Online Information Compression in Sensor Networks

Song Lin, Vana Kalogeraki, Dimitrios Gunopulos, Stefano Lonardi
Computer Science & Engineering Department
University of California, Riverside
{slin, vana, dg, stelo}@cs.ucr.edu

***Abstract*-In the emerging area of wireless sensor networks, one of the most typical challenges is to retrieve historical information from the sensor nodes. Due to the resource limitation of sensor nodes (processing, memory, bandwidth, and energy), the collected information of sensor nodes has to be compressed quickly and precisely for transmission. In this paper, we propose a new technique -- the ALVQ (Adoptive Learning Vector Quantization) algorithm to compress this historical information. The ALVQ algorithm constructs a codebook to capture the prominent features of the data and with these features all the other data can be piece-wise encoded for compression. In addition, with two-level regression of the codebook's update, ALVQ algorithm saves the data transfer bandwidth and improves the compression precision further. Finally, we consider the problem of transmitting data in a sensor network while maximizing the precision. We show how we apply our algorithm so that a set of sensors can dynamically share a wireless communication channel.**

I. INTRODUCTION

The ever changing developments in electrical embedded systems have enabled the widespread deployment of sensor networks consisting of small sensor nodes with sensing, computation, and communication capabilities. The sensor nodes can monitor various characteristics of the environment such as temperature, humidity, pressure, light, sound, chemicals, noise levels, radioactivity, movement, etc. The applications of sensor networks have been seen in a large variety of areas. For example, the sensor networks can help biologists automatically recognize and track different species of birds. The environmental scientists can utilize sensor networks to monitor and record the development of environmental conditions. In the intelligent building, sensors are deployed in offices and hallways to measure temperature, noise, light, and interact with the building control system. In the battle fields, soldiers equipped with sensor can be easily tracked and organized by commander.

Suppose there is a sensor network with a base station (sink) and *N* sensors. An interesting application is to let the sink collect the historical information from each sensor to perform some query processing or statistical analysis. Several constraints of sensor networks make the information retrieval implementation difficult. The first constraint is the resource (i.e. memory, bandwidth and power, etc) limitation of sensor node. Thus it is impractical to transmit the original data feed from each sensor to the base station. The second constraint is communication channel sharing. When transmitting data to the sink, several sensors have to share the communication channel. In this paper, we address the historical information retrieval problem by addressing the following sub-problems: a) At each sensor, how to compress its historical information given a bandwidth allocation? b) In the sensor network, how to allocate bandwidth to sensors in order to maximize and balance their compression qualities?

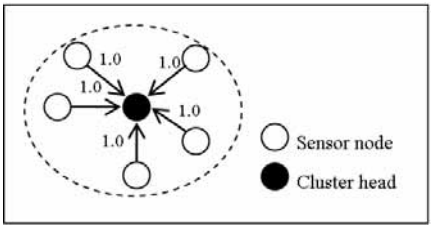

Figure 1. The *LEACH* model

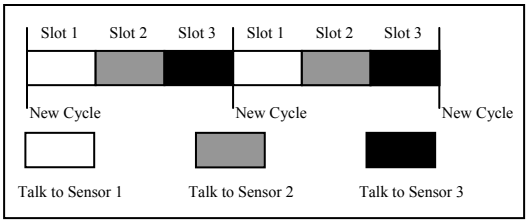

Figure 2. A sample communication channel

An efficient data compression technique SBR (Self Based Regression) [6] has been proposed recently and has been shown to work better than other well-known techniques in sensor network settings. A problem of SBR is its codebook is not precise enough for compression. In addition the codebook updating in SBR wastes a lot communication bandwidth. As to the network topology, a large amount of energy-saving algorithms [7][10][11] group nearby sensors into clusters. All the sensors in the same cluster share the cluster head's communication channel evenly (as in Figure 1), that is, they utilize the same data transmission bandwidth if they want to talk to the cluster head. For example, in TDMA scheduling, a different time slot is assigned to each sensor (as in Figure 2), while in FDMA scheduling, a different frequency range slot is allocated to each sensor. For our compression problem, as the information is distributed differently at different sensors, the compression qualities show large variances among sensors given the same bandwidth allocation. As the bandwidth is an important factor for increasing compression qualities, if we could assign more bandwidth to the low quality sensors and assign less band-width to the sensors whose compression qualities are high enough, the overall compression qualities of all sensors would be maximized and balanced. In this paper we address this problem and present a Dynamic Bandwidth Assignment algorithm to solve it.

A. Our Contributions

Our contributions are summarized as follows:

a) We apply the LVQ (Learning Vector Quantization) algorithm to construct the codebook for data compression. Our results show that the LVQ learning process can further improve the codebook for high compression precision.

b) We introduce the concept of two-level regression for higher precision compression. The two-level regression is applied to the codebook update in order to save more bandwidth while keeping the codebook updated with high precision.

c) We consider the problem of dynamic allocation of bandwidth in wireless sensor networks. We present a new algorithm, DBA (Dynamic Bandwidth Allocation), which works in combination with our compression algorithm, and dynamically allocates bandwidth among channel sharing sensors. The algorithm uses the recent history to predict future bandwidth requirements and balance the expected loss for the different sensors.

The rest of this paper is organized as follows. Section II states the background and definition of our problem. In Section III we present the related work and we describe our ALVQ algorithm in Section IV. Dynamic bandwidth allocation problem is addressed in section V. In Section VI we provide our experimental results. Finally, we conclude our remarks and future work in Section VII.

II. Problem Definition

A sensor node S is equipped with a measuring system which generates a data record $r = (t, val1, val2,\ldots)$ every ε seconds, where t is the timestamp on which the record was generated, and $(val1, val2,\ldots)$ are the measurements at that time instance. The sensor has its local data buffer $\boldsymbol{B}$ which stores these records. When $\boldsymbol{B}$ is full, the sensor compresses the data and transfers the compression representative to the sink. In this paper, we address the following two problems:

Problem I: Given a one dimensional time series data $\boldsymbol{X}$ collected by a sensor, the goal is to find a proper encoder function F making $\boldsymbol{Y} = F(\boldsymbol{X})$ and a decoder function G, so that

a) $| \boldsymbol{X} | / | \boldsymbol{Y} | \geq R$ and b) $\| \boldsymbol{X} - G(\boldsymbol{Y}) \|$ is minimized.

In a) R is the compressing rate determined by to application specifications. In b) the distance between the retrieval values from the compressed information and the original values is minimized.

Problem II: Given a sensor network cluster $\boldsymbol{C}$ with k sensors, given the historical compression and transmission statistics in the cluster, the goal is to dynamically allocate different bandwidth to different sensors, such that the overall compression qualities of all k sensors in the cluster are maximized and balanced.

III. Dictionary Lookup Schemes

In high rate lossy compressions, Dictionary Lookup schemes [4] are widely used in graphics, pattern recognition, etc. In such schemes, there is a codebook (or base signal) that captures the prominent patterns of the data. For each data piece to be compressed, we "look up" the codebook, find the best approximation pattern and then use the approximation parameters to represent the original data piece.

The characteristics of sensor data make the Dictionary Lookup Scheme very appealing. Firstly, the data in sensor networks is collected from the environment and therefore is likely to show similar patterns over time. Secondly, some sensor nodes collect different measurements at the same time. These measurements show intrinsic correlation between each other, as is the case between pressure and humidity in weather monitoring systems.

Therefore, Dictionary Lookup Scheme is a good choice for historical information compression in sensor networks.

A. Piece-wise Approximation

In sensor networks, many physical quantities are correlated, like air temperature, pressure etc. Therefore we can use piece-wise linear regression to capture those properties.

1) Piece-wise Linear Regression: Given two time series data pieces, $\boldsymbol{X}$ and $\boldsymbol{Y}$, we use $\boldsymbol{X}$ to approximate $\boldsymbol{Y}$ by piece-wise linear regression, that is $\boldsymbol{Y'} = a*\boldsymbol{X} + b$ so that the regression error $\| \boldsymbol{Y} - \boldsymbol{Y'} \|$ is minimized. In sensor networks, many quantities show strong linear relationships between each other and the quantities themselves show similar patterns over time. Therefore if we choose the most prominent patterns, we can piece-wise approximate other patterns with high precision.

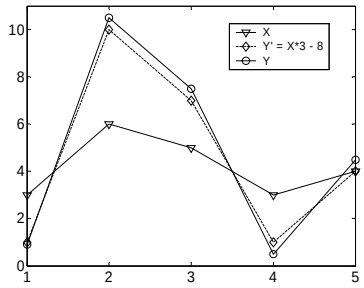

Figure 3. Piece-wise approximation

In Figure 3, there are two time series $\boldsymbol{X}$ and $\boldsymbol{Y}$. If we let $\boldsymbol{Y'} = 3\boldsymbol{X} - 8$, then $\boldsymbol{Y'}$ is a good approximation of $\boldsymbol{Y}$.

2) 2-level Piece-wise Linear Regression: Given three vectors of same size, $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$, we could use linear combination of $\boldsymbol{X}$ and $\boldsymbol{Y}$ to approximate $\boldsymbol{Z}$:

1. We piece-wise approximate $\boldsymbol{Z}$ by $\boldsymbol{X}$, that is $\boldsymbol{Z'} = a*\boldsymbol{X} + b$ so that the regression error $\| \boldsymbol{Z} - \boldsymbol{Z'} \|$ is minimized.
2. We construct a new vector $\boldsymbol{D}$ to store the difference of Vector $\boldsymbol{Z}$ and the approximation vector $\boldsymbol{Z'}$, $\boldsymbol{D} = \boldsymbol{Z} - \boldsymbol{Z'}$.
3. Employ $\boldsymbol{Y}$ to approximate the difference vector $\boldsymbol{D}$ we obtained above, that is $\boldsymbol{D'} = c*\boldsymbol{Y} + d$, so that the regression error $\| \boldsymbol{D} - \boldsymbol{D'} \|$ is minimized.
4. The compression parameters a, b, c, d are transmitted as the compression representative of $\boldsymbol{Z}$.

It should be noted that *2-level Piece-wise Linear Regression* is at least as good as standard *Piece-wise Linear Regression*.

Usually it is much more precise than its 1-level counterpart. It could be utilized in applications where higher compression precision is required.

B. Previous Work: LVQ (Learning Vector Quantization)

LVQ algorithm [8] is used in many applications such as image and voice compression, voice recognition, etc.

Assume there is a training data set with n vectors in k classes. Each vector has the class label indicating which class it belongs to. We want to construct a set of representative codeword vectors (called codebook) to classify the data in real world. Each new vector to be classified finds the nearest codeword vector in the codebook and is assigned the same class as that codebook vector.

Unfortunately designing a codebook that best represents the set of training vectors is NP-hard [14]. We therefore resort to a suboptimal codebook design scheme − LVQ (Learning Vector Quantization). First, we randomly select a training vector from each class. We denote theses vectors as $m_{1,\dots}m_k$ and take them as initial codebook. For every other training vector x we perform the following learning process:

for $i = 1, \dots k$,

$m_i = m_i + a(t)[x - m_i]$ *if x and* m_i *are in the same class;*

$m_i = m_i - a(t)[x - m_i]$ *if x and* m_i *are in different classes;*

Here $0<a(t)<1$, and $a(t)$ may be constant or decrease monotonically with time.

LVQ continuously adjusts its codebook with all the training data, so that the codebook is trained nearer to the optimal representative of each class.

C. Previous Work: SBR (Self Based Regression)

Because the historical information in sensor networks shows similar patterns over time and different measurements show natural correlation between each other, the SBR (Self Based Regression) algorithm was recently proposed in [6] to exploit these characteristics of the data. The authors show it outperforms other standard approximation techniques such as DCT, Wavelets and Histograms in precision. The SBR algorithm extracts the prominent feature from the training data to construct the codebook and the base station keeps a codebook for each sensor. When the data buffer at some sensor node is full, the collected data is compressed by piece-wise regression and the update codebook data piece is calculated. These results are then transmitted to the base station where the approximation of sensor data is retrieved.

Compared with SBR, our ALVQ algorithm increases the compression precision by improve the codebook accuracy. In addition, ALVQ compresses the update of codebook too, which saves bandwidth for data transmission and increases the quality of the approximation.

IV. The ALVQ Framework

We now present the *Adoptive LVQ* (ALVQ) algorithm for compressing historical information in sensor networks.

The ALVQ algorithm receives as input the latest n (size of data buffer in sensor node) data values, a bandwidth constraint *TotalBand* (number of values to transmit, including any codebook update values), the maximum size of the codebook M_{code} and the current codebook C_{base} of size $|C_{base}| < M_{code}$.

Our ALVQ algorithm works in the following ways: First, in the codebook construction, ALVQ performs a LVQ (Learning Vector Quantization) learning process on the codebook, which adjusts the codebook, to be nearer to the optimal codebook. Second, for codebook updates, ALVQ compresses the codebook update data pieces and transfers the compressed information to the base station. Using 2-level piece-wise regression, ALVQ can compress the update with high precision while saving more bandwidth for data transmission in order to increase the quality of the approximation.

A. Codebook Construction from Training Dataset

A training dataset is needed to construct the original codebook. The training dataset can be sampled from the environment as follows. We divide the data into several Data Pieces (*DP*s) each with same size W ($W = n^{1/2}$). Each *DP* could then be approximated by another *DP* using piece-wise regression. According to the memory limitations of the sensor nodes and base station, the size of codebook is restricted to some limitation *Mcode*. We take the first (*Mcode*/W) *DP*s that can best approximate the other *DP*s by piece-wise regression as the "raw" codebook. This is implemented by repeatedly finding in the training dataset the top *DP* that can improve the approximation precision most if inserted into the codebook.

When the "raw" codebook is full, we perform the following LVQ learning process to polish the codebook:

For each *DP* $\boldsymbol{X}$ in the training dataset, we find the best *CDP* (Codebook Data Piece) in the codebook that can approximate it with the smallest error. We denote this *DP* in the codebook as CDP_i, and update CDP_i:

$$CDP_i = CDP_i + \alpha\,[(\boldsymbol{X}\text{-}\boldsymbol{b})/\boldsymbol{a}\text{-}\ CDP_i],$$

where $\boldsymbol{a}$, $\boldsymbol{b}$ are the regression parameters; and $0 <\alpha< 1$ is the training parameter.

After all the *DP*s in the training dataset have been tested, the codebook is adjusted and transmitted to the base station.

B. Compressing time series data in real world

After the training process is finished, the sensor node collects measured data continuously and adds it to its buffer. When the buffer is full, it divides the buffer data into several intervals each having the same size W ($n^{1/2}$). First, it maps each interval to the best *CDP* (Codebook Data Piece) in the codebook using piece-wise regression. Then it finds the interval with the largest regression error and divides it into half. If the data interval size is smaller than W, it can shift in the *CDP* until the best approximation is achieved. We keep dividing the intervals with the highest error until the maximum number of intervals is achieved (depending on the bandwidth and the buffer size). Then the approximation parameters are transmitted to the base station as compression representation.

C. Codebook Update

Because the environmental data feature may change over time, as a new data stream is collected at the sensor node, old data patterns in the codebook may become out of date and

inappropriate for the new data regression. Thus we need to insert new frequently occurring patterns into the codebook and remove out-of-date patterns. Please note that it is not always desirable to update the codebook with too many new patterns. Since the transmission bandwidth to the base station is upper bounded by *TotalBand*, the more new *DP*s we use to update codebook, the less bandwidth there is left that can be utilized for data transmission.

The algorithm works as follows:

1. Similar to [6], ALVQ found the set of *CDP*s to be inserted into the codebook, a 2-level regression subroutine is called to compress these *CDP*s and the compression representatives are transmitted to the sink.

2. The out-of-date *CDP*s are found using Least Frequently Used (*LFU*) policy and their ids are transmitted to the sink.

3. When transmission is finished, the sink first replaces out-of-date codebook with the newly transmitted *CDP*s to update the codebook. Then it utilizes the new codebook and the compression representatives to approximate the original values at the sensor node.

D. Computing Complexity of ALVQ

According to [6], the time complexity of SBR is $O(n^{1.5})$ where n is the size of data buffer in sensor node. In ALVQ, for each *DP* in the buffer, our learning process finds the best codebook data piece *CDP* and adjust *CDP* in $O(W)$ time. In addition, the compression of codebook update takes $O(M_{base} \cdot W)$. Therefore, the running time complexity of ALVQ is $O(n^{1.5} + W \cdot W + M_{base} \cdot W) = O(n^{1.5})$ which is the same as SBR and is acceptable for real sensors.

V. DYNAMIC BANDWIDTH ASSIGNMENT IN WSN

Now let us consider the problem in a general case where we want to gather the historical information from all the sensors in the sensor network. Recently a large amount of energy-saving algorithms have been proposed for efficient routing in sensor networks. Algorithms [5][7][10][11][12][13] are based on the collect and send scheme, treat all the sensors equally in the data transmission scheduling. In our collect-compress-transmit scenario, as the compression qualities show variances at different sensors, to assign more bandwidth priorities to low-compression-quality sensors is more appealing.

A. Network Topology: The LEACH Framework

LEACH (Low-Energy Adaptive Clustering Hierarchy) [7] and its variants [10][11] are among the most popular hierarchical routing algorithms for sensor networks. The idea is to form clusters of the sensor nodes based on the received signal strength and use local cluster heads as routers to the sink. The cluster heads are elected from sensor nodes randomly over time in order to balance the energy dissipation of nodes.

As described in Figure 1, in *LEACH*, all the sensor nodes send packages directly to its local cluster head and the cluster head transmits these packages to the sink through multi hops in the sensor network. Since the communication between the sensor node and the cluster head is wireless, the communication channel of the cluster head is the bottleneck of the total data transmission of all the sensors in the cluster. In *LEACH*, all the sensors in the cluster share the communication channel of cluster head evenly (for example TDMA schedule), namely, they share the same communication bandwidth of transmission data to the cluster head.

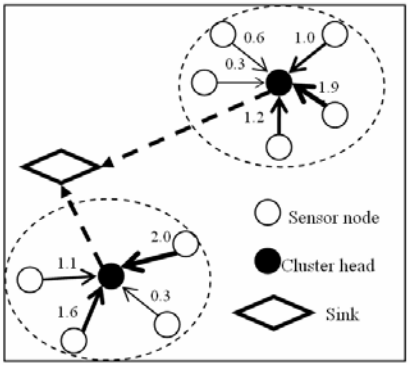

Figure 4. Dynamic Bandwidth Assignment model

As different sensors usually collect different data, it is highly possible that the compression qualities at different sensors are different even given same compression rate. Since *LEACH* assigns same transmission bandwidth for all the sensors in the cluster, the compression rates for these sensor nodes are the same. Therefore, the compression qualities of different sensors cannot be maximized and balanced well in *LEACH*.

In applications where similar compression qualities are required for all the sensors, different compression rate should be assigned to different sensors in order to maximize the overall compression qualities. As all the sensor nodes share the communication channel of the cluster head, we can take better advantage of it. We can let the cluster head assign its communication channel unevenly and dynamically to different sensors. For those sensors with low compression qualities, cluster head assigns more bandwidth to them; for those sensors with high compression qualities, less bandwidth are assigned. Therefore, with different transmission bandwidth, similar compression qualities of sensors are achieved.

This can be done by changing the channel schedule. For example, in TDMA mechanism, sensors that need more bandwidth can use the communication channel longer.

B. Dynamic Bandwidth Assignment (DBA)

Here we introduce the Dynamic Bandwidth Assignment Algorithm (Figure 4) for sensor information transmission and compression qualities balancing in sensor networks.

Our algorithm works as follows: After setting up of a cluster (with k sensors), the cluster head collects the compression quality of each sensor Q_1, $Q_2 \ldots Q_k$, and the bandwidth assigned to them $B_1, B_2 \ldots B_k$ of last transmission between these sensors and the cluster head.

Then the average compression quality for all these sensors are computed $Q_A = \sum_{i=1\ldots k} Q_i/k$. For data transmitted later, the cluster head assigns bandwidth to sensor i as $B_i - \alpha(Q_i - Q_A)$, where α is the bandwidth adjusting parameter.

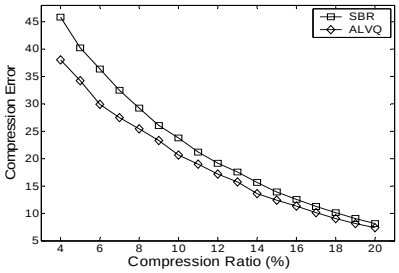

Figure 5 Varying Compression Rate

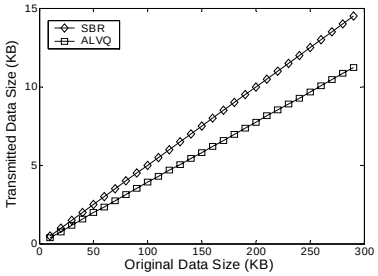

Figure 6 Transmitted Data Size

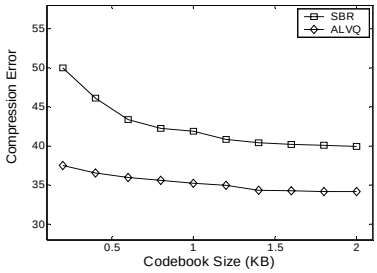

Figure 7 Varying Codebook Size

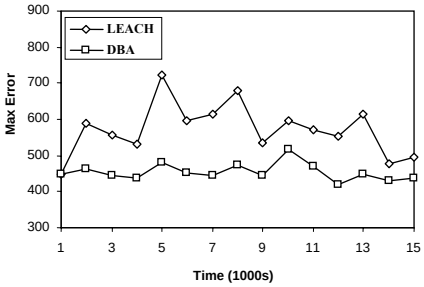

Figure 8 Maximum Errors on Synthetic Dataset

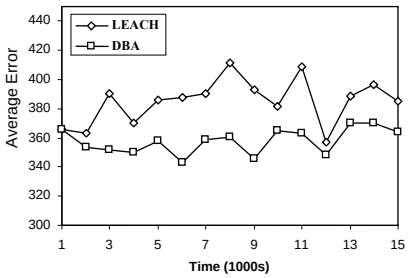

Figure 9 Average Errors on Synthetic Dataset

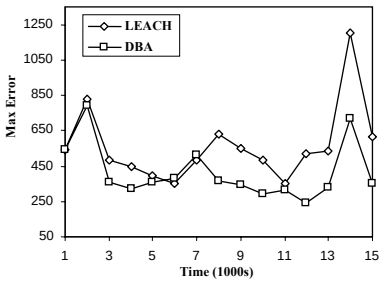

Figure 10 Maximum Errors on Real World Dataset

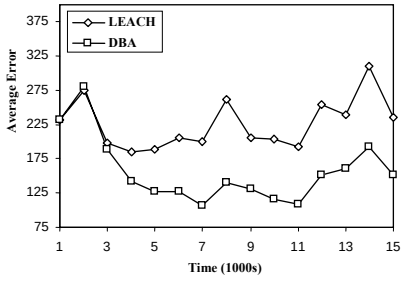

Figure 11 Average Errors on Real World Dataset

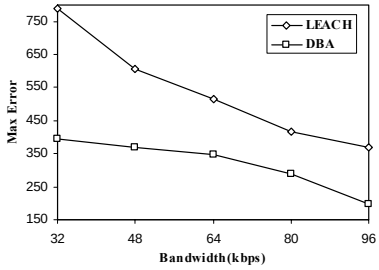

Figure 12 Maximum Errors with Varied Bandwidth

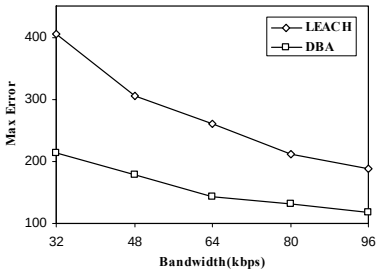

Figure 13 Average Errors with Varied Bandwidth

With the dynamic bandwidth assignment to different sensors, those sensors with low approximation quality can get more bandwidth and those with high quality will get less bandwidth. Therefore, the compression qualities for all the sensor nodes are balanced well.

VI. EXPERIMENTAL EVALUATION

A. *Description of the Dataset*

We provide a thorough experimental evaluation of our ALVQ and DBA technique on real world data set. To better demonstrate the advantage of DBA, we evaluate our DBA algorithm on synthetic dataset, too.

For synthetic data set, there are totally 300k synthetic data records generated for 8 sensors. To simulate the real sensor measurements at different spots, we generate data records with different variation for different sensors. In addition, some sharp changes are embedded to simulate the exception or abnormal events in the environment.

For real world data set, we utilize Atmo - a real dataset of atmospheric data collected at 32 sites in Washington and Oregon. More specifically, each of the 32 sites maintains the average temperature on an hourly basis for 208 days between June 2003 and June 2004. We connect these sites by a network (like in Figure 3) according their positions, and use the DBA technique to minimize the compression error given a bandwidth constraint.

B. *Data Compression within a Sensor*

We compare the performance of ALVQ and SBR [6] on the Atmo data set using the same network constraints and compression parameters. The main objective of our experimental work is to quantify the advantage of ALVQ, namely, quantify the advantage of employing LVQ learning process in the construction of the codebook, and the transmission of compressed updates. We only compare with SBR because (1) The two techniques (SBR and ALVQ) are using the same framework and (2) the experimental results of [5] show that SBR outperforms other techniques in the sensor networks' setting.

1) Varying the Compression Rate

With 3.6k data items in the buffer and codebook 2KB, we vary the compression rate from 4% to 20%. Figure 5 shows that the compression precision decreases gradually as the compression rate increases for both SBR and ALVQ. In addition, ALVQ improves the overall precision over SBR by

an average of 15% for all the compression rates, which is due to the higher accuracy of the codebook in ALVQ.

2) Comparing the Transmission Data Size

As time goes on, more data is received and transmitted to the base station. We fix the SBR compression rate as 5%, codebook size as 2KB and try to find the minimum size of transmitted data by ALVQ to achieve the same compression error. Figure 6 shows that with the same error constraint, ALVQ transfers less data than SBR, which means ALVQ achieves a higher compression rate and uses less network communications than SBR.

3) Varying the Codebook Size

We varied the codebook size from 0.1KB to 2 KB while fixing the compression rate to 5% and data file size to 10KB. Figure 7 shows that the higher the codebook size, the more precise compression is achieved for both SBR and ALVQ. This is because data can be aggregated more accurately if we have more patterns in the codebook. On average, our ALVQ algorithm improved the compression precision of SBR by around 15% for different codebook sizes.

C. Maximizing Overall Data Compression Qualities in WSN

We compare the performances of DBA and LEACH under the same network settings. The main objective of our work is to quantify the advantage of DBA algorithm over LEACH, namely, quantifying how well DBA algorithm can maximize and balance the compression qualities among different sensors.

1) Data Compression at different time moments

We evaluate the performances of DBA and LEACH on both synthetic and real world data sets. For synthetic data set, it is assumed that the data record is generated every second and the sensors transfer their compressed information every 1000 seconds. The bandwidth between sensors is set as 64 Kbps which is a typical sensor node setting. For real world data set, each measuring site is taken as a sensor and the 32 sites are assigned to 4 clusters (with 8 sensors each) according to their positions. We try to compare the maximum and average compression errors within a cluster under DBA and LEACH model. As is shown in Figure 8, 9, 10 and 11, the DBA algorithm is always consistent with small error as time goes on, while the LEACH approach which just divides the bandwidth evenly to all the sensors is sometimes good (if all the sensors need the same bandwidth), but sometimes very bad. This is because LEACH can not adjust bandwidth to achieve global optimization as in DBA for biased data distribution among sensors.

2) Varying the Bandwidth of Channel

With the real world Atmo dataset having 32 sensors, we vary the communication bandwidth between sensors from 32 kbps to 96 kbps. Figure 12 and Figure 13 show that DBA outperforms LEACH in maximum and average compression errors under all circumstances. As the bandwidth become smaller, the advantage of DBA over LEACH is more obvious although the compression errors for both model increase.

VII. CONCLUSIONS

We present a new data compression technique, designed for historical information compression in sensor networks. Our method splits the collected data into variable length and encodes each of them by an artificially constructed codebook. The values of the codebook are extracted from the training dataset and maintained dynamically as data changes. The LVQ learning process is employed to polish the codebook in the codebook construction step and codebook's updates are compressed to save bandwidth for sensor data transmission. In the last section, we propose a *DBA* algorithm to dynamically adjust communication bandwidth in order to maximize and balance compression qualities of different sensors.

In our experiments on synthetic datasets and real datasets, we show that both our ALVQ technique and DBA algorithm improve the overall precisions of the approximation in the sensor networks.

ACKNOWLEDGMENT

This work was supported by NSF grant 0330481.

REFERENCES

[1] J. Chen, D.J. Dewitt, F. Tian, and Y. Wang. "NiagaraCQ: AScalable Continuous Query System for Internet Databases." In *Proceedings of ACM SIGMOD*, 2000.

[2] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. "Multi-Dimensional Regression Analysis of Time-Series Data Streams." In *Proceedings of VLDB*, 2002.

[3] R. Cheng, D. V. Kalashnikov, and S. Prab-hakar. "Evaluating Probabilistic Queries over Imprecise Data." In *Proceedings of ACM SIGMOD Conference*, 2003.

[4] V. Cherkassky and F. Mulier. "Learning from data: Concepts, Theory and Methods." *John Wiley & Sons*, 1998.

[5] C. Liu, K. Wu, J. Pei, "A Dynamic Clustering and Scheduling Approach to Energy Saving in Data Collection from Wireless Sensor Networks" In *Proceedings of SECON*, 2005

[6] A. Deligiannakis, Y. Kotidis and N. Roussopoulos. "Compressing Historical Information in Sensor Networks." In *Proceedings of ACM SIGMOD*, 2004.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks." In *Proceeding of the Hawaii International Conference System Sciences,* Hawaii, January 2000.

[8] T. Kohonen. "Self-Organizing Maps." *pringer-Verlag*, 1995

[9] T.Kohonen, "Improved versions of learning vector quantization." In *Proceedings of the International Conference on Neural Networks*, 1990.

[10] A. Manjeshwar and D. P. Agrawal, "TEEN : A Protocol for Enhanced Efficiency in Wireless Sensor Networks." In *Proceedings of the 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, CA, 2001.

[11] A. Manjeshwar and D. P. Agrawal, "APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks." In *Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile computing*, FL, 2002.

[12] M. Younis, M. Youssef and K. Arisha, "Energy-Aware Routing in Cluster-Based Sensor Networks." In *MASCOTS*, TX, 2002.

[13] F. Ye, G. Zhong, J. Cheng, S. Lu and L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks," In *International Conference on Network Protocols*, Paris, France, 2002.

[14] M. Ruhl, H. Hartenstein, "Optimal Fractal Coding is NP-Hard", In *Data Compression Conference*, Snowbird, Utah, 1997.