Prediction or Not? An Energy-Efficient Framework for Clustering-based Data Collection in Wireless Sensor Networks

Hongbo Jiang, *Member, IEEE,* Shudong Jin, *Member, IEEE,* and Chonggang Wang, *Senior Member, IEEE*

Abstract—For many applications in wireless sensor networks, users may want to continuously extract data from the networks for analysis later. However, accurate data extraction is difficult – it is often too costly to obtain all sensor readings, as well as not necessary in the sense that the readings themselves only represent samples of the true state of the world. Clustering and prediction techniques, which exploit spatial and temporal correlation among the sensor data, provide opportunities for reducing the energy consumption of continuous sensor data collection. Integrating clustering and prediction techniques makes it essential to design a new data collection scheme so as to achieve network energy efficiency and stability.

We propose an energy-efficient framework for clustering-based data collection in wireless sensor networks by integrating adaptively enabling/disabling prediction scheme. Our framework is clustering based. A cluster head represents all sensor nodes in the cluster, and collects data values from them. To realize prediction techniques efficiently in wireless sensor networks, we present adaptive scheme to control prediction used in our framework, analyze the performance tradeoff between reducing communication cost and limiting prediction cost, and design algorithms to exploit the benefit of adaptive scheme to enable/disable prediction operations. Our framework is general enough to incorporate many advanced features; and we show how sleep/awake scheduling can be applied, which takes our framework approach to designing a practical algorithm for data aggregation: it avoids the need for rampant node-to-node propagation of aggregates, but rather it uses faster and more efficient cluster-to-cluster propagation. To the best of our knowledge, this is the first work adaptively enabling/disabling prediction scheme for clustering-based continuous data collection in sensor networks. Our proposed models, analysis, and framework are validated via simulation and comparison with competing techniques.

Index Terms—Sensor networks, algorithm/protocol design, clustering, adaptive prediction.

1 Introduction

Wireless sensor networks(WSNs) have a broad range of applications, such as battlefield surveillance, environmental monitoring, and disaster relief. A sensor network consists of a set of autonomous sensor nodes which spontaneously create impromptu communication links, and then collectively perform tasks without help from any central servers.

In sensor networks, accurate data extraction is difficult – it is often too costly to obtain all sensor readings, as well as not necessary in the sense that the readings themselves only represent samples of the true state of the world. As such, one technique so-called prediction emerges to exploit the temporal correlation of sensor data. Technology trends in recent years have resulted in sensors' increasing processing power and capacity [3]. Implementing more sophisticated distributed algorithms in a sensor network becomes possible. One important class of such algorithms are predictors, which use past input values from the sensors to perform prediction operations. The existence of such prediction capability implies that the sensors do not need to transmit the data values if they differ from a predicted value by less than a certain prespecified threshold, or error bound.

- *H. Jiang is with Huazhong University of Science and Technology, 430074. China. E-mail: hongbojiang2004@gmail.com*
- *S. Jin is with Case Western Reserve University. OH 44106. U.S.A.*
- *C. Wang is with NEC Laboratories America, Princeton, NJ 08540. U.S.A.*

A simple approach to developing a predictor in sensor networks is simply to transmit the data from all sensors to the base station (i.e., the sink), which has been realized in many previous studies [16], [24], [5]. Predictor training and prediction operations are carried out by the base station only, but not the sensor nodes, despite their increasing computing capacity. This solution, while practical, has many disadvantages, such as a high energy consumption incurred by transmitting the raw data to the base station, the need for wireless link bandwidth, and potential high latency.

One solution is clustering-based localized prediction [26], where a cluster head, also a sensor node, maintains a set of history data of each sensor node within a cluster. We expect the use of localized prediction techniques is highly energy-efficient, due to the reduced length of routing path for transmitting sensor data. On the downside, clustering-based local prediction in sensor networks faces a couple of new challenges. First, since the cost of training a predictor is non-trivial, we should carefully investigate the tradeoff between communication and computation. To support prediction techniques, energy is consumed on communication (e.g., sending and receiving sensor data) and computation (e.g., processing sensor data and calculating a predicted value). Motivated by this observation, we analytically study how to determine whether prediction techniques is beneficial in this paper. We qualitatively derive sufficient conditions for this, and reveal that the decision is a function of both the desired error bound and the correlation among the sensor data values. For instance,

Digital Object Indentifier 10.1109/TPDS.2010.174

when the error bound is very tight or the correlation is not significant, a sensor node always has to send its data to the cluster head. The second challenge is due to the characteristics and inherent dynamics of the sensor data. When the data distribution, in particular the data locality, evolves over time, prediction techniques may not work well for a set of less predictable data. Global re-clustering is costly if it is initiated periodically. We propose an algorithm for dynamic updates of clustering, and the algorithm requires mostly local operations and very low communication cost. This adaptive update of clustering is expected to facilitate clustering-based localized prediction by maintaining the similarity within clusters at low communication cost.

The rest of the paper is organized as follows. In Section 2, we describe related work on prediction and clustering techniques in sensor networks. In Section 3, we describe the models, analysis, and algorithms our framework. Section 4 discuss the implementation issues, and describes the application of our framework to the design of more efficient and scalable data aggregation algorithms and sleep/awake scheduling. Section 5 provides a performance comparison of different techniques. Finally, we conclude the paper in Section 6.

TABLE 1
Symbols used in this paper

ϵ	Error bound.
x_t	Sensor reading at time instance t. μ_x is the mean value and σ_x^2 is the variance.
$\hat{x}_t$	The predicted data value of x_t.
$P(\cdot)$	Linear predictor.
c_p	The cost of a single prediction.
α	Confidence level.
$\Phi(\cdot)$	The CDF of Gaussian white noise.
VAR$(\cdot)$	The variance.
STD$(\cdot)$	The standard deviation.
m, p	The order AR predictor.
γ	The covariance.
ρ	The correlation coefficient.
k	The ratio between transmission cost and computation cost.
δ	δ-compactness for clustering.
CH	Cluster head.
F_{CH}	The feature of cluster.
r_{CH}	The radius of this cluster.
$d(\cdot,\cdot)$	The distance between two features.

2 Related Work

Energy conservation is crucial to the prolonged lifetime of a sensor network. Many approaches for energy-efficient monitoring have been explored to minimize energy consumption. One class of techniques— prediction-based algorithms, is based on the observation that the sensors capable of local computation create the possibility of training and using predictors in a distributed way [16], [24], [5], [2], [21]. Taking lessons from MPEG encoding process, Goel *et al.* [5] proposed a prediction-based monitoring mechanism in sensor network. McConnell *et al.* [16] proposed that each sensor transmits to the base station the predicted target class rather than the entire raw data. Chu *et al.* [2] proposed a robust approximate technique that uses prediction models to minimize communication from sensor nodes to the base station. Likewise, Silberstein *et al.* [21] proposed the data-driven processing to provide continuous data without continuous reporting. To do this, they developed a suppression strategy that adopts models for optimization of data collection.

One alternative approach to selecting representative sensor data is using clustering technique [28]. With clustering, only cluster heads need to communicate with the base station via multi-hop communication. Several clustering algorithms have been designed, with particular attention to energy-efficient query processing. Younis *et al.* [27] proposed a protocol called HEED where a node uses its residual energy as the primary parameter to probabilistically elect itself to become a cluster head. The LEACH protocol [6] is an application-specific clustering protocol, which has been shown to significantly improve the network lifetime. Hussain *et al.* [8] extended LEACH to a hierarchical clustering-based routing (HCR) technique. Kuhn *et al.* [14] proposed a probabilistic technique to select cluster heads, in which the probability is dependent on the node degree.

One work that is close to our work is ASAP [4]. They also consider clustering such that nodes with similar sensor data values are assigned to the same clusters. In addition, adaptive data collection and model-based prediction are used to minimize the number of messages used to extract data from the network. However, we emphasize the differences from their work. First, and most importantly, we provides an error-bounded data collection scheme. We are uniquely interested in when the prediction scheme really benefits data collection, and have derived solutions after careful energy-aware analysis. Second, [4] focused on clustering and cluster head selection. We present a scheme for dynamic update of clustering. Third, we developed our framework integrating data aggregation to demonstrate the usefulness of our framework. Finally, we must note that the our approach is not exclusive. Instead, it can used in conjunction with other techniques, for example, the clustering and cluster head selection algorithms in [4].

3 Our Proposed Framework

Our framework consists of two main functional components: (1) data processing and intra-cluster prediction. It is noted that unlike previous dual-prediction techniques, our prediction operation can be enabled/disabled to achieve energy efficiency, and (2) adaptive cluster split/merge. Table 1 lists symbols used in this paper.

3.1 Adaptive Scheme to Enable/Disable Prediction Operations

Consider a cluster of sensor nodes, which can be awake or sleeping. If the sensor nodes are sleeping, the prediction problem is reduced to estimating data distribution parameters using history data. In this case, it could well be the case that the estimates are already available. We can neglect this case.

If the sensor nodes are awake, they continuously monitor an attribute x and generate a data value x_t at every time instance t. Without local prediction capability at the cluster head, a sensor node has to send all data values to the cluster head, who estimates data distribution accordingly. With local prediction, however, a sensor node can selectively send its data values to the cluster head. One model for selective sending is ϵ-loss approximation: Given an error bound $\epsilon > 0$, a sensor node sends its value x_t to the cluster head if $|x_t - \hat{x}_t| > \epsilon$, where $\hat{x}_t$ is a predicted representative data value to approximate the true data. The intuition of this choice is that, if a value is close to the predicted value, there is not much benefit by reporting it. If the value is much different from the predicted value, it is important to consider it for computing the data distribution[1]. A question to ask is, is prediction really energy-efficient by trading computation power for communication saving?

To solve this problem, we first develop a localized prediction model. Very complex models are not practical in our application due to the limited computational capacity of sensor nodes. Fortunately, simple linear predictors are sufficient to capture the temporal correlation of realistic sensor data, as shown by previous studies [2], [17]. A history-based linear predictor is one of popular approaches to predicting the future based on past n measurements: $\hat{x} = P(x_{t-1}, x_{t-2}, \cdots, x_{t-n})$. In particular, it has the properties: (1) $k\hat{x}_t = P(kx_{t-1}, kx_{t-2}, \cdots, kx_{t-n})$, and (2) $P'_{x_{t-i}}(\cdot) = c$, where c is constant, and $\sum_{i=1}^{n} P'_{x_{t-i}}(\cdot) = 1$ if the predictor is unbiased. One of the examples, autoregressive model, denoted by AR(p), is written as $x_t = c + w_t + \sum_{i=1}^{p} \phi_i x_{t-i}$ where $\phi_1, \cdots, \phi_p$ are the parameters of the model. Here c is a constant, always presumed to be zero, and w_t is a white noise process with zero mean and variance σ^2. The process is covariance-stationary if $|\phi_i| < 1$. Accordingly, a p-order AR predictor is: $\hat{x}_t = \sum_{i=1}^{p} \phi_i x_{t-i}$. The parameters can be calculated by Yule-Walker equations [18], or using the least square method. The parameters are often updated on every measurement and the estimation is carried out by both the cluster header and cluster members to achieve synchronization.

With this model, the first question to be answered is: *Given an error bound ϵ, what is the confidence level to which we can trust the predicted value?* First, for a linear predictor, we prove that the error of a m-step prediction is less than m times that of an one-step prediction.

Lemma 1 Given a linear predictor P, if the condition $\sum_{i=1}^{p} |\phi_i| \leq 1$ holds, a m-step prediction error is less than m times the one-step prediction error.

Lemma 1 implies that even all m previous values are predicted, the cumulative error of m-step prediction is not significant. The variance of a m-step prediction error is obtained similarly: $\mathbf{VAR}[e(m)] \leq m^2\sigma^2$. The confidence interval is $\hat{x}_t \pm \Phi^{-1}(\alpha)m\sigma$, where Φ is the cumulative distribution function of Gaussian white noise. Conversely, given an error bound ϵ, the predictor can provide data at a confidence level of $\alpha_m = 2\Phi(\frac{\epsilon}{\mathbf{SDV}[e(m)]}) - 1$. (**SDV** is the standard deviation)

After the prediction model is formulated, we look at the algorithm selection problem. Without local prediction, all sensor nodes will send original data values to the cluster head. This scheme incurs significant communication cost. With local prediction, the communication cost is reduced by selectively sending data to the cluster head, but the computation cost can be prohibitive. Clearly, there is a tradeoff between the two schemes, and the question is which is more energy-efficient?

Let c_p denote the cost of a single prediction. Let the transmission cost be k times c_p, where $k >> 1$. We have

Theorem 1. *If the error bound ϵ satisfies $\Phi(\frac{\epsilon}{m\sigma}) > \frac{k+1}{2k}$, the scheme with local prediction is more energy-efficient.*

A careful investigation of above results reveals that this model is not complete since in practice the variance σ is unknown. To solve this problem, we turn to analysis based on covariance and correlation coefficient. Consider a stationary time series x with mean μ_x, variance σ_x^2, and covariance $\gamma_x(j)$. The correlation coefficient is $\rho_x(j) = \gamma_x(j)/\sigma_x^2$. The one-step error variance is: $\mathbf{VAR}[e(1)] = \sigma_x^2(1 - \sum_{j=1}^{p} \phi_j \rho_x(j))$. Accordingly, $\mathbf{VAR}[e(m)] \leq m^2\sigma_x^2(1 - \sum_{j=1}^{p} \phi_j \rho_x(j))$. We can therefore eliminate the unknown σ in Theorem 1, but have

Corollary 1: If the error tolerant ϵ and correlation coefficients $\rho_x(j)$ satisfy

$$\frac{\epsilon}{\sqrt{1 - \sum_{j=1}^{p} \phi_j \rho_x(j)}} > m\sigma_x \Phi^{-1}(\frac{k+1}{2k}), \qquad (1)$$

the scheme with local prediction is more energy-efficient.

This result says, if the correlation coefficient is too small, prediction will not be accurate. As a result, sensor data values are often not within the error bound and will still be transmitted to the cluster head. Meanwhile, if the error bound is small, the condition is not easily satisfied either, and transmissions are still required. Together, this corollary tells that algorithm selection should be determined based on both the desired error bound and the predictability of the sensor data. In addition, we found from experimental results that the effect on the parameter m is not deterministic. While a large value of m often leads to the condition of (1) hardly satisfied and thereby the prediction scheme at the node is prone to being disabled, it can reduce the energy consumption as a long term predictor when the condition is satisfied.

Figure 1 shows the pseudo-code description of the algorithms at the cluster head. The cluster head maintains a set (a circular array) of history data for each cluster member. Lines (08)–(12) show, the cluster head will continuously receive data values from each cluster member to update the set of history data, or when no data values are received, will use the predicted value instead for update. The cluster head also runs a periodic process, Lines (01)–(06), to determine algorithm selection, with or without local prediction. The decision is broadcast to all cluster members[2].

Figure 2 shows the pseudo-code description of the algorithms at each cluster member. Each cluster member maintains a set of history data of its own. If the algorithm selection is "no local prediction", it simply transmits the data values. If local prediction is turned on, the cluster member will perform

1. This is especially true for outlier detection [22], [20].

2. To reduce the cost, a decision may not be broadcast if it is the same as the previous one.

```
Process at the cluster head
01: if timeout after m * Δ seconds
02:    for each member i in the cluster
03:       if condition (1) holds
04:          send message to member i to enable prediction
05:       else
06:          send message to member i to disable prediction
07: else
08:    for each member i in this cluster
09:       if receive a data value from member i
10:          update the history data for member i
11:       else
12:          perform prediction to update the history data
```

Fig. 1. Operations at the cluster head.

prediction on each data value. If the data value is not within the error bound, it will be sent to the cluster head too. Meanwhile, the local set of history data should be updated as well. In particular, if local prediction is enabled and the data value is within the error bound, the predicted value, not the actual value, will be included in the set of history data. The purpose is to maintain the consistency between local and the cluster head.

```
Process at the cluster members
01: if prediction is disabled or |x_i − x̂_i| > ε
02:    send the data value to the cluster head
03:    update the history data using the data value
04: else
05:    perform prediction to update the history data
```

Fig. 2. Operations at the cluster members.

3.2 Adaptive Update of Clustering

Although we state that many clustering algorithms can be used in our framework, adaptive update of clustering is often required to capture the change in locality patterns. A complete re-clustering is an option, but can also be expensive. Not only it involves the establishment of clustering map for all sensors, the complete change in cluster membership also implies all history data and models must be constructed from scratch. In this section, we present algorithms for the dynamic split and merge of clusters, which require low communication cost.

Let us first formulate the problem. Let F_i be the feature of node i and F_{CH} be the feature of a cluster head. In our case, the feature is defined to be the coefficients of AR model. The similarity between two feature F_i and F_{CH} is defined by the distance $d(F_i, F_{CH})$. Given a real number δ, a δ-clustering (or δ-compactness) means for any two nodes i and j in a cluster, $d(F_i, F_j) \leq \delta$. Since re-clustering is expensive, local operations such as split/merge could be performed to avoid global computation. The problem is that the design of such local operations should guarantee that they will not lead to violations of the δ-clustering condition.

For the algorithm of splitting and merging clusters, we consider two cases. First, if the δ-clustering condition is violated, re-clustering within the cluster is necessary, and the cluster will be split. Local split requires less communication cost compared to global re-clustering. In particular, since in our framework, each cluster head maintains sensor data for each of its cluster members, it has fresh knowledge on the distance $d(.)$ to the cluster member. Thus, our framework does not incur much communication cost. When a split occurs, say, a cluster with head CH_i is split into multiple clusters with heads $CH'_1, CH'_2, \cdots$. The new cluster head nearest CH_i will inherit its position in the routing tree, and all other new cluster heads will become its children.

In the second case, clusters may be merged. Each cluster head will check whether a pair of children can be merged, or whether itself should be merged with a child. Let us say we need to merge two clusters, with CH_i and CH_j as cluster heads, and assume the first cluster is larger. CH_i will become the head of the merged cluster (this requires the update of all membership changes, e.g, via a broadcast). In addition, history data of the corresponding nodes will migrate from CH_j to CH_i.

4 Discussions

4.1 Implementation Details

4.1.1 Estimation of the Parameter k

One of the biggest concerns in our framework is the estimation of k, which is the ratio between communication cost and prediction cost. We acknowledge this estimation is not straightforward and may depend on the applications. For instance, some recent works [13], [23] discuss the tradeoff between computation cost and communication cost. We do not elaborate the optimization of this tradeoff as we found previous works validated a variety of values for this ratio k. As a concrete instance, in [7], the authors simulated the LEACH protocol for the random network with 50nJ/bit for energy dissipated for both transmitter and receiver, as well as a computation cost of 5nJ/bit with local compression. In this sense, this ratio k is around 10. Their study showed that energy dissipation can be reduced through intelligent communication and computation energy tradeoff. Besides, [25] reported often commercial radios consume around 150nJ/bit for communication versus the StrongARM dissipating 1nJ/bit. As such, 1 bit communication is thus equivalent to performing 150 instructions. As AR(3) model is typically used in our implementation, this means that the ratio k could be roughly 150/7≈21. Overall, we emphasize this ratio k is varying according to the application and the hardware configuration.

Here we propose one possible solution by applications and do not claim it is exhaustive(we do so in anticipation of further research exploiting the techniques we herein apply as well as to encourage more research to bring new techniques to bear on this problem space). In realistic systems, the node itself periodically estimates the ratio between communication cost and computation cost according to the residual energy. That is, the energy consumption for a predictor include: computation cost when performing prediction, flash read/write cost, and

radio start-up/shut-down cost [19]. Additional solution for estimation is that, for simplicity, this value can be viewed as an input from system operator after trial experiments.

4.1.2 The Scenario with Packet Loss

Failure may not be rare in wireless sensor networks. Clearly, if an update message is lost, that is, the message from Line (04) or (06) in Figure 1 is lost, the dual prediction (cluster head and cluster member) will not be correct. In that case, each node will perform a different prediction, therefore, leading to a possible misbehavior. This is a key issue that need to be addressed in applications. One possible solution is by the cluster member replying a small ACK message to cluster head. As for the packets containing sensor readings, as long as the packet loss rate is not significant and approximation is acceptable, the impact of failure could not be crucial. As an example, we found in our previous aggregation work [9] that a small packet loss rate does not have significant impact on the final results. As a result, we focus on the discussion of adaptive scheme to control prediction and adaptive cluster provided justification for not considering packet loss.

4.1.3 Adaptive Update for System Input Changes

It it noted we do not claim a set of fixed parameters for the linear predictor and for the value of error bound ϵ. In practice, for many applications the model parameters and the error bound could be dynamics after setup. For instance, the system operator may not satisfy an initial error bound ϵ and want to adjust it after the system has been set up for a long time. In that case, the cluster head, after receiving the updated system input from the sink, should re-estimate the model parameters and diffuse to the cluster members. This adjustment is therefore performed locally.

4.2 Accommodation with Sleep/Awak Scheduling

This variation with sleep/wake scheduling is based on the following observation. For some applications, the ϵ-approximation may not be strictly required. The applications may tolerate few, if any, missing data values not within the error bound. If the confidence level (of having data values within the ϵ error bound) is very high, e.g., above a specified threshold, say $\alpha_{\text{threshold}}$, the cluster members may never report data values to the cluster head. Therefore, there is no paramount need for the cluster members to stay awake to obtain data values, most of which will be discarded anyway.

To allow sleep/wake scheduling for the cluster members, we replace Lines (01)–(06) in Figure 1 by Lines (01')–(07') in Figure 3, and by default, disable local prediction at cluster members. When a cluster member is awake, the cluster head checks if the member's data values are within the error bound with high probability. If yes, the cluster head will send a message to power off the member. The condition should be, the confidence level α_m is higher than the threshold $\alpha_{\text{threshold}}$, i.e.,

$$2\Phi\left(\frac{\epsilon}{m\sigma_x\sqrt{1-\sum_{j=1}^{p}\phi_j\rho_x(j)}}\right)-1>\alpha_{\text{threshold}}. \qquad (2)$$

Algorithms at the cluster head

```
// ...
// sleep scheduling for members, Lines (01')–(07')
01': while member i is awake
02':   if timeout after m * Δ seconds
03':     if condition (2) holds
04':       let member i power off for m_f * Δ seconds
05': while member i is sleeping
06':   if timeout after m_f * Δ seconds
07':     awake member i
```

Fig. 3. A variation with sleep/wake scheduling.

However, when the cluster members sleep, the cluster head will not receive any data values, and hence it is impossible to perform accurate prediction. For this reason, periodic but infrequent collection of data from the cluster members is still necessary. The frequency of this infrequent data collection is due an optimization problem: if the frequency is high, the cost of collecting data can also be high; if the frequency is low, the prediction can be inaccurate and result in erroneous sleeping decisions. In this paper, we provide only a heuristic solution to the problem. Let Δ be the time interval between two consecutive reportings by a member. We set the duration of a sleep period to $m_f * \Delta$, and when a cluster member wakes up, it will continuously perform data reading (and possibly reporting) for the next $m * \Delta$ time. Initially, m_f is set to m. It can be increased if condition (2) consistently holds, or decreased if the condition does not hold.[3]

4.3 Accommodation with Data Aggregation

With our framework, each cluster head will maintain the representative data values for all members in the cluster. This approach eliminates the need for collecting all data values from individual members when the sink wishes to obtain them. Instead, the cluster heads can form a routing tree, along with a small number of non-head nodes en routed. There are many potential applications of our proposed framework, such as outlier detection and sensor queries.

To demonstrate the utility of our framework, in this section, we use data aggregation as an example, which often exploits the spatial correlation of sensory data. Here, the sink would like to obtain the overall distribution of the sensor data in the entire network. A traditional in-network aggregation technique would require building a routing tree to contain all sensor nodes, and then the summary information is propagated towards the sink. Each sensor node, on receiving the summary information from its downstream descendants in the routing tree, can aggregate the summary information, and further propagate the aggregate. With our framework, we avoid the need for rampant node-to-node aggregation; rather, it uses faster and more efficient cluster-to-cluster aggregation.

We develop our aggregation algorithm based on the framework in Section 3, by combining our framework and our

3. It is also important to avoid waking up the cluster members at the same time, since there can be wireless communication interference among them. One simple idea is to schedule different sleeping times for all members.

previous aggregation scheme [9]. When the objective is to obtain the statistical information of the sensor data, it suffices if the aggregation algorithms return the probability distribution of the data. Thus, a packet only contains a few parameters of the data distribution. The aggregation process starts at the remote cluster heads towards the sink. A remote cluster head first sends packets containing the parameters of the local data distribution to its parent. Intermediate cluster heads, upon receiving packets from other cluster heads, will aggregate them (along with its own data). For the parameters estimation problem, the input information includes both the values acquired from sensors within the local cluster, and the values received from its children in the routing tree. A cluster head then uses the mixture model and the Expectation Maximization (EM) algorithm, which is standard for finding maximum likelihood estimates of parameters in probabilistic models. After parameter estimation, the cluster next sends packets to its parent too. This distributed, iterative process continues until the sink receives and aggregates the final results.

5 Performance Evaluation

To evaluate the performance of our proposed framework, we conduct a series of experiments. We have implemented the simulator, and in this paper we quantify several performance aspects of our algorithms. The simulation parameters are set based on hardware configurations of MICA2 [3]. Since for Intel Lab Data, sensor node reports the data every 30 seconds, we measure the energy consumption on the unit of 30 seconds. During every 30 seconds, the energy consumption is 240mJ for active power, and 2.4mJ for inactive power. For transmission, the energy consumption is 10mJ/byte, and thus 80mJ per message (we assume the value is a 8-byte double-precision number). Besides, our proposed framework is inserted on top of the data routing layer. In this paper we do not consider the mobile sensors. The communication model we used in this paper is unit disk modeling which is typically used in many previous works. In the rest of this section, we first describe the cluster model used in this work, and then describe our data-sets and alternative techniques, and then present the results and analysis.

5.1 Cluster model

With our framework, a sensor network is partitioned into multiple subnetworks, i.e., clusters. A cluster can be formed by the set of sensor nodes in a geographical area, where data locality exists among the sensor nodes, and clusters are dynamically split/merged to maintain good locality within each cluster. All sensor nodes in a cluster are called cluster members, including one elected cluster head. Within each cluster, the cluster head receives data selectively reported by all cluster members, and performs local prediction on the data distribution of the sensor data. Cluster members also perform prediction, and data values are transmitted to the cluster head only if they are not within a specified error bound. In this way, a cluster head can perceive an accurate view of all sensor data across the cluster, while communication cost is drastically reduced.

It is noted that our framework is clustering-based, but the construction of initial clusters is not an objective of this work. Numerous previous studies have focused on sensor node clustering algorithms and cluster head selection. We bear this in mind and state that many of these algorithms can be used in conjunction with our framework. Our implementation in this work of cluster model is similar to [17]. Briefly speaking, in the initial phase, each node of some randomly selected candidates expands its cluster until it is δ-compact and this node becomes the cluster head and the rest nodes in this cluster become cluster members. This process is recursively perform till each node belongs to one cluster exactly. After that, as we mentioned in Section 3.2, the cluster will be updated adaptively due to the changes of the data distribution (that is, the locality pattern).

5.2 Data-sets

Intel Berkeley Lab Data: This publicly available sensor data-set was collected by the Intel Berkeley Research Lab during a one-month period [1]. The data consists of environmental data regularly collected from 54 nodes spread around their lab. We observed some missing data values for various nodes at different time epochs, and filled there in with the average of the values during the previous and subsequent epochs at the same node. In our simulation, we select the entire temperature records during a one-week period (Feb 28 to Mar 5). We choose a node close to the center of the area as the sink in each experiment.

Synthetic data: To evaluate the algorithms in larger networks and with larger data-sets, we also generated synthetic networks and synthetic data. The size of synthetic networks ranges from 100 nodes to 1,000 nodes. We have used a random placement of nodes with a uniform probability distribution. Each node has on the average 5 nodes within its radio range. Data at every node i is modeled as, $x_t = \alpha_i x_{t-1} + e_t$ where $e_t \sim N(0, 0.1)$ and $\alpha_i \sim N(1, 0.01)$. For each node, 2,880 data values were generated. Every node is initialized with $\alpha_1 = 1$ and $x_0 = 0$. This model is updated on every measurement.

5.3 Alternative Techniques

Centralized exact: In TinyDB [15], all sensor values are always reported to the sink. This technique offers an error-free propagation of data.

Centralized prediction: This technique [2] caches the last a few reported data values at the sink and sensors. The sensor does not report a data value if it is within an error bound. For fair comparison, we use a simple predictor AR(3) for this technique, as used in our framework. In both this technique and our framework, we assume the sink (cluster heads for our framework) and sensors can cache 10 history data values, and the error bound is 0.2^oC.

GMM without clustering: This technique [9] is to transmit a data distribution instead of exact data values along a routing tree. Via hop-by-hop propagation and aggregation, each sensor sends data distribution parameters to its parent. The base station is able to obtain the data distribution information at

low communication cost. Unless otherwise noted, our implementation is based on 2-GMM model for this technique. For fair comparison, our framework integrating data aggregation also uses 2-GMM model for inter-cluster aggregation, and the two techniques have comparable message sizes.

Without adaptive scheme to control prediction operations: Unless otherwise noted, we evaluate our framework with local prediction control, which guarantees ϵ-approximation. Figure 1, Lines (01)–(06), shows how a cluster head adaptively enables or disables local prediction. We will evaluate the benefits of using this adaptive scheme in Section 5.4. In parallel, we label by "w/o adaptive scheme" the case where the cluster head does not perform adaptive scheme, that is, Lines (01)–(06) in Figure 1 are omitted and the prediction operations are always performed at cluster member.

5.4 Benefits of Adaptive Scheme to Enable/Disabe Prediction

We first investigate the effectiveness of our framework in reducing the energy consumption. To that end, we compare the energy consumption with/without adaptive scheme to control local prediction. Since our energy-aware scheme only benefits cluster members instead of cluster heads, we only measure the energy consumption at cluster members during the entire day of Feb 28. On average, there are 13 clusters and 40 cluster members. We output the sum of the energy consumption of all these 40 cluster members. As for our framework with control of sleep scheme, we set the confidence level threshold $\alpha_{\text{threshold}} = 0.9$ for nodes to sleep.

We vary the value of k to investigate the performance sensitivity. Note that k represents the ratio between the transmission energy consumption and the prediction energy consumption. As an example, for $k = 10$ or $k = 100$, the average energy consumption is 8mJ or 0.8mJ to predict a value from the history data. Figure 4 shows that energy consumption with different k values. First, it shows that energy consumption is a decreasing function of k. Second, the energy-aware technique is more beneficial when k is smaller. We emphasize that while communication is more expensive compared to prediction, the scheme is still applicable due to the existence of other computational operations (e.g., calculating coefficients, maintaining/updating history data, etc) not mentioned in this work. Finally, accommodating with sleep scheduling improves the performance by up to 10%, mainly because we set the confidence level threshold at 90%.

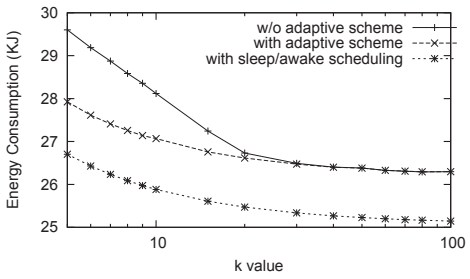


Fig. 4. Energy consumption with/without energy-awareness.

5.5 Accommodating to Data Aggregation

5.5.1 Overall Communication Costs

In this section, we evaluate the performance of the algorithms in terms of the total number of transmitted packets by all sensor nodes. For brevity, we only report the representative results.

Figure 5 shows the scalability of the algorithms with the size of network using the synthetic data-set. We see the superior scalability of our framework integrating data aggregation. This is because our distributed techniques perform data update and prediction locally, whereas the centralized scheme (labeled "centralized exact") incurs a high communication cost for transmitting the data to the sink.

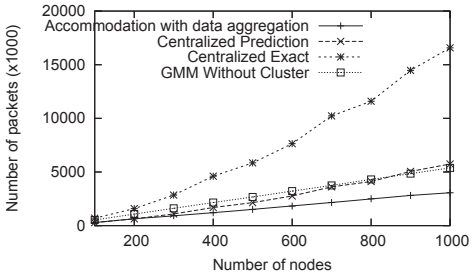


Fig. 5. Total number of packets required for a variety of algorithms.

6 Conclusions

We have proposed and described our framework for clustering-based data collection. Our framework is (1) clustering-based: sensor nodes form clusters and cluster heads collect and maintain data values, and (2) prediction-based: energy-aware prediction is used to find the subtle tradeoff between communication and prediction cost. We have presented the detailed analysis and description of its two main components: *adaptive scheme to enable/disable prediction operations* and *adaptive update of clustering*, and further demonstrated the usefulness of our framework by accommodating to in-network aggregation and the sleep/awake scheduling. Via performance evaluation, we have shown that it achieves energy efficiency when sensor data is spatially and temporally correlated. To summarize, we have demonstrated that this is a viable framework to facilitate data collection in large-scale wireless sensor networks.

There are several future directions. First, we plan to do real testing in real sensor networks. Second, we are seeking the possibility of using more efficient algorithms to reduce the computational overhead of our prediction and aggregation techniques. Third, we plan to integrate other techniques such as skeleton extraction [11], [12] to improve the quality of clustering. The fourth is to further improve the framework in order to facilitate other realistic applications such as object tracking [26].

Acknowledgment

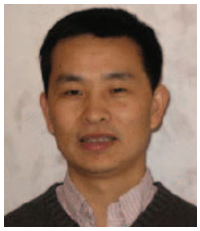
This work was supported in part through National Natural Science Foundation of China (No.60803115, No. 60873127),

the Fundamental Research Funds for the Central Universities (No.M2009022), the Youth Chenguang Project of Wuhan City (No.201050231080), the CHUTIAN Scholar Project of Hubei Province, and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry. An earlier version of this work appeared as [10].

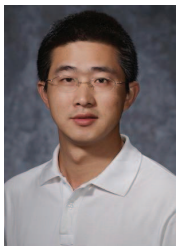
REFERENCES

[1] http://db.lcs.mit.edu/labdata/labdata.html.
[2] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *Proceedings of IEEE ICDE*, 2006.
[3] Crossbow. MICA2 wireless measurement system datasheet. 2003.
[4] B. Gedik, L. Liu, and P. S. Yu. ASAP: An adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and Distributed Computing*, 18(12):1766–1783, 2007.
[5] S. Goel and T. Imielinski. Prediction-based monitoring in sensor networks: Taking lessons from mpeg. *ACM SIGCOMM Computer Communication Review*, 31(5):82–98, 2001.
[6] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, 2002.
[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol forwireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences*, 2000.
[8] S. Hussain and A. W. Matin. Hierarchical cluster-based routing in wireless sensor networks. In *Proceedings of IPSN (Poster)*, 2006.
[9] H. Jiang and S. Jin. Scalable and robust aggregation techniques for extracting statistical information in sensor networks. In *Proceedings of IEEE ICDCS*, 2006.
[10] H. Jiang and S. Jin. Leap: Localized energy-aware prediction for data collection in wireless sensor networks. In *Proceedings of IEEE MASS*, 2008.
[11] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, Y. Wu, and W. Liu. Case: Connectivity-based skeleton extraction in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, 2009.
[12] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, Y. Wu, and W. Liu. Connectivity-based skeleton extraction in wireless sensor networks. *IEEE Transaction on Parallel and Distributed Systems*, 5, 2010.
[13] J. Koshy, I. Wirjawan, R. Pandey, and Y. Ramin. Balancing computation and communication costs: The case for hybrid execution in sensor networks. *Elsevier Ad Hoc Networks*, 6, 2008.
[14] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proceedings of ACM MOBICOM*, 2004.
[15] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.
[16] S. M. McConnell and D. B. Skillicorn. A distributed approach for prediction in sensor networks. In *Proceedings of SIAM International Conference on Data Mining Workshop on Sensor Networks*, 2005.
[17] A. Meka and A. K. Singh. Distributed spatial clustering in sensor networks. In *Proceedings of EDBT*, 2006.
[18] M. Pourahmadi. *Foundations of Time Series Analysis and Prediction Theory*. John Wiley and Sons, 2001.
[19] C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proceedings of ACM Sensys*, 2006.
[20] B. Sheng, Q. Li, W. Mao, and W. Jin. Outlier detection in sensor networks. In *Proceedings of ACM MOBIHOC*, 2007.
[21] A. Silberstein, R. Braynard, G. Filpus, G. Puggioni, A. Gelfand, K. Munagala, and J. Yang. Data-driven processing in sensor networks. In *Proceedings of CIDR*, 2007.
[22] A. Silberstein, K. Munagala, and J. Yang. Energy-efficient monitoring of extreme values in sensor networks. In *Proceedings of ACM SIGMOD*, 2006.
[23] M. Tahir and R. Farrell. Optimal communication-computation tradeoff for wireless multimedia sensor network lifetime maximization. In *Proceedings of IEEE Wireless Communications and Network Conferece*, 2009.
[24] D. Tulone and S. Madden. An energy-efficient querying framework in sensor networks for detecting node similarities. In *Proceedings of IEEE/ACM MSWiM*, 2006.
[25] A. Wang and A. Chandrakasan. Energy-efficient dsps for wireless sensor networks. *IEEE Signal Processing Magzine*, 7, 2002.
[26] Y. Xu and W.-C. Lee. On localized prediction for power efficient object tracking in sensor networks. In *Proceedings of IEEE ICDCS Workshop*, 2003.
[27] O. Younis and S. Fahmy. Heed: A hybrid energy-efficient distributed clustering for adhoc sensor networks. In *Proceedings of IEEE INFOCOM*, 2004.
[28] O. Younis, M. Krunz, and S. Ramasubramanian. Node clustering in wireless sensor networks: recent developments and deployment challenges. *IEEE Network*, 20(3):20–25, 2006.

Hongbo Jiang received the B.S. and M.S. degrees from Huazhong University of Science and Technology, China. He received his Ph.D. from Case Western Reserve University in 2008. After that he joined the faculty of Huazhong University of Science and Technology as an associate professor. His research concerns computer networking, especially algorithms and architectures for high-performance networks and wireless networks. He is a member of the IEEE.



Shudong Jin (M'00 / ACM'01) received his B.S. and M.S. from Huazhong University of Science and Technology, China, and received his Ph.D. from Boston University. He is an assistant professor in Computer Science at Case Western Reserve University. His research interests include network protocols and algorithms, network modeling and performance evaluation, multimedia streaming, and pervasive computing. He is a member of the IEEE.



Chonggang Wang received his PhD degree from Beijing University of Posts and Telecommunications (BUPT). He was awarded a National Award for Science and Technology Progress in Telecommunications. He is currently with NEC Laboratories America. His research focuses on Hybrid Optical and Wireless Networks, Sensor Networks and Applications, Cognitive Radio Networks, Ubiquitous and Distributed Computing, and Data Center. He is an editor of ACM/Springer Journal of Wireless Networks and an associate technical editor of IEEE Communications Magazine. He is a senior member of the IEEE.