networks

Ling Huang, Ben Y. Zhao, Anthony D. Joseph and John Kubiatowicz
University of Californa at Berkeley
{hling, ravenben, adj, kubitron}@eecs.berkeley.edu

Abstract— We explore techniques to reduce the sensitivity of large-scale data aggregation networks to the loss of data. Our approach leverages multi-level modeling and prediction techniques to account for missing data points and is enabled by the temporal correlation that is present in typical data aggregation applications. The result can tolerate significant *involuntary* data loss while minimizing overall impact on accuracy. Further, this technique permits nodes to probabilistically remove themselves from the network in order to reduce overall resource usage such as bandwidth or power consumption. In simulation, we explore the tradeoff between algorithmic complexity and prediction performance across a variety of data sets with different dynamic properties. We quantify the temporal correlation in several real-world datasets, and achieve more than 50% resource savings in an environment with significant loss, while maintaining high accuracy.

Index Terms— Statistics, Simulations, Network Measurements.

I. INTRODUCTION

The growing deployment of large-scale distributed networks, such as sensor and overlay networks, presents an interesting opportunity for distributed data measurement and collection applications. Nodes in these networks perform local data collection operations, and cooperate to disseminate the data to other decision making nodes. For example, overlay nodes deployed across a wide-area network, can each monitor local network traffic and collaborate to detect network intrusions or attacks [1]. Similarly, each overlay node could monitor the performance quality of access to a set of web sites. In sensor networks, individual nodes collect and propagate physical information, such as temperature, humidity, light, etc [2].

For communication efficiency, a large network of data gathering nodes is typically organized into a hierarchical, latency-optimized tree structure that aggregates collected data and delivers the result to a small set of root servers, as shown in Figure 1. At periodic intervals, each node performs a local measurement, optionally aggregates it with data from other nodes by applying data aggregation
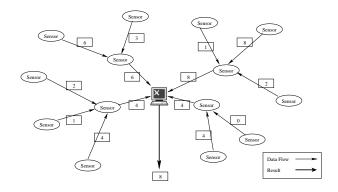

Fig. 1. Calculating MAX using tree-based aggregation.

functions such as *COUNT*, *MAX*, *SUM* and *AVERAGE*, and then sends the result towards the root server by forwarding to its parent in the hierarchy.

Large-scale data aggregation alone does not solve the challenge of dealing with the associated high communication costs. Timely dissemination of collected data often directly conflicts with overall network stability. In sensor networks, the frequent communication required for timely result delivery taxes the computational and power resources of limited, battery-powered sensors. In wired peer-to-peer overlays, frequent communication consumes bandwidth and may cause congestion and data loss, which can be particularly costly when measuring performance or detecting failures and network attacks.

In this paper, we apply statistical algorithms to hierarchical in-network data aggregation systems in distributed networks. We propose to exploit existing time-series correlation in real-world measurements for resource savings by allowing nodes in an aggregation network to voluntarily opt-out of certain rounds of aggregation. We describe a number of prediction techniques that allow the network to minimize the resulting loss of accuracy by approximating or *predicting* values. We show that by accepting a small loss in accuracy, nodes can significantly reduce power and bandwidth consumption. More specifically, we provide the following contributions:

- We observe that real-world measurement data, such as those taken by sensors or network measurement

nodes, generally exhibit a high correlation across time epochs. We validate this by applying statistical methods to real sensor data.

- We propose the use of *multi-level prediction* to accurately recover lost data and evaluate its effectiveness. To the best of our knowledge, this is the first practical application of multiple-level prediction algorithms for data recovery in real systems.
- We propose the idea of selective participation of nodes to reduce the overall consumption of resources such as bandwidth and power.
- We explore the tradeoff between algorithmic complexity and prediction performance across a variety of datasets with different dynamic properties.
- We run extensive simulations on two large real-world datasets and a synthetic dataset, and show that we can achieve more than 60% resource savings while maintaining 90% accuracy in the lossless environment and 80% accuracy in an environment with a 20% average loss rate.

The rest of the paper is structured as follows: We first discuss previous and related work in Section II, and define our aggregation model in Section III. Next, we compare time-series correlation patterns of real-world and synthetic data traces in Section IV and propose the use of several prediction algorithms to recover from data loss in Section V. Given these mechanisms, we propose a selective participation mechanism for reducing resource utilization in Section VI. Finally, we provide simulation and analytical results in Section VII, followed by a discussion of future work and our conclusions in Section VIII.

II. RELATED WORK

A number of previous and ongoing projects in the networking and database research areas are relevant to our work. Astrolabe [3] performs aggregation per partitioned zone, and propagates data across zones using a gossip protocol. In resource-constrained sensor networks, several systems use the in-network processing approach to compute aggregates based on disjoint subsets for decomposable functions [4]–[8]. Among them, TAG [8] proposes an aggregation service for ad-hoc networks using an SQL-style query interface. However, the current TAG approach is sensitive to data loss, especially for large size networks. Based on TAG framework Considine et al. investigate the use of approximate in-network aggregation using small sketches [9].

In a lossy environment, data aggregation systems can approximate or predict lost values based on a history of past values, or nearby (in terms of logical or physical location) current values. A natural tradeoff exists between prediction accuracy and computational and storage cost. While most data aggregation systems to date use the simplest mechanism based on last-value prediction, more complex and effective statistical algorithms have been applied in other fields of network research. For audio/video and network data traffic, linear prediction method has been considered as an efficient and effective alternative [10], [11].

Qiao et al. empirically study the multi-scale predictability of network traffic [12]. They use wavelet approximations as the key tool for multi-scale data representation and prediction. However, they only consider "one step-ahead" predictability, instead of the multi-step-ahead prediction required to deal with bursty and consecutive loss in real networks. Their results are also limited to single data traces. Finally, using wavelets for decomposition is computationally expensive, and would be difficult to integrate into an online algorithm for continuous aggregation.

Few data aggregation systems use advanced statistical algorithms to predict lost values [7], [8], [13]. None of the above algorithms have been used to predict data based on temporal correlation of measured data.

The database community has looked at approximate queries based on partial information using sampling algorithms, generally using centralized aggregation. Olken discusses random sampling on B^+ trees, hash files, and relational operations in [14]. Hellerstein et al. propose using online aggregation to evaluate queries progressively with random sampling [15]. Chaudhuri et al. propose a combination of outlier-indexing with a weighted sampling mechanism for approximate queries [16]. Finally, Olston et al. propose adaptive filters for continuous queries based on centralized techniques [17]. However, it is unclear whether these mechanisms can be applied to the online data aggregation in a decentralized, high loss environment.

III. AGGREGATION MODEL

In this section, we define our operating context and system model for data aggregation. Our work targets large-scale networks performing periodic measurements yielding an ordered stream of aggregated outputs. Because a tree-based hierarchy can scale to large networks while operating with well-defined latency bounds, we assume here a tree-based aggregation model (see Figure 1), where end hosts (*nodes*) perform measurements, aggregate and forward results to an output server (*root server*). While we recognize that how aggregation trees are constructed impacts its performance (*e.g.* minimal latency spanning trees in ad-hoc routing [18] or structured

Technique	Multi-tree	Retransmission	Prediction
CPU Usage	moderate	low	low to high
Comm. Overhead	very high	high	none
Latency	low	high	low

TABLE I

Comparison of fault-tolerance techniques.

peer-to-peer routing [19]), construction algorithms are orthogonal to our work, and our approach would work on top of any tree-based structure.

We focus on using selective participation and prediction techniques (see Section VI and V) on tree-based in-network processing to enable a large-scale aggregation network to conserve resources (*e.g.*, bandwidth and power) with minimal loss in aggregation accuracy. We address the issue of continuous queries, where aggregate values are continually computed and routed up towards the root server. We also assume all nodes have relatively synchronized clocks (*i.e.* their offsets are kept constant). We partition time into epochs, and produce a single aggregated result per epoch for each running query. At each epoch, each node in the tree sends its aggregated value to its parent node. Once a node has heard from all of its children, it aggregates the data with its own value and forwards it to its own parent.

However, if all nodes participate in every epoch, a sensor will quickly drain its batteries and lose power, while nodes in an overlay can consume a lot of communication bandwidth and collectively cause network congestion. We propose an alternative to the traditional model where all nodes participate in data aggregation in every epoch. Instead, nodes can participate probabilistically according to a well defined rate. Nodes in the network can voluntarily opt-out of certain rounds of aggregation to reduce power consumption and communication overhead. Meanwhile, the network can exploit existing time-series correlation in data streams to predict lost values and maintain accuracy in the aggregated result.

Our prediction algorithms are geared towards "recovering" from data loss due to loss at the network layer and loss from nodes not participating in the aggregation. Our prediction algorithms currently require parent nodes to build models and predict lost data for its children. A useful optimization is for the child to periodically generate statistics based on history of its own values, and send the compressed model up to its parent node. The parent can then recalibrate its model to improve its accuracy. Over time, this synchronization mechanism bounds the amount of deviation present in the parent node's model.

We do not explicitly address issues related to nodes failing/crashing. They can be addressed by techniques in dynamic topology management to modify and adapt aggregation trees to failures. Existing systems have used other mechanisms to improve resiliency of data aggregation, including aggregating data across multiple independent aggregation trees and using data retransmission for reliability. These mechanisms make use of additional resources to improve reliability. These techniques are orthogonal and complementary to the ones we propose here. Table I shows the tradeoffs between different mechanisms.

IV. TEMPORAL PROPERTIES OF REAL-WORLD DATASETS

Before we present our algorithms, we test our observation that temporal correlation exists in most application sensor measurements. We introduce several statistical metrics and apply standard statistical techniques to several real-world traces to extract their temporal variance.

A. Datasets And Metrics

We have been looking for datasets from different fields and with different degrees of temporal correlation for our analysis and experiments. However, a few environment/network traffic monitoring datasets with large number of individual data streams are publicly available.

We download a large temperature dataset from National Climatic Data Center [20]. It consists of more than 5,000 streams of temperature measured by stations all over the world on daily intervals in year 2000. We also obtain a small dataset (less than 100 streams) of bandwidth measurements for network interfaces on Abilene routers in year 2004 [21].

Since large sets of stock quote data streams are publicly available, we examine their temporal characteristics and consider the validity of using them as a large time-based stream dataset. We downloaded monthly stock quotes for a random set of 5,000 stocks, where each stock's data stream included at least 200 data points across time.

In addition, we generated two artificial datasets, including a random dataset and a random walk dataset. The random dataset is generated with values uniformly distributed over the range $[0, 50]$. The random walk dataset is generated using the statistics of the stock dataset. For each data stream i in the stock dataset, we compute its mean μ_i and the variance σ_i of its relative change between data points. We then generate each data stream i in the random walk dataset such that its mean is μ_i, and its relative change is drawn from the Gaussian distribution $G \sim N(0, \sigma_i^2)$.
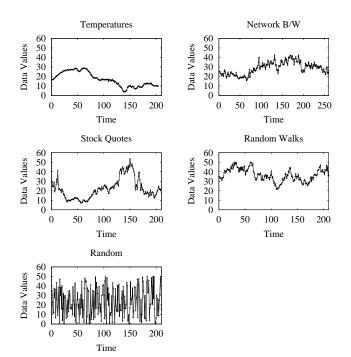
Fig. 2.   Real-world data traces and synthetic traces.

Fig. 3.   Density distributions of Relative Increment.

Dataset	Mean	RSD	MRI
Temperatures	17.10	3.11	0.040
Network B/W	16.88	1.31	0.063
Stock Quotes	22.08	5.09	0.073
Random Walk	36.60	1.17	0.055
Random	24.09	7.99	0.810

TABLE II

Statistical properties of different data streams.

We normalize values in all datasets to the same absolute range for comparison, and plot trace segments in Figure 2. We compare the temporal characteristics of stock, random and random walk datasets to the two measurement datasets using two key statistical metrics: Relative Standard Deviation (RSD) and Mean absolute Relative Increment (MRI). Given a series of data $x_i, i = 1, ...n$, let $\bar{x}$ be the mean and $std(x)$ be the standard deviation. The RSD is defined as:

$$RSD = \frac{std(x)}{\bar{x}} \tag{1}$$

Intuitively, RSD measures how data values in the dataset spread out based on their mean value. It represents the static distribution of data values in the space.

Let $\Delta x_i^d = x_{i+d} - x_i$, the increment in d intervals starting at i. The Relative Increment in d intervals is defined as:

$$RI_i^d = \frac{\Delta x_i^d}{\bar{x}} \tag{2}$$

The $d - interval$ MRI is defined as:

$$MRI^d = \frac{1}{(n-d)} \sum_{i=1}^{n-d} \left| \frac{\Delta x_i^d}{\bar{x}} \right| \tag{3}$$

Intuitively, MRI measures how big data values change in given time intervals. It represents how dynamic a data stream is along time epochs.

B. Correlation

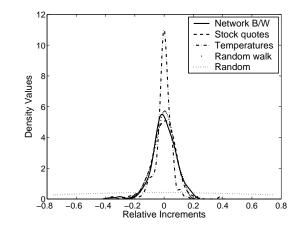We compare the statistical properties of all five datasets in Table II. While the stock data and the random walk data share similar statistical properties with those of the temperature and network datasets, the random data differs strongly from measurement datasets, with a MRI that is an order of magnitude greater than the other four. Figure 3 plots the density distributions of Relative Increment in single interval for all traces. Again, the two measured traces share similar distributions with the stock data and the random walk data, while the random data distribution is significantly different. Thus, we conclude that the temperature, stock and random walk datasets are all reasonable datasets to use in our measurements. The temperature dataset is not very dynamic, and shows high temporal correlation. In contrast, the stock quotes dataset is highly dynamic, and shows less temporal correlation than the other measurement datasets. These two datasets cover the two extreme ends of dynamics in real datasets, and their results should be indicative of those from real world sensor data.

V. RECOVERING FROM DATA LOSS

In this section we explore ways in which we can exploit the temporal correlation of Section IV to recover lost data items. When a node does not report its aggregate value during a cycle, either due to intentionally not sending it or to data loss at the network link layer, its parent node can attempt to mitigate the negative impact on accuracy by using statistical techniques. Our focus is on improving accuracy with minimal commitment of additional resources. We discuss several existing statis-

tical estimation approaches for recovering missing data values, and then propose the use of a novel *multi-level estimation* algorithm.

A. Existing Estimation Algorithms

For datasets that exhibit temporal correlation, a simple estimation technique is to reuse the last received value from the child node. This technique requires no computation and very little state, but has limited effectiveness (see Section VII).

An alternative is to use more advanced statistical techniques to improve accuracy. One choice is *linear prediction* of order m, where the estimated value is expressed as a linear function of the previous m values [22], [23]. This function can be visualized in a m dimensional space spanned by $x(n-1), x(n-2), ..., x(n-M)$, where:

$$\hat{x}(n|n-1, n-2, ..., n-m) = \sum_{k=1}^{k=m} a_k x(n-k) \quad (4)$$

where a_k are constant coefficients.

The Mean Square Error (MSE) metric is often used to quantify the difference between estimated and actual values. Improving accuracy by minimizing the MSE is defined as: $Minimize \ \ E[e^2(n)]$ where $e(n) = |x(n) - \hat{x}(n)|$, reducing the problem to linear prediction with Minimum Mean Square Estimation (MMSE). Several practical algorithms apply to this problem [23] and we evaluate them in Section VII.

We have observed that real-life data streams contain both long-term trends and transient oscillations. Therefore it is very difficult for a single linear prediction algorithm to model trends in both short and long time-scales. To make things worse, data loss is bursty and consecutive drops occur often in real network settings. During a bursty loss phase, linear prediction algorithms generally do not have enough history information to make accurate estimates, resulting in high relative errors.

This problem, along with results we show in Section VII, show that the accuracy of linear prediction is limited for real datasets. We have even observed that the last-value estimation method often outperforms a more complex linear prediction algorithm.

B. Multi-level Estimation

To address the need to recognize and utilize both long term trend and short term oscillations, we now propose a novel *multi-level estimation* algorithm.

Data streams from real world we examined in Section IV can be decomposed into two or more levels. We see from Figure 2 that while temporal correlation exists
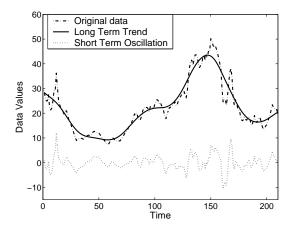

Fig. 4. Two-level representation of a stock data trace.

in the form of a long-term trend, high frequency, low-amplitude oscillations can be found in the short time-scale.

The general intuition behind multi-level estimation is to decompose a sequence of values (in our case the sequence of values reported by a child node) into multiple independent functions with different time-scales of statistical behavior. For our purposes, we identify two time-scales corresponding to long term temporal correlation and short term oscillations. We now discuss how a two-level estimation algorithm can be implemented.

1) Two-level Representation: We propose to decompose a data stream into two levels, as shown in Figure 4, using B-spline curves to model the long-term trend and the AutoRegressive Moving Average (ARMA) model to model short-term oscillations. Several techniques can be employed to decompose data stream into two levels, including Low Pass Filtering and Least-Square Regression. For our purpose, we use the Least-Square B-spline Regression technique to decompose data streams [24].

Although the long-term trend changes slowly, it has a complex shape along the time axis, and is difficult to represent using an analytical curve. Hence a piecewise parametric curve must be used. A B-spline curve is a piecewise polynomial curve that we use to represent the complex curves of time series data. The advantages of using B-spline regression model are:

- We can use a low degree B-spline curve (we use degree $d = 3$) to describe an arbitrary complex shape and avoid the instability of numerical computation of a high degree polynomial.
- There are fast linear algorithms for computing positions and derivatives of points on a B-spline curve.
- It provides a compact representation of the original data. As shown in Figure 4, we create a B-spline curve one-eighth the original data's size that captures the original data's dominant change with a

	Last-Value	Linear	ARMA	Two-Level
CPU	0	$O(m^3T)$	$O(k^3T)$	$O(k^3T) + O(d^3T)$
Storage	$O(1)$	T	T	$c \cdot T,\ 1.0 < c < 2.0$

TABLE III

CPU and storage complexity of prediction algorithms.

mean relative error between them of only 0.021.

We observe that short term oscillations in the data have a zero mean and a form of stationarity property. The ARMA model provides one of the basic time series modeling tools for capturing the statistical dependence of these oscillations [25]. An ARMA(p, q) model consists of two components, an AutoRegressive (AR) component of order p and a Moving Average (MA) component of order q. There are standard routines to determine the value of p and q. We omit the details due to space constraints. For our purposes we use $p = q = 2$.

To provide immediate estimations of lost values, we need online algorithms to incrementally decompose a data stream. With each new value, we update the long-term trend spline curve, and calculate the short-term oscillation as the difference between the result and the real value. We use a piecewise continuous B-spline with degree $d = 3$ because it has a low overhead online regression algorithm. Incoming data will only affect the shape of a spline in a local area, thus we only need to update the several adjacent pieces of segments.

2) Two-Level Prediction: Each node in the aggregation tree maintains a window of two-layer history data (we currently use window size of 30 measurement periods). Upon a data loss, the two level prediction algorithm is invoked to estimate the lost data.

We obtain a smooth curve of long-term trend up to the current epoch by fitting a spline to the history data. There are several ways to extend this spline to predict future values. One is a linear extension of the spline in the direction of the first derivative. A better alternative is a curvature continuous spline extension [26].

We apply ARMA forecasting to the residual data (transient oscillations), using an algorithm in [25], to build an ARMA model, estimate its parameters, and derive a recursive equation that can be used to estimate future data values with low complexity [25].

With our approach, every data stream is decomposed into multiple layers only once at the node which produces this data stream. In our data aggregation system, data streams are stored, transferred and aggregated in this multi-layer format. Whenever there is a data loss, we use multi-level prediction to estimate the lost data. Note that the storage and transmission bandwidth requirements are both low, since we can maintain a compact multi-level representation by borrowing ideas from Wavelet multi-
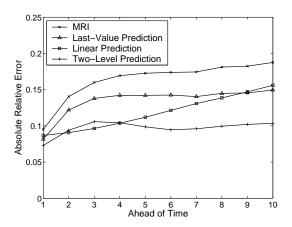

Fig. 5. Performance of prediction algorithms in multi-step prediction error on a data trace.

resolution decomposition and representation [27].

3) Complexity of Prediction Algorithms: We analyze the computation and storage complexity of several prediction algorithms and summarize the results in Table III. In the table, T is the window size for history data and $T = 30$ in our setting. In linear prediction, m is the order of the predictor and $m = 4$ in our setting. In both ARMA and two-level prediction, $k = max(p, q + 1)$, where (p, q) are the order of ARMA model and $p = q = 2$ in our setting. In two-level prediction, d is the degree of B-spline curve and $d = 3$ in our setting. Last-value prediction has the lowest complexity with no computation and a little storage. All other algorithms have storage complexity that increases with T. Two-level prediction has the highest computation complexity, comprised of the complexity of ARMA model and the complexity of B-spline model. In general, algorithms of linear prediction, ARMA model and B-spline regression all require computation to optimize a set of curve-fitting parameters. Their complexity goes up with the cube of the number of parameters, which corresponds to the order of the model. For B-spline regression with degree $d = 3$, there is a fast algorithm for parameters estimation, and its complexity is $O(3^2 \cdot T) = O(9 \cdot T)$ [24].

4) Performance of Prediction Algorithms: Here we use a simulation to compare the accuracy of last-value prediction, linear prediction and two-level prediction algorithms, where accuracy is measured by the absolute relative error of predicted values with multiple steps ahead of time. We implement a predictor with the three prediction algorithms. In the experiment, the predictor makes real-time predictions multiple time steps into the future using only a history cache of the previous 30 data values reported. We measure the absolute relative difference between the predicted future data and real incoming future data, and plot them on the Y-axis. We

perform continuous prediction for 200 epochs and plot mean values for each data point. Figure 5 shows the result on one data trace.

The MRI curve is the mean absolute relative increment of the data trace with intervals equal to the multi-step-ahead of time. As described in Section IV, it captures the dynamic property of the data trace. We can see that all three prediction algorithms are effective in predicting values with small relative error, even when predicting multiple steps ahead. Two-level prediction performs the best and last-value prediction performs the worst. Their prediction error curves are greatly shaped by MRI, the dynamic property of the data trace. As expected, linear prediction performs well a small number (< 6) of steps into the future, but suffers when required to predict with a larger number (> 7) of missing values.

VI. Selective Participation

Since the techniques of the previous section permit our data aggregation network to tolerate data loss, we now introduce the notion of *selective participation* of nodes as a technique for reducing overall resource utilization. We present two approaches to distributing participation load across the network, and discuss their impact on overall aggregation accuracy. We also briefly discuss how participation rates are realized, and their relationship to lossy networks.

To perform selective participation, each node makes a local decision about whether it should participate (listen to and receive data from children, perform aggregation with local measurement and send results to parent node in tree), or not participate (go to sleep and ignore all data for this aggregation cycle). In each cycle, if a parent hears nothing from a child, it can proceed by estimating the missing value based on a history of its previous values, as discussed in Section V.

While our scheme allows the network topology itself to change over time, we do not address the issues surrounding topology adaptation in this paper. Existing works ([28], [29]) address issues of how to adaptively maintain dissemination trees across changes in network membership.

Having defined the context of our work, we now discuss two approaches to distributing aggregation load (or participation rates) across the network:

a) Uniform Participation: Nodes participate in aggregation at an equal rate across the network. The drawback of this simple approach is that nodes near the aggregation root will discard a large portion of the aggregated values when they opt out of a cycle.
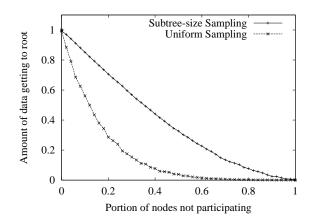

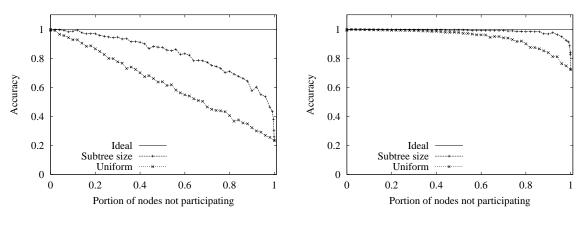Fig. 6. The amount of data getting to root versus node non-participation rate

b) Subtree-Size Based Participation: This algorithm sets participation rates across the network so that the probability a node participates in aggregation is proportional to the number of nodes in its aggregation subtree. Since nodes higher in the tree handle values representing the aggregate result of more data points, this scheme increases their likelihood of participating. Nodes low in the tree handle fewer values, and have a lower probability of participating.

To calculate subtree sizes, nodes aggregate and report their subtree sizes along with measurement values. For fair distribution of resource consumption and avoidance of hotspots, the network could distribute different aggregation operations across interior-disjoint trees, similar to bandwidth distribution in SplitStream [30].

Figure 6 shows the actual number of data values that get through the aggregation tree using both approaches. As the portion of nodes choosing to opt out increases, subtree-size participation distributes most of sleeping nodes near the bottom of the tree, allowing more data values to aggregate through. In contrast, in uniform participation nodes near the root are equally likely to opt out, removing large numbers of values from consideration.

Once a node is assigned a participation rate, it can choose different ways to implement the rate to create its participation pattern. A node can act without regard to its history, by calculating its current action independently at each aggregation cycle. An alternative approach is to generate a near-ideal short term participation pattern, by using short-term history to guide current choices.

Finally, from the perspective of a parent node in the aggregation tree, a *child node*'s voluntary non-participation is equivalent to the data for that aggregation cycle being lost. In fact, a uniform participation rate distribution is analogous to a network with evenly

(a) MAX *Operation*   (b) AVG *Operation*

Fig. 7. Accuracy for the stock dataset versus participation rate distribution.

distributed link loss rates. Thus, the value estimation techniques we introduce to increase aggregation accuracy are also applicable as data recovery algorithms in a lossy environment.

VII. SIMULATION RESULTS

In this section, we evaluate our selective participation and prediction algorithms using two large real world datasets and a synthetic dataset. They are the stock, temperature and random walk datasets we discussed in Section IV. They are from different applications, exhibit drastically different degrees of temporal correlation, and demonstrate the applicability of our algorithms to a broad range of applications. We mainly use the stock dataset for most of our experiments, and perform select experiments on the random walk and temperature data traces to verify results obtained using the stock dataset. The network bandwidth dataset is not sufficiently large for our experiments.

For simulation, we use a packet-level simulator that performs tree-based data aggregation on different topologies[1]. We present here the results based on a 5,000 node transit-stub topologies [31]. We construct aggregation trees with randomly chosen roots at transit nodes. Each node reads data from a trace (stock and temperature) or generates values according to a distribution (random walk).

The simulator includes the following components: (1) a *Tree builder* that uses Dijkstra's shortest-path-tree algorithm [32] to construct and maintain a tree topology; (2) a *Data aggregator* that performs aggregation using functions (*e.g.*, MIN, MAX, COUNT, SUM, and AVG); (3) a *Data sampler* that performs node selective participation

[1]Note that the simulator does not model network effects such as cross traffic and retransmission.

using the local strategies in Section VI; and (4) a *Data estimator* that performs statistical estimation based on available data using one of the discussed algorithms whenever there is data loss or non-participation.

We use accuracy as our key metric, and define it as $(1 - error/TrueValue)$, where *error* is the difference between *TrueValue* (calculated without loss), and the result calculated with selective participation and loss. We perform continuous queries for at least 200 epochs and plot mean values for each data point. For the simulations, we vary each algorithm's parameters to generate distributions that correspond to a range of participation from full participation by all nodes to non-participation by most nodes.
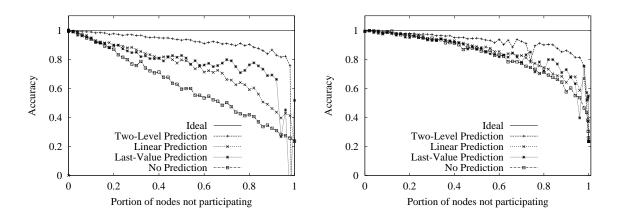
A. Extensive Evaluation Using Stock Trace

We randomly map individual stocks to single nodes in the topology and perform continuous queries on the aggregation tree.
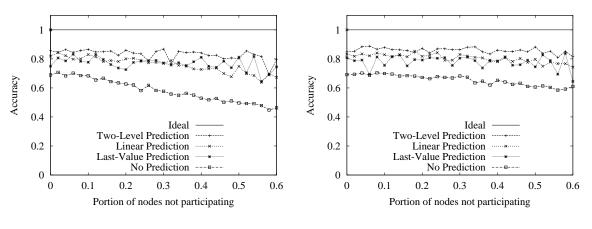
Figure 7 (a) shows the tradeoff between accuracy and resource consumption for the MAX operation under uniform participation and subtree-size based participation approaches. As fewer nodes participate, overall accuracy decreases for all algorithms. As expected, subtree-size based participation performs much better than uniform participation. Figure 7 (b) shows similar results for the AVG operation. Comparing the two figures, we can see that for the same participation rate, results of the AVG operation are generally more accurate than those of the MAX operation. This is as expected, since the AVG operation is less sensitive to data loss than MAX.
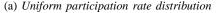
These results confirm that simple techniques to exploit temporal correlation can yield good approximation results despite partial node participation. Tuning the participation rate gives applications fine-grained control over

(a) *Uniform participation rate distribution*   (b) *Subtree-size participation rate distribution*

Fig. 8. MAX aggregation operation accuracy using different prediction algorithms on the stock dataset in a lossless network.

(a) *Uniform participation rate distribution*   (b) *Subtree-size participation rate distribution*

Fig. 9. MAX aggregation operation accuracy using different prediction algorithms for the stock dataset, on a network with an average 20% loss rate. The loss rate ratio of transit-transit links to transit-stub and stub-stub links is 3:1.

the tradeoff between aggregation accuracy and resource consumption. For optimal results, the distribution of a node's participation rate should be proportional to the amount of data the node is aggregating (*i.e.*, the size of its subtree), and the dynamicity of the data.

Next, we examine how various prediction algorithms perform in improving the accuracy of the MAX operation in a lossless environment, In this and all following figures, *No Prediction* means that if a parent node does not receive data from a child node, it just ignores the child and performs aggregation with data from the rest of other nodes. Figure 8 (a) shows the result of node non-participation using a uniform participation rate distribution, and Figure 8 (b) shows the result when using a subtree-size based participation rate distribution. We can see that all prediction algorithms are effective in improving aggregation accuracy, but two-level prediction algorithm performs the best, achieving an accuracy of more than 90%, even for node non-participation rate of

up to 60% (0.6). Compared to simple last-value prediction, the more complex and computationally expensive linear prediction algorithm performs worse because of its limited ability to cope with consecutive data loss.

We observe through experiments that prediction algorithms only improve accuracy up to a point. As shown in Figure 6, the amount of data reaching the aggregation root under uniform participation decreases exponentially with the rate of non-participation. Once the rate of non-participation increases beyond 0.6, less than 1% of data is getting to the top of the tree. At this point, so much data has been lost that any temporal correlation will be destroyed, and prediction algorithms will be rendered useless. The result is a "cliff" point past which prediction algorithms fail to provide any improvement in accuracy. Therefore we will only show non-participation up to 0.6 in the rest of our figures.

To demonstrate the value of prediction algorithms under lossy conditions, we plotted the accuracy im-
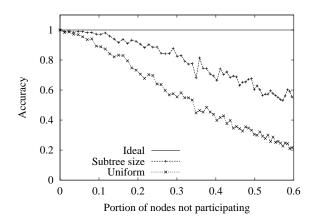
Fig. 10. Aggregate MAX operation accuracy for the random walk dataset versus participation rate distribution.

Fig. 12. Aggregate MAX operation accuracy for the temperature dataset versus participation rate distribution.

provement for MAX operation with node voluntary non-participation and prediction algorithms in a lossy environment. We added involuntary loss on each link, with an overall average link loss rate of 20%[2]. The link loss rate distribution is as follows: links between stub nodes have the same loss rate as those between stub nodes and transit nodes. The ratio of loss rate of links between transit-transit nodes to those between transit-stub nodes is $3 : 1$ (We also tried network with different loss settings and all results are similar). As in previous figures, Figure 9 (a) shows the results of different prediction algorithms with node non-participation using a uniform rate distribution, and (b) shows the result using a subtree-size based rate distribution. These results demonstrate that all prediction algorithms can improve accuracy significantly even in a high loss environment. And again, two-level prediction algorithm performs the best. In a high loss environment with average loss rates of up to 20%, two-level prediction can still achieve 80% of accuracy even with a 0.6 node non-participation rate. Linear prediction again underperforms the simple last-value prediction algorithm.

B. Application of Algorithms to Random Walk Trace

In this section, we perform selected experiments on the random walk dataset to verify the results obtained using the stock dataset. For these experiments, we generated 5,000 random walk data streams using the method in Section IV. Each stream is randomly mapped to a node in the transit-stub topology, as if the data were generated by this node.

Figure 10 shows the effects of different participation rate distribution algorithms on the MAX operation for
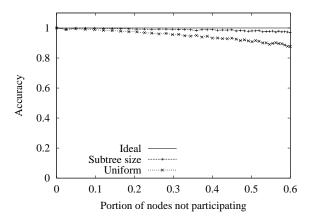
the random walk data trace. Figure 11 shows the performance of prediction algorithms for MAX operation under uniform participation rate distribution. Figure 11 (a) shows results for a lossless environment, and (b) shows results for a lossy environment. We observe that all results are consistent with those on stock dataset in following respects:
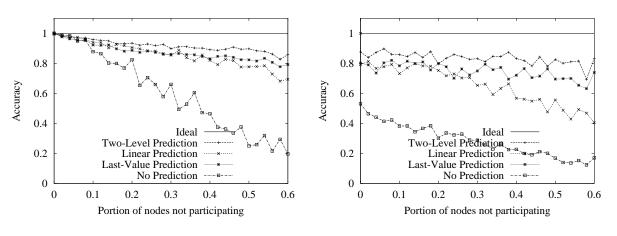
- As fewer nodes participate, overall accuracy decreases for all algorithms, and subtree-size based participation performs much better than uniform participation.
- All prediction algorithms are effective in improving aggregation accuracy.
- Two-level prediction performs the best.
- Linear prediction performs worse than the simple last-value prediction algorithm in most cases.

C. Application of Algorithms to Temperature Trace

In this section, we perform selected experiments on the temperature dataset to verify results obtained on previous datasets.

Figure 12 shows the effect of uniform and subtree-size based participation rate distribution on the MAX operation. Figure 13 shows the performance of last-value, linear and two-level prediction algorithms for the MAX operation under uniform participation rate distribution. Figure 13 (a) shows results in a lossless environment, and (b) shows results in a lossy environment. From these figures, we can see that all results here are consistent with those obtained using the stock and random walk datasets. When comparing these results with those of stock dataset and random walk dataset, we can see that temperature dataset accuracies are less sensitive to node non-participation and data loss, because there is less dynamic range in temperature data streams, as we
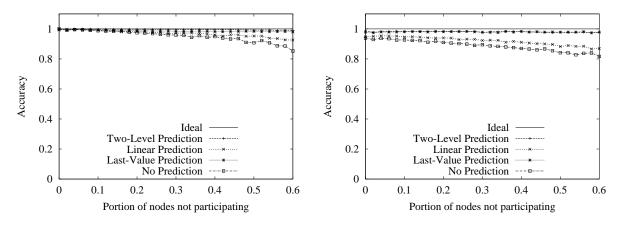
---

[2]Note that such high loss is common in wireless or sensor networks. High loss also occurs on the Internet in the presence of worm or DDos attacks.

(a) *Lossless environment*

(b) *Environment with 20% average loss rate.*

Fig. 11. Aggregate `MAX` operation accuracy on random walk dataset, using different prediction algorithms with node uniform participation rate distribution.

(a) *Lossless environment*

(b) *Environment with 20% average loss rate.*

Fig. 13. Accuracy of an aggregate `MAX` operation on temperature dataset, using different prediction algorithms with uniform participation rate distribution.

explored in Section IV. For such dataset with rich temporal correlation, simple last-value prediction algorithm is good enough to explore the correlation among data and is as effective as the complex two-level prediction algorithm.

D. Summary of Experiment Results

Using large datasets with different degrees of dynamics and temporal correlation, we ran extensive simulations to examine the performance of a set of prediction algorithms in a hierarchical tree-based data aggregation system.

For the temperature dataset, which has rich temporal correlation and is less dynamic, simple last-value prediction algorithm is enough to improve the accuracy, and is as effective as the complex two-level prediction algorithm. However, for data that contains less temporal

correlation and is more dynamic like the stock dataset, the more complex two-level prediction algorithm does a much better job of extracting short and long terms trends in order to significantly improve accuracy.

We conclude that across a wide variety of datasets, statistical estimation algorithms can exploit and leverage any existing temporal correlation to improve aggregation accuracy in both lossless and lossy environments. In our experiments, the two-level prediction algorithm we proposed outperforms all other techniques.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we propose the use of prediction techniques to recover from data loss in large-scale data aggregation networks. We evaluate the effectiveness of several prediction algorithms that exploit temporal correlation on real and artificial datasets. In addition, we

propose a novel multi-level prediction algorithm that extracts both short-term and long-term trends. Given the ability to recover from lost data items, we propose selective participation to reduce resource consumption and communication overhead in tree-based data aggregation networks.

Simulation results show that by adding statistical algorithms to hierarchical in-network aggregation, we can effectively improve accuracy in a lossy environment and achieve good approximation results with only a subset of nodes participating. Our main contributions are:

- We propose a multi-level prediction algorithm that significantly outperforms previous techniques to provide accurate results even under high loss and with many nodes not participating.
- We explore the tradeoff between algorithmic complexity and prediction performance across a variety of datasets with different dynamic properties.

By evaluating the performance of algorithms on different datasets, we clarify the fine-grained tradeoff between aggregation accuracy and the resources we want to allocate to this operation.

In our ongoing work, we are designing low-overhead wavelet algorithms for multi-level data decomposition, representation and prediction. We will also further explore tradeoffs between prediction accuracy and computation and storage cost. Finally, we are building a real system to support network health monitoring, traffic measurement, and router statistics aggregation applications, and plan to deploy and evaluate it on the Planet-Lab [33] testbed.

REFERENCES

[1] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *DARPA Information Survivability Conference and Exposition (DISCEX III)*, April 2003, vol. 1, pp. 303–314.
[2] D. Steere, A. Baptista, D. McNamee, C. Pu, and J. Walpole, "Research challenges in environmental observation and forecasting systems," in *Proc. of MOBICOM*, 2000, pp. 292–299.
[3] R. van Renesse and B. Kenneth, "Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining," *ACM TOCS*, vol. 21, no. 2, May 2003.
[4] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *SIGMOD Record*, vol. 31, no. 3, 2002.
[5] Y. Yao and J. Gehrke, "Query processing in sensor networks," in *Proc. of CIDR*, 2003.
[6] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. of MOBICOM*, Boston, MA, August 2000.
[7] J. Zhao, R. Govindan, and D. Estrin, "Computing aggregates for monitoring wireless sensor networks," in *Proc. of SNPA*, 2003.
[8] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," in *Proc. of OSDI*, 2002.
[9] J. Considine, F.-F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proc. of ICDE*, Boston, Massachusetts, March 2004.
[10] A. Sang and S. Q. Li, "A predictability analysis of network traffic," in *Proc. of INFOCOM*, 2000.
[11] A. Adas, "Supporting real time VBR video using dynamic reservation based on linear prediction," in *Proc. of INFOCOM*, San Francisco, CA, 1996.
[12] Y. Qiao, J. Skicewicz, and P. Dinda, "An empirical study of the multiscale predictability of network traffic," in *Proc. of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC 2004)*, 2004.
[13] I. Gupta, R. van Renesse, and K. Birman, "Scalable fault-tolerant aggregation in large process groups," in *Proc. of DSN*, 2001.
[14] F. Olken, *Random Sampling from Databases*, Ph.D. thesis, University of California, Berkeley, CA, 1993.
[15] J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online aggregation," in *Proc. of SIGMOD*, 1997.
[16] S. Chaudhuri, G. Das, V. Narasayya, M. Datar, and R. Motwani, "Overcoming limitations of sampling for aggregation queries," in *Proc. of ICDE*, Heidelberg, Germany, April 2001.
[17] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proc. of SIGMOD*, 2003, pp. 563–574.
[18] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Workshop on Mobile Computing and Systems Applications*, 1999.
[19] Antony Rowstron, Anne-Marie Kermarrec, Peter Druschel, and Miguel Castro, "SCRIBE: The design of a large-scale event notification infrastructure," in *Proc. of NGC*, Nov 2001.
[20] "National climatic data center," http://www.ncdc.noaa.gov/oa/ncdc.html.
[21] "Abilene connection traffic statistics," http://stryper.uits.iu.edu/abilene/.
[22] P. Strobach, *Linear Prediction Theory-A Mathematical Basis for Adaptive Systems*, Springer-Verlag, 1990.
[23] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, New Jersey, 1986.
[24] G. Farin, J. Hoschek, and M. S. Kim, Eds., *Handbook Of Computer Aided Geometric Design*, Elsevier Science, 2002.
[25] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications*, Springer, 2000.
[26] S. Shetty and P. R. White, "Curvature continuous extensions for rational B-Spline curves and surfaces," *Computer Aided Design*, vol. 7, no. 23, pp. 484–491, 1991.
[27] J. Skicewicz, P. Dinda, and J. Schopf, "Multi-resolution resource behavior queries using wavelets," in *Proc. of HPDC*. IEEE, August 2001, pp. 395–405.
[28] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. of the INFOCOM*. IEEE, 1997, p. 1405.
[29] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. of SIGMETRICS*. ACM, June 2000, pp. 1–12.
[30] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in a cooperative environment," in *Proc. of SOSP*, October 2003.
[31] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. of INFOCOM*. IEEE, 1996.
[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, second edition, 2001.

[33] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe, "A blueprint for introducing disruptive technology into the internet," in *Proc. of HotNets-I*. ACM, 2002.