An Energy-Efficient Data Collection Scheme for Wireless Sensor Networks

Zhidan Liu*†, Wei Xing *†, Yongchao Wang†, Dongming Lu*†
*College of Computer Science and Technology, Zhejiang University, Hangzhou, 310027, P. R. China
†Cyrus Tang Center for Sensor Materials and Applications, Zhejiang University, Hangzhou, 310027, P. R. China
Email: {danielliu,wxing,ychwang,ldm}@zju.edu.cn

***Abstract*—Keeping data collecting while preserving the scarce energy of sensor nodes is always one of the most crucial problems in wireless sensor networks. In this paper, we propose the DCS scheme to effectively exploit the ubiquitous temporal-spatial correlation in most natural phenomena for energy-efficient data collection of wireless sensor networks. Specifically, for temporal correlation, we build lightweight AR model locally to capture data distribution at sensor node; for spatial correlation, by making use of our novel definition of similarity measure between sensor nodes, we perform centralized model clustering, which is a kind of clustering that emphasizes data similarity between nodes but ignores geographical distance, to group sensor nodes with similar data distribution on both magnitude and trend into the same cluster. Then through scheduling sensor nodes to report readings alternately and performing dual-prediction at both sensor nodes and Sink, DCS acquires sensing readings without compromising too much data accuracy loss. Simulation results illustrate the efficiency of DCS scheme on a data set synthesized from real-world temperature data, i.e., 82.94% communication overhead reduction while keeping data error as low as $0.0456^{\circ}C$ when user-provided error-tolerance threshold sets as $0.2^{\circ}C$.**

***Keywords*— Energy Efficient, AR Model, Time Series Analysis, Temporal-Spatial, Data Collection, Wireless Sensor Network**

I. Introduction

Wireless sensor networks (WSNs) enable people to observe details of real-world phenomena in both temporal and spatial dimensions. Data collection is the fundamental function of WSNs, but also a challenging task due to limited resources of those tiny sensor nodes. Among all activities of a sensor node, it is well-known that data communication causes the maximum energy drain. Therefore, data collection scheme, which avoids abundant communication overhead yet keeps the data quality, becomes the effective method to achieve a longer network lifetime of WSNs for data-driven applications, which require sensor nodes to perform data sampling and transmit data to Sink periodically, such as environmental monitoring [1] [2].

To Conserve the finite resources, such as energy, network bandwidth and CPU usage, extensive research work has been done and various energy-saving protocols and algorithms have been proposed for these data-driven applications [3]. Among of these work, *model-driven data acquisition* has been proved to be an effective approach to reduce communication without compromising data quality, not only in theory [4] [5] [6] [7] [8] but also in practice [9]. BBQ [4] and Ken [5] approximate the data with user-specified confidence by keeping statistical model local and global in sync. However, both BBQ and Ken need amount of data to train an appropriate statistical model at expensive communication cost. Besides, these two frameworks are so complicated that related domain knowledge is needed. As typical time series data, sensing readings can be modeled and analyzed with methods of *time series analysis* [10]. L. Chong et al. [8] firstly apply the *ARIMA* model in energy-efficient data collection for WSNs. In their data collection scheme, Sink node builds suitable *ARIMA* model for each sensor node at first. During the adaptive data collection phase, both node and Sink perform forecasting for next sampling value with the same model, and Sink keeps the prediction value as sampling data if it does not receive the real value from sensor node, which sends the actual value only when the prediction error is beyond a pre-defined error-tolerance threshold. With models built by Sink, large amount of data communication are triggered. Compared to *ARIMA*, *AR* model is more lightweight but still offers competitive prediction accuracy. PAQ [7] and SAF [11] both adopt *AR* model to capture the underlying trend of data distribution. With dual-prediction at both node and Sink, redundant data communication are suppressed and energy is conserved. Furthermore, PAQ has proposed monitoring algorithm to maintain a local dynamic model to adapt to the changing phenomenon. Similar work relying on linear regression to perform data collecting are also presented in [6] [12] [13].

Though most of the work stated above have taken advantage of temporal correlation among consecutive samples of a sensor node, they have overlooked the ubiquitous spatial correlation among neighboring sensor nodes. As an alternative to model-based data collection, some solutions seek to identify some clusters, whose members share the similar data distribution, and schedule nodes to work alternately as representative nodes to obtain approximate data for other nodes in the same cluster. EEDC [14], for example, identifies this kind of clusters based on both magnitude and trend similarity with raw readings in a centralized manner. After clustering, sensor nodes work and report readings following the schedules drafted by Sink. With raw readings collected from all sensor nodes to Sink for clustering, it is inevitable that EEDC will produce considerable communication overhead. SAF [11] also pays attention to spatial correlation, and proposes *model clustering* to distinguish similar nodes. Unlike traditional clustering in WSNs, model clustering emphasizes the data similarity between nodes but

Corresponding author.

ignores the real geographic distance between them. Moreover, model clustering allows sensor nodes to be unaware of their membership and cluster variations as Sink will process all of these issues transparently. However, SAF identifies model clusters based on the predicted values at Sink, which could reduce data communication between sensor nodes and Sink yet obviously reduce the accuracy of acquired data.

In this paper, inspired by ideas of dual-prediction and model clustering in SAF [11], we propose an energy-efficient data collection scheme (referred as **DCS**) to perform long-term data acquisition without losing too much data accuracy. Specifically, each sensor node builds and transmits its *AR* model to Sink firstly. With help of our novel definition on similarity measure, we perform model clustering to group nodes with similar data distribution and variation trend into the same cluster. It is worthy to note that model clustering in DCS is based on the *AR* models but not the predicted values. Subsequently, sensor nodes in the same cluster are arranged to report sensing readings alternately. Furthermore, schedule making and dynamic model cluster maintaining strategies are also designed. Simulations on synthetic data set prove the efficiency of DCS scheme, namely that more energy can be preserved while incurring negligible error shown as results.

The rest of this paper is organized as follows. In Section II, we present some preliminaries about our DCS scheme, including a brief introduction to *AR* model, system model and similarity measure method. The proposed DCS scheme is elaborated in Section III, followed with performance evaluation in Section IV. Finally, we draw conclusions in Section V.

II. Preliminaries

A. AutoRegressive Model

In general, a time series data is a set of observations x_t, each one being recorded at a specific time t. An important objective of time series analysis is the description of a suitable uncertainty mode for these data. The AutoRegressive Integrated Moving Average (*ARIMA*) model belongs to such kind of models and has been widely used for univariate time series. Obviously, readings of sensor node belongs to such kind of time series data, and the *ARIMA* model can be used for data forecasting in WSNs. Compared to *ARIMA* model, the *AR* model is much simpler and more lightweight but still offer excellent accuracy in WSN-based applications [7]. An *AR* model with order p, which is the number of lagged values in a linear regression, is usually denoted as $AR(p)$, and expressed as

$$x_t = c + \sum_{i=1}^{p} \alpha_i x_{t-i} + \epsilon_t \tag{1}$$

Where $\vec{\alpha} = \{\alpha_1, ..., \alpha_p\}$ are the coefficients of *AR* model, c is a constant but always omitted for simplicity, and ϵ_t is the *White Noise*. The calibration of *AR* model is quite simple, and various methods can be adopted, such as Burg, Yule-Walker, ordinary least-square, or maximum likelihood estimation. More details of *AR* model and time series analysis are referred to [10].

B. System Model

In this paper, we consider a sensor network consisting of a collection $S = \{s_1, s_2, ..., s_N\}$ of N sensor nodes and one Sink node. All sensor nodes are uniformly and randomly distributed in a size of $L \times L$ sensing field. The Sink node locates outside of the field but not far away. Each sensor node has the identical communication radius R, and communicates with distant node hop-by-hop. Periodically, sensor nodes generate environmental sensing readings which evolves over time, and transmit these readings to Sink via *data collection tree* (DCT). In *multi-hop* scenario, DCT is indispensable and can be easily built in a distributed manner. For instance, by circulating a tree formation message originated by Sink and making use of a min-hop parent selection strategy, or other algorithms used for constructing maximum-lifetime data gathering tree [15].

Furthermore, each sensor node maintains a local queue Q_l to store the most recent W sensing readings and keeps tracking of the average value μ of those W readings. By exploiting these data in queue, each sensor node can learn an *AR* model using least-square or other methods. Similar to PAQ [7], we adopt a narrow prediction window, such as $p = 3$, to neglect the impact of non-stationary physical environment, and also to make the *AR* model simple and lightweight enough for resource-constrained sensor nodes. To cope with dynamic nature of environment and keep accuracy of *AR* models, we adopt the monitoring algorithm of PAQ to maintain a dynamic local *AR* model for each node, namely, sensor node re-learns its *AR* model using the latest W readings in queue Q_l if the number of times the prediction error beyond error threshold θ within consecutive Λ epochs exceeds the pre-defined threshold ν. Besides, for dual-prediction case, Sink keeps tracking the latest *AR* model of each sensor node, and maintains a smaller size of global queue Q_g to store the latest p historical values for each sensor node.

C. Similarity Measure

To perform model clustering effectively, in this subsection we will introduce our novel similarity measure method, which is in accord with the context of model-driven data collection. Unlike methods presented in EEDC [14] or SAF [11], we argue that similarity measure should not only consider the magnitude similarity on raw time series data, but also take the underlying trend of time series into account. Capturing the relation between latest historical data and recent future data, *AR* model built at each sensor node could be a good structure to reflect the trend of time series data in that region. In addition, average value μ maintained at each sensor node is an ideal baseline to represent the magnitude situation of the monitored region. Consequently, it is reasonable and feasible to combine their *AR* models and corresponding average values μ to measure the similarity of two sensor nodes. As typical linear systems, correlation between two *AR* modes can be well measured with *Pearson Correlation Coefficient*. Formally, assume two sensor nodes s_i and s_j with their *AR*(3) model coefficients $\vec{X} = \{x_1, x_2, x_3\}$ and $\vec{Y} = \{y_1, y_2, y_3\}$ respectively, *trend similarity* between them can be calculated as

$$\rho_{s_i,s_j} = \frac{\mathrm{cov}(\vec{X},\vec{Y})}{\sigma_{\vec{X}}\sigma_{\vec{Y}}} = \frac{E((\vec{X}-\mu_{\vec{X}})(\vec{Y}-\mu_{\vec{Y}}))}{\sigma_{\vec{X}}\sigma_{\vec{Y}}}. \quad (2)$$

Moreover, *Manhattan distance* could be used to detect the similarity between their baseline values, namely, if their *magnitude similarity* $M_{s_i,s_j} = \left|\mu_{s_i} - \mu_{s_j}\right| \leq \frac{\varepsilon}{2}$ then these two nodes are considered magnitude similar, where ε is a user-defined parameter to bound the maximum difference of readings of any two nodes in the same cluster. Therefore, we have following definition to estimate whether two nodes are similar.

Definition: *similar nodes.* two sensor nodes, s_i and s_j, are similar nodes and can be grouped into the same model cluster if their trend similarity $\rho_{s_i,s_j} \geq cth$ and magnitude similarity $M_{s_i,s_j} \leq \frac{\varepsilon}{2}$, where both cth and ε are user-defined parameters to guide model clustering.

III. The Data Collection Scheme

A. Overview

By exploiting the temporal and spatial correlation in WSNs, our scheme, i.e., DCS, aims to suppress data communication through making tradeoff between tiny data accuracy loss and large energy saving. Keeping *AR* models between sensor nodes and Sink node in sync, DCS firstly groups nodes into clusters based on their data similarities, and then alternately schedules nodes to act as representative nodes, which report readings to Sink periodically. On the other side, Sink restores approximate data for each node to feed its queue Q_g respectively based on the collected real values. Meanwhile, Sink adjusts all model clusters timely once variation of data distribution is detected by sensor node or variation of cluster similarity is detected by Sink. In general, DCS includes three major phases:

1) ***Local model learning phase****:* To avoid transmitting vast of raw sensing readings to Sink node to build probabilistic model for each sensor node, we prefer to learn and maintain the *AR* model locally at each sensor node. After accumulating enough data, i.e., W data to feed the queue Q_l full, each node estimates the coefficients of *AR*(3) via least-square regression method. Notice that, other parameter estimation methods for *AR* model, such as maximum likelihood, could be used, but least-square regression could be the best fit method as it is simple enough to avoid complicated computation.

2) ***Centralized model clustering phase****:* Once completing the local *AR* model learning phase, each sensor node transmits the model coefficients $\vec{\alpha}$ and average value μ of the W data in Q_l to Sink via DCT. With these information, Sink performs model clustering in a greedy manner to obtain as few clusters as possible. Details of model clustering are presented in III-B.

3) ***Approximate data collection phase****:* In this phase, Sink establishes working schedules for each model cluster. Instead of reporting readings to Sink by all sensor nodes, *only* some nodes in each model cluster are appointed as representative nodes to report data periodically. To balance energy consumption, working schedule of each model cluster is re-designed every W epoches. With the acquired data from representatives, Sink restores approximate data for other *unworking* nodes. Moreover, depending on these real values and their corresponding predicted values via *AR* models at Sink, Sink node could detect potential dissimilarity among cluster members, which will trigger the model cluster maintaining procedure at Sink. Besides, sensor nodes which detect variation of data distribution, such as *AR* model updating or drastic change of average value μ, could also trigger the model cluster maintaining procedure. More details are described at III-C.

Algorithm 1: centralized model clustering

```
1  Label nodes unclustered: L(s_i)=false, i=1→N;
2  Compute similar nodes set Γ_{s_i}, i=1→N;
3  Descending sort nodes S according to cardinality |Γ|;
4  for i=1 to N do
       /* s_i is the node with the i-th
          largest cardinality in S        */
5      if !L(s_i) then
6          L(s_i)=true;
7          C_{s_i}={s_i};
8          foreach s_j in Γ_{s_i} do
9              if !L(s_j) then
10                 flag=true;
11                 foreach s_k in C_{s_i} do
12                     if s_j is dissimilar with s_k then
13                         flag=false;
14                         break;
15                 if flag then
16                     L(s_j)=true;
17                     C_{s_i}=C_{s_i} ⋃{s_j};
18 Output all m model clusters C, ⋃_{i=1}^{m} C_i = S
```

B. Centralized Model Clustering

With user-defined model clustering parameters cth and ε, Sink node computes similar nodes set Γ for each sensor node firstly. For example, if node s_i and s_j are similar nodes, then s_j is included in the similar nodes set Γ_{s_i} of s_i. Obviously, relation of similar nodes is symmetric, namely s_i is included in set Γ_{s_j} too. To produce the minimum number of clusters, we adopt a greedy algorithm to finish the centralized model clustering, as shown in **Algorithm 1**. At first, we sort all nodes in descending order according to their cardinalities of similar nodes sets. Heuristically, node with larger similar nodes set could form cluster with more members, resulting in fewer clusters to cover all nodes. Therefore, our greedy model clustering starts from sensor node with the largest cardinality, and iteratively its similar nodes, which are similar with all nodes already in the cluster, are added in to expend current cluster. This process is repeated until all nodes are covered. It is necessary to remind that model clustering does not take the node location into account.

C. Approximate Data Collection

Actually, approximate data collection phase of DCS combines the concepts of representative node and dual-prediction, which make use of ubiquitous spatial correlation and temporal correlation in WSNs respectively. Generally, there are three primary tasks in this phase, namely, schedule making, data acquisition and dynamic model cluster maintenance.

schedule making: During data collection phase, at least one node in each model cluster is scheduled to report sensing readings to Sink. These working nodes are called *representative nodes* of their clusters respectively. Specifically, each node is assigned with a different probability λ to be selected randomly as representative node. Given the size of model cluster, e.g. κ, and hop distance, e.g. h_i, between node s_i and Sink, λ could be computed as $\lambda_i = \frac{1}{\kappa} \times \frac{(h_{max}+1-h_i)^2}{(h_{max}+1)^2}$, where hop distances can be obtained from the messages transmitted by all sensor nodes, and h_{max} is the maximum hop distance in the whole network. Once completing the schedule making, Sink will transmit the schedules to all nodes. To balance energy consumption among all nodes, Sink should make another random schedule for each model cluster every W epoches.

data acquisition: Obviously, Sink can get the real values of representative nodes during their working period. Besides, Sink can also obtain piggyback real values of specific sensor nodes which will transmit notifications about variation of data distribution to Sink for dynamic model cluster maintenance at some epoches. Like conventional dual-prediction based data collection scheme, sensor nodes which have great prediction error will also send their real values to Sink to replace current prediction values. For sensor nodes of these cases mentioned above, Sink feeds queues Q_g with their real values respectively. By keeping *AR* models at Sink in accord with local *AR* models at all sensor nodes, DCS acquires data for sensor nodes which have no real values of current epoch via prediction. However, we feed the queues Q_g of this kind of nodes in the same model cluster with the same restored data, i.e., $\frac{max_t+min_t}{2}$, where max_t and min_t denote the maximum and minimum values of the collected real values in the same cluster at epoch t. We adopt this rule to restore data as it is considered that the restoration error could be bounded [14]. Remember that *AR* models at Sink predict data based on these data stored in queues Q_g.

dynamic model cluster maintenance: Both variations of data distribution and cluster similarity will trigger the procedure of dynamic model cluster maintenance. For the former case, when a sensor node re-learns its *AR* model as indication by monitoring algorithm of PAQ [7] or detects drastic change on its average value μ of the latest W data in queue Q_l, i.e., $|\mu_i - \mu_i{}'| > \varepsilon$ where μ_i is the current average value and $\mu_i{}'$ is the value used in the most recent model clustering, this sensor node, e.g. s_i, should notify Sink that an inspect of cluster membership of s_i is necessary. Once notified, Sink will check whether s_i is still similar with all other nodes in the model cluster. If not, Sink merges s_i with other model cluster whose members are all similar with s_i. In the worst case, node s_i would form a new model cluster solely. Notice that there is no distance constraint when finding another similar cluster for s_i. For the latter case, Sink will check the similarity situation of all model clusters at every epoch by exploiting the collected real values and the predicted values of working nodes. Formally, let's assume that the working nodes set of a model cluster is $S_W = \{w_1, w_2, ..., w_n\}$ with their real values $V_t = \{v_1, v_2, ..., v_n\}$ and predicted values $V_t{}' = \{v_1{}', v_2{}', ..., v_n{}'\}$ respectively at epoch t. If the average absolute prediction error $\bar{e} = \frac{\sum_{i=1}^{n} |v_i - v_i{}'|}{n} > \omega \cdot \varepsilon$, where ω is a system parameter to adjust the sensitivity of cluster similarity detection. If prediction error $\bar{e}$ of a model cluster exceeds the defined threshold, namely $\omega \cdot \varepsilon$, Sink will split this cluster into serval clusters with similar method like **Algorithm 1**. If necessary, re-clustering of all nodes could be performed when the total cluster number is beyond certain threshold, i.e., $Max\_Clusters$. Lastly notice that after cluster maintaining, Sink should check the schedules of all model clusters to ensure that each cluster has at least one sensor node in working status.

IV. Performance Evaluation

A. Simulation Setup

To study and evaluate our DCS scheme, we have performed simulation study with *Matlab*. In our simulations, $N = 100$ sensor nodes are deployed in a size of $100m \times 100m$ sensing field, and Sink node locates at $(120m, 50m)$. Communication radius R sets to be $30m$ for all sensor nodes. For the monitoring algorithm to maintain dynamic local *AR* models, we employ similar parameters setting as PAQ [7], i.e., $\theta = 0.03$, $\Lambda = 15$, and $\nu = 8$. Besides, we set $Max\_Clusters = 30$, namely re-clustering is needed when cluster number is greater than 30. Regarding to cluster similarity detection, we adjust value of system parameter ω to ensure $\omega \cdot \varepsilon = 0.1$ with various setting of user-defined ε. We argue that data error with 0.1 is sufficient to make an decision on variation of cluster similarity.

For our simulation study, we generate synthetic data set with following method: 25 event sources are fixed in the sensing field in uniform distribution, readings of each sensor node are comprehensive influencing results of all event sources, and the influence of an event source, say e_w, to sensor node s_i is inversely proportional to the geographic distance between them. To simulate the evolutive values of event sources, we use the publicly available Intel Lab dataset [16] which consists of 54 sensor nodes to measure various attributes, such as temperature, humidity, light and voltage. Owning to some data missing, We pick out and restore the temperature values of 51 nodes on March 9, 2004. The restored data series of each sensor node contains 1060 values which correspond with 1060 epoches. At the beginning of each simulation, we randomly select 25 data series from the selected 51 sensor nodes to map to the 25 event sources. Lastly, notice that all results in this section are the average values of 10 simulations.

B. Results on Model Clustering

There are serval key parameters in our DCS scheme, such as ε, cth and W. In this subsection, we study the impacts of

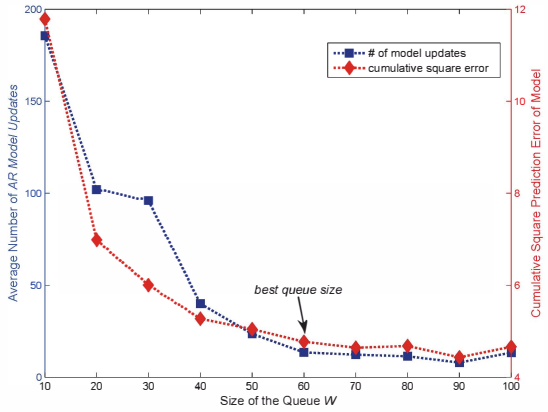

Fig. 1. Impact of W on AR model

these parameters on both model clustering and data collection with simulations. By varying W from 10 to 100, firstly we study the impacts of the size of queue Q_l maintained at each sensor node on model learning and data collection. Generally, a larger queue Q_l means that there are more data to train a better AR model. However, it means more computations are needed as well. As shown in Fig.1, we find that larger queue could not always lead to the best choice. When below 60, a smaller W will result in more model updates, which incur more data communication between sensor nodes and Sink to keep models in sync. At the same time, less accurate AR model will generate larger prediction error, just as indicated by the metric of cumulative square prediction error. On the other side, increasing the size of queue to larger than 60 will not bring too much benefit but more computations, shown as when $W > 60$. As a result, $W = 60$ could be the best choice for our $AR(3)$ model, and we employ $W = 60$ for the rest of simulations.

In DCS scheme, both ε and cth are the user-defined parameters to guide model clustering, meanwhile these two parameters will also affect the accuracy of data collection. In Fig.2 with cth fixed as 0.9, we observe that with increase of ε, the number of formed model clusters drops rapidly. It is easy to understand that when user-provided error-tolerance threshold becomes larger, sensor nodes are easy to be grouped into the same cluster, resulting in fewer clusters needed to cover all nodes. For the similar reasons, if we have a stronger requirement on correlation between two AR models, nodes are more difficult to meet the requirements of similar nodes. Just as reported in Fig.3 with ε fixed as 0.2, more clusters are needed to cover all nodes when correlation requirement becomes more rigorous, namely cth becomes larger. Actually, correlation of two AR models with $cth = 0.9$ could be considered as highly correlated. As stated before, model clustering does not take geographic distance into account, so the number of clusters are not very great no matter what settings of ε and cth are. Besides, we also carry out simulations to study the impact of ε on data collection. As illustrated in Fig.4, with the increase of ε and a fixed value of $cth = 0.9$, the messages generated for data collection decrease dramatically, while the average absolute error rises at the same time. Results in Fig.4 also

TABLE I
SUMMARY INFORMATION FOR VARIOUS SCHEMES

metric	PureDC	PureAR	DCS
number of messages	356160	73013	60771
average absolute error	/	55.58	45.59

demonstrate our statement on tradeoff between data accuracy and energy consumption, i.e., the improvement on accuracy of collected data builds on expensive cost of energy consumption (here we substitute messages for energy consumption as data communication is the dominating energy consumer).

C. Results on Energy Efficiency

To prove the energy efficiency of DCS scheme, in this subsection we will perform compared experiments with other two alternative data collection scheme, namely, PureDC and PureAR. In PureDC scheme, all sensor nodes report their real values to Sink via DCT periodically. In PureAR scheme, only dual-prediction mechanism similar with methods in [7] [8] [12] is adopted, i.e., both sensor node and Sink perform forecasting with the same AR model, and sensor node transmits the real value only when the prediction error is beyond defined threshold, while Sink node keeps the predicted values as sampling readings when there are no real values received. In this compared experiments, we set $cth = 0.9$ and $\varepsilon = 0.2$ for DCS, other parameters are the same as in the IV-A. Summary results are presented in Table I, thanks to the efficiency of AR model, PureAR reduces 79.50% messages, and our scheme cuts down 82.94% data communication further. On the other side, though less data communication generated between sensor nodes and Sink, DCS still improves the data accuracy with 17.97% than PureAR. With DCS scheme, taking the starting $W = 60$ epoches for local AR model learning phase off from the total 1060 epoches, there are only approximate $\frac{45.59}{1060-60} \approx 0.0456°C$ error between the acquired data at Sink and the real values. Obviously, by effectively exploiting the temporal-spatial correlation in WSNs in the forms of dual-prediction and model clustering, DCS can save large amount of energy without compromising too much data accuracy loss.

V. CONCLUSION

In this paper, we propose an energy-efficient data collection scheme named DCS, for WSNs to reduce communication overhead yet keep data acquisition without too much accuracy loss. Taking advantages of lightweight AR model and novel concept of model clustering inspired by SAF [11], DCS performs data collection by perfectly exploiting temporal-spatial correlation in WSNs. Simulation results illustrate the efficiency of our model clustering algorithm and data collection scheme. Specifically, DCS can reduce 82.94% communication overhead while incurring approximate $0.0456°C$ error when user-defined error-tolerance threshold ε sets as $0.2°C$, which are much better than previous dual-prediction based data collection schemes on both communication overhead reducing and accuracy retaining.

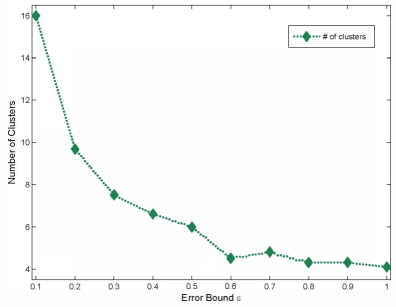

Fig. 2. Impact of ε on model clustering

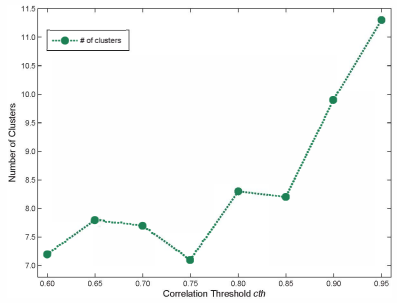

Fig. 3. Impact of cth on model clustering

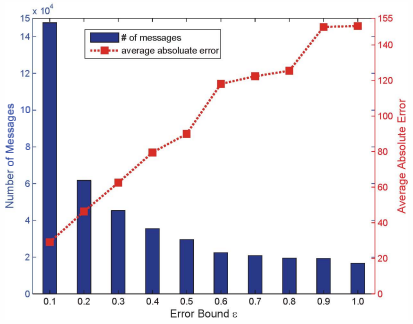

Fig. 4. Impact of ε on approximate data collection

ACKNOWLEDGMENT

This work is partially supported by Scientific and Technical Innovation Team Project of Zhejiang Province for Digital Culture and Multimedia Technology (2010R50040).

REFERENCES

[1] P. Corke, T. Wark, R. Jurdak, H. Wen, P. Valencia, and D. Moore, "Environmental Wireless Sensor Networks," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1903–1917, 2010.

[2] L. Yunhao, H. Yuan, L. Mo, W. Jiliang, L. Kebin, M. Lufeng, D. Wei, Y. Zheng, X. Min, Z. Jizhong, and L. Xiang-Yang, "Does Wireless Sensor Network Scale? A Measurement Study on GreenOrbs," in *IEEE INFOCOM*, 2011.

[3] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy Conservation in Wireless Sensor Networks: A Survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009.

[4] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," in *Proceedings of the 30th International Conference on Very large Data Bases (VLDB)*, 2004.

[5] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks using Probabilistic Models," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, 2006.

[6] S. Santini and K. Romer, "An Adaptive Strategy for Quality-based Data Reduction in Wireless Sensor Networks," in *Proceedings of the 3rd International Conference on Networked Sensing Systems (INSS)*, 2006.

[7] D. Tulone and S. Madden, "PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks," in *European Conference on Wireless Sensor Networks (EWSN)*, 2006.

[8] L. Chong, W. Kui, and T. Min, "Energy Efficient Information Collection with the ARIMA Model in Wireless Sensor Networks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2005.

[9] U. Raza, A. Camerra, A. Murphy, T. Palpanas, and G. Picco, "What Does Model-Driven Data Acquisition *Really* Achieve in Wireless Sensor Networks?" in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2012.

[10] J. Peter and A. Richard, *Introduction to Time Series and Forecasting, 2nd Edition*. Springer, 2002.

[11] D. Tulone and S. Madden, "An Energy-Efficient Querying Framework in Sensor Networks for Detecting Node Similarities," in *Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2006.

[12] A. Ghaddar, T. Razafindralambo, I. Simplot-Ryl, D. Simplot-Ryl, and S. Tawbi, "Towards Energy-Efficient Algorithm-based Estimation in Wireless Sensor Networks," in *The 6th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2010.

[13] L. Jialiang, F. Valois, M. Dohler, and W. Min-You, "Optimized Data Aggregation in WSNs Using Adaptive ARMA," in *The 4th International Conference on Sensor Technologies and Applications (SENSORCOMM)*, 2010.

[14] L. Chong, W. Kui, and P. Jian, "An Energy-Efficient Data Collection Framework for Wireless Sensor Networks by Exploiting Spatiotemporal Correlation," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, no. 7, pp. 1010–1023, 2007.

[15] L. Junbin, W. Jianxin, C. Jiannong, C. Jianer, and L. Mingming, "An Efficient Algorithm for Constructing Maximum Lifetime Tree for Data Gathering Without Aggregation in Wireless Sensor Networks," in *IEEE INFOCOM*, 2010.

[16] "Intel Lab dataset," http://db.csail.mit.edu/labdata/labdata.html.

Zhidan Liu received the B.E. degree in Computer Science and Technology from Northeastern University, Shenyang, China, in 2009. He is currently a PhD student in College of Computer Science and Technology at Zhejiang University, Hangzhou, China. His research interests include data management and wireless communication in sensor networks.

Wei Xing received the B.E., M.E. and Ph.D. degrees from Zhejiang University, Hangzhou, China in 1989, 1992, and 2009, respectively. He joined Department of Control in College of Information Technology, Zhejiang University, in 1992. Since 2002, he has been with the College of Computer Science and Technology in Zhejiang University, where he is currently an Associate Professor. His research interests cover computer network, multimedia technology, Internet of Things.

Yongchao Wang was born in China in 1975. He received the B.E. degree from Zhejiang University of Technology and M.E. degree from Zhejiang University, China, in 1997 and 2004, respectively. He joined Wireless and Next Generation Network Corporation, Hangzhou, China in 2005, where he is currently a Senior Engineer. His main areas of research interest are IPv6 network and Internet of Things. Mr. Wang is a member of the Innovation Alliance of the Internet of Things in Zhejiang, and the Computer Association of China in Zhejiang.

Dongming Lu received the B.E., M.E. and Ph.D. degrees from Zhejiang University, China, in 1989, 1991 and 1994, respectively. He is currently a Professor in the College of Computer Science and Technology at Zhejiang University. His research interests include Internet of Things, multimedia technology, digital protection of cultural relic, virtual reality and digital museum. Prof. Lu is the Director of Professional Committee of Network Technology in Zhejiang, and the Standing Director of the Computer Association of China in Zhejiang.