Energy-Efficient Data Compression in Clustered Wireless Sensor Networks using Adaptive Arithmetic Coding with Low Updating Cost

Tossaporn Srisooksai, Kamol Kaemarungsi, Poonlap Lamsrichan and Kiyomichi Araki

Abstract—**This work presents a study of energy reduction technique using data compression based on arithmetic coding in clustered wireless sensor networks to maximize the network's lifetime. Initially, a simulation approach is used to investigate the effect of multiple data types found in environmental monitoring application on data compression and the effect of cluster's parameters on their energy consumption. This study points out the important of probability models of multiple sensor data such as temperature and relative humidity on the arithmetic coding's performance. The investigation results provide insights for designing an energy-efficient arithmetic coding framework that is suitable for compressing multiple data types in clustered multi-hop wireless sensor networks. Finally,an implementation of an adaptive local data compression algorithm derived from our findings and design framework on a set of four TinyOS based Tmote Sky wireless sensor nodes equipped with temperature and relative humidity sensors is presented with approximately 54 percent data compression results.**

Index Terms—**Arithmetic Coding, Data Compression, Energy-Efficient, Wireless Sensor Network.**

I. Introduction

A wireless sensor network (WSN) is suitable for a large scale *data gathering* in environmental monitoring application [1] where a large number of sensor nodes are scattered around a wide geographical area. This type of network is often organized in a form of clustered multi-hop topology where sensor data are forwarded from clusters of end-nodes toward a sink node in a multi-hop fashion. The energy consumption of each node is one of critical issues that requires careful management in order to maximize the lifetime of the WSN. The main energy consumer in each node is the wireless transceiver when it requires energy to transmit and receive data over the air [2].

A data compression [3], [4], [5], [6] is one of possible techniques that can reduce the amount of data exchanged between wireless sensor nodes along the path toward the sink node. Consequentially, it can reduce the energy consumption of wireless sensor node. However, the energy required to perform data compression in each node must be lower than the energy that can be saved by its communication operations. In the literature, there is an approach of data compression algorithms in WSNs called the *local data compression* [5], [6] in which each sensor node performs data compression locally without distributed collaboration among sensor nodes. In this approach, several classical entropy compression schemes such as Huffman coding and arithmetic coding (AC) can be modified to use in WSNs. This approach is studied here because it is simple to implement and has lower complexity.

Although simple data compression algorithms may be able to save the energy consumption in single-hop WSNs such as star topology, they might not maximize the network's lifetime in case of multi-hop WSNs. Typically, the network's lifetime will depend on the remaining energy of sensor nodes which locate next to the sink nodes on multi-hop topology or the cluster head nodes in clustering topology. These nodes are usually called *energy-critical nodes*. The energy saving of data compression algorithm will depend on the topology and the routing protocol of the network.

An interesting issue raised by [7] is that there are often multiple data types generated in each sensor node from different environmental phenomena. For instance, SHTxx series devices from Sensirion, which are the well-known commercial sensors used in WSNs, have two sensor types on the same package which are temperature and relative humidity. This characteristic of multiple data types can influence the choice of data compression algorithm that can maximize the energy saving performance of WSN. Therefore, in this paper we investigate a data compression using arithmetic coding (AC) to compress multiple data types in clustered WSN. The optimal statistical parameters of input symbols to the data compression algorithm and network topological parameters are identified in order to maximize the energy saving of energy-critical nodes in WSNs. Based on the findings of this study, an implementation of energy-efficient data compression in a small clustered wireless sensor networks is incorporated into TinyOS [8] and tested on a set of four Tmote Sky [9] wireless sensor nodes equipped with Sensirion's SHT15 sensors.

The rest of this article is organized as follows. First, Section II describes the system model of wireless sensor network in this study, which can be divided into the network topology model, the arithmetic coding model, and the energy

Manuscript received April 16, 2011; revised June 2, 2011. This work was financially supported by Thailand Advanced Institute of Science and Technology - Tokyo Institute of Technology (TAIST-Tokyo Tech), National Science and Technology Development Agency (NSTDA), Tokyo Institute of Technology (Tokyo Tech), National Research Council of Thailand (NRCT) and Kasetsart University (KU).

T. Srisooksai is with TAIST-Tokyo Tech, ICT for Embedded System Program, Department of Electrical Engineering, Faculty of Engineering, Kasetsart University, Thailand (email: srisooksai@gmail.com).

K. Kaemarungsi is with Embedded System Technology Laboratory, National Electronics and Computer Technology Center, Thailand.

P. Lamsrichan is with Department of Electrical Engineering, Faculty of Engineering, Kasetsart University, Thailand.

K. Araki is with Department of Electrical and Electronic Engineering, Tokyo Institute of Technology, Japan.

consumption model. Next, the simulation analysis and results of energy saving are discussed in Section III. Section IV derives the real world probability model of sensor data from the actual temperature and relative humidity data of SensorScope project [1], [10]. The findings from our simulation study are summarized in Section V, which provide a design guideline for energy-efficient data compression in clustered wireless sensor networks. After that, details of actual implementation of an adaptive local data compression algorithm based on arithmetic coding with low cost updating procedure are explained in Section VI. In Section VII, the data compression performance results of the actual implementation are presented. Finally, Section VIII is the conclusion.

II. Procedure for Paper Submission

A. Clustered Network Topology

A cluster of wireless sensor nodes can be formed by a routing protocol. The nodes within the vicinity of a cluster head will join the cluster if their distance to this cluster head is shorter than the other cluster head. In this work, we focus on a small-area cluster in which the distances (d) between sensor nodes and a cluster head are approximately 10 to 15 meters, and each sensor node need only one hop to send their data to the cluster head. One of existing routing protocol techniques that can construct this kind of cluster is the Semantic/Spatial Correlation-aware (SCT) scheme [11]. Assuming that a simple cluster consists of three wireless sensor nodes as shown in Fig. 1, one node is selected as a cluster head node using the SCT routing protocol [11]. Other two sensor nodes are cluster's members.

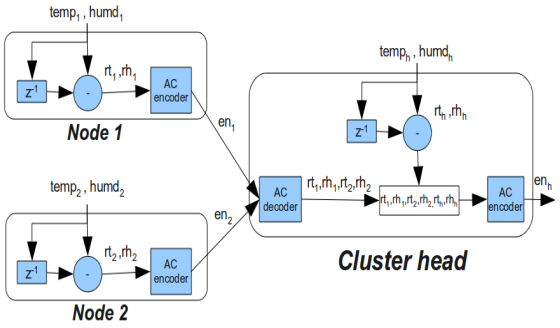

Fig. 1. Cluster Arithmetic Coding

In Fig. 1, each sensor node first obtains its own sensor data ($temp_i$, $humd_i$). The residue values of multiple data types which are temperature (rt_i) and relative humidity (rh_i) are then computed by subtracting current observed value with previous observed value in this system model. Each sensor node feeds both residue values to its data compression algorithm called encoder based on arithmetic coding (AC) algorithm described in [12]. Then, all member nodes transmit their encoded values (en_i) to the cluster head over a single hop. When all encoded values from cluster's members reached the cluster head, it decodes the received bits using data decompression or arithmetic coding decoder. These decoded values are then concatenated with residue values from local sensors of cluster head node and are fed into another AC encoder. The newly encoded bits are transmitted to the next level of cluster head that will forward them to the next cluster head without re-compression until all data reach a sink node. Finally, the sink node decodes the total received bits to obtain all residue values. Assuming that the sink node has all network routing information, the sink node can determine which residue value belongs to which node in the network. When adding these values to their previous values, the sink node can recover the original values of sensor data. These values are maintained in sink node for the next cycle of data decompression.

B. Arithmetic Coding

The class of entropy coding is usually used to compress differences of consecutive data called *residue* values. In [6], the authors pointed out that the residue values from real WSNs system could be fitted with *Laplacian* distribution which has a highly skew probability function. In theory, AC is especially useful when dealing with data sources with highly skewed probability distributions [13].

To the best of our knowledge, there are only a few literatures that discuss the compression of multiple data types in wireless sensor nodes. For local data compression approach, the simple compression scheme in [5] was applied on multiple data types and a slightly modified scheme in [7], which was specifically designed for multiple data types, showed a better data compression performance than the previous one. However, both works adopted Huffman coding and exploited only temporal correlation in local sensor nodes but did not exploit other correlation existing in clustered WSNs for environmental monitoring such as spatial correlation and correlation of residue values from multiple data types. In this work, we investigate alternative data compression technique for multiple data types that differs from the work of [5], [7].

The important information needed to be considered when applying AC in a system is the probability model (M) of data symbol applied in AC encoder/decoder as depicted in Fig. 1. The probability model is the probability of occurrence of symbols fed into AC encoder/decoder. Symbols in our case are the residue values. For example, if we consider a compression of residue values fallen in range of integer from 0 to 3 and probabilities of occurrence of residue values 0, 1, 2 and 3 are 0.60, 0.20, 0.15 and 0.05, respectively. In AC algorithm, cumulative probability of each residue value is used to determine the compressed output data. Thus, the model (M) applied in AC encoder/decoder can be defined as Table I.

TABLE I: Example Model Residue Value Used in AC Algorithm

Residue value	Cumulative Probability
0	0.60
1	0.80
2	0.95
3	1.00

C. Energy Consumption Model

An energy consumption model is needed to investigate the performance of data compression using arithmetic coding in clustered WSN in the next two sections. Basically, we follow the energy consumption model of [6]. In clustered WSN, a cluster head node is considered as an energy-critical node.

The life-time of a WSN depends on this node. Therefore, the energy consumption model is applied to this type of nodes.

First, the energy consumption of a transmission with no compression can be calculated as a baseline quantity from the energy required to transmit and receive non-compressed bits (N) which can be written as

$$E_{noncompressed} = (N \cdot V \cdot I_{Tx} \cdot T_{Tx}) + (N \cdot V \cdot I_{Rx} \cdot T_{Rx}), \quad (1)$$

where I_{Tx} is a transmitter current, T_{Tx} is a transmitted time per bit, I_{Rx} is a receiver current, T_{Rx} is a received time per bit, and V is an operating voltage. Note that the energy consumption $E_{noncompressed}$ has a unit of Joule or Watts-seconds. In this study, the number of non-compressed bits (N) is based on transmitted/received packets with fixed header size of 8 bytes and variable payload in multiple bytes. For example, a sensor node that needs to transmit/receive its temperature and humidity data in one packet will require 4 bytes of payload.

Second, the energy consumption model for a transmission with data compression can be derived based on the simple clustering scenario in Fig. 1. This energy consumption should include the energy consumption of receiving, transmitting, and compressing operations of compressed bits ($\hat{N}$). Generally, a data compression using arithmetic coding (AC) encoder consists of two processes which are common updating process and scaling process. Conventionally, the sequence of symbols is encoded into a floating point number at the output which is a value between 0 and 1.

This floating point can be converted into a binary sequence which uniquely represents the input sequence. However, the algorithm can be adapted to use integer number too as described by [12]. To obtain the compressed output, the common updating process is an iterative process that updates low-high interval which depends on each input symbol in a sequence and its corresponding cumulative probability values defined in Table. I. The scaling process is a process that adjusts the value in the interval to be within an allowable precision (number of allowable bits) as the input sequence gets longer. Ultimately, the encoded bit sequence which uniquely represents the final updated low-high interval of the input sequence is obtained.

For this study, the common updating energy cost was originated from performing four additions, two integer multiplications, two shifts, and two comparisons for every single residue value fed into the encoder. The scaling energy cost requires uniform cost of three additions and two shifts. In our work, the number of scaling ($N_{scaling}$) for each residue value is counted by our simulator. For AC decoder, the energy cost is assumed to be equivalent to the AC encoder cost because it can be considered as a reverse operation of the encoder. Another energy cost is incurred when calculating a residue value which is the difference between current observed value and previous observed value. This energy cost requires two additions and one comparison.

Based on the calculation of the baseline quantity in (1), the energy consumption for a transmission of compressed data can be modified as

$$E_{compressed} = \hat{N} \cdot V \cdot I_{Tx} \cdot T_{Tx} + \hat{N} \cdot V \cdot I_{Rx} \cdot T_{Rx} + E_{perform}, \quad (2)$$

where most of the variables are similar to (1), the new term called $E_{perform}$ is the energy consumption of data compression algorithm. The $\hat{N}$ is the number of compressed bits. It contains the fixed header of packet which is not compressed and the compressed payload data which may be varied. The size of the header is the same for both compressed $\hat{N}$ and non- compressed N data which is 8 bytes. The $E_{perform}$, which is the energy required to perform AC, can be obtained by summing up the energy consumption of encoding, decoding, and differing operations as

$$E_{perform} = E_{encode} + E_{decode} + E_{diff}. \quad (3)$$

The energy consumption of encoding and decoding operations are the same and can be obtained from (4) while the energy consumption of calculating the residue value can be deter- mined by (5).

$$\begin{aligned} E_{encode} &= N_{residue}(4e_{add} + 2e_{mul} + 2e_{sht} + 2e_{cmp}) \\ &\quad + N_{scaling}(3e_{add} + 2e_{sht}) \\ &= E_{decode} \end{aligned} \quad (4)$$

$$E_{diff} = N_{diff}(2e_{add} + 1e_{cmp}), \quad (5)$$

where $N_{residue}$ is the number of residue value fed into the encoder, $N_{scaling}$ is the number of scaling of each residue value fed into AC encoder/decoder, and N_{diff} is the number of calculation of the residue values of sensor node for each encoding. The energy consumption for basic operations, which are the addition (e_{add}), the multiplication (e_{mul}), the shift (e_{sht}), and the comparison (e_{cmp}) used in our simulator, are based on the values obtained from data sheets of a sensor node that uses the CC2420 RF transceiver and the ARM7TDMI micro-controller [6]. The values of all required parameters are listed in Table II.

TABLE II: PARAMETER OF ENERGY CONSUMPTION MODEL

Parameter	Value	Parameter	Value
I_{Tx}	17.4 mA	e_{add}	2.13 nJ
I_{Rx}	19.7 mA	e_{mul}	6.39 nJ
T_{Tx}	3.2^{-5} s	e_{cmp}	2.13 nJ
T_{Tx}	3.2^{-5} s	e_{sht}	4.26 nJ
V	3.3 V		

Finally, the saved energy (E_{saving}) in energy-critical sensor node can be determined by

$$E_{saving} = E_{noncompressed} - E_{compressed}. \quad (6)$$

III. SIMULATION ANALYSIS

In order to understand the effect of residue data on the energy saving defined by (6), we study the effect of various probability models (M) based on different statistical parameters of residue sequence (S) in a small clustered WSN consisting of five sensor nodes. One sensor node is selected as a cluster head while the rest are cluster's members. Each cluster's member requires only one hop to send its residue values to the cluster head. Using simulation approach, the probability models (M) used in AC described in Section II-B are configured as follows. The residue value range used in all

models is between -40 and 40. The cumulative distribution functions of models were generated based on three well-known distributions: *normal distribution*, *laplacian distribution* and *gamma distribution*. The cumulative distribution functions were created with zero mean and three different standard deviations (σ) which were 1, 2 and 5. Thus, we have nine probability models in this simulation study. Note that these distributions were selected based on the findings of [14] that residue data are fitted well with normal distribution and the subsequent findings of [6] that real world data are fitted well with Laplacian instead of normal distribution. The gamma distribution is included because it has a narrowest distribution among the three distributions and intuitively it could fit the residue data that usually deviate slightly around 0 value.

The residue sequences (S) were randomly generated based on these three distributions and three standard deviations. For each simulation, 3,000 samples of residue data were generated. In each sensing period, assuming that each node was equipped with two different sensors as shown in Fig. 1, two random samples were fed into its own AC encoder. Then, four sensor nodes transmitted encoded or compressed bits over a single hop to the cluster head. Assuming that there was no interference over the wireless channel and no data frame was lost during the simulation.

After receiving all data from each member, the cluster head decoded the received bits using AC decoder and concatenated its own two sensor samples into the sequence. Therefore, the new sequence of ten samples from all sensors in this cluster were put into the AC encoder of the cluster head. This ended one sensing period. The simulation continued until 3,000 samples were processed. At the end of each simulation, the energy saving defined by (6) for the energy-critical node or the cluster head, which determines the network's lifetime, was measured. We repeated this simulation for 100 times and calculated the average energy saving with 95% confident interval. The results are plotted in Fig. 2 and Fig. 3.

Fig. 2 plots the results of energy saving on y-axis over different combination of distributions between the residue sequence (S) and the probability model (M) used in the AC encoder on x-axis. We drew lines to connect between different combinations that used the same standard deviation on both random residue sequences and probability models. An example of result for *gam-nor* with standard deviation $\sigma = 1$ means that the residue sequence was generated with gamma distribution (S_{gam}) and it was fed into an AC encoder with normal distribution (M_{nor}). The average energy saving result for this particular simulation setup was 83%.

Fig. 3 shows a different perspective of energy saving on y-axis over the standard deviation parameters of the residue sequence and the probability model in encoder on x-axis. In this figure, we fixed the distributions of both residue sequence and probability model to be normal distribution. The result above the label 5-1 in Fig. 3 means that the residue sequence was normally distributed with standard deviation of $\sigma = 5$ and it was fed into AC encoder that used normal distribution model with $\sigma = 1$. The energy saving result in this particular simulation was 52.5%.

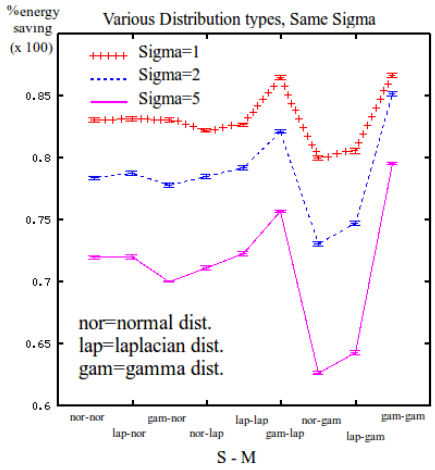

Fig. 2. Sequence (S) and model (M) are generated using the same standard deviation (σ).

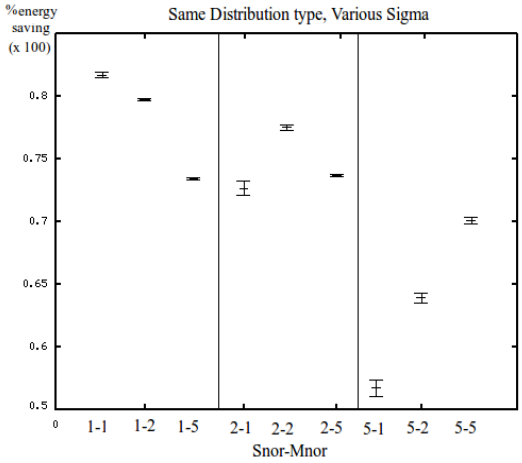

Fig. 3. Sequence (S) and model (M) are generated using the same distribution types but difference standard deviation (σ).

IV. Real World Data Analysis

Based on the simulation results in Fig. 2, the highest energy saving can be obtained under the conditions that the sequence of residue values (S) and the probability model (M) in AC encoder were fitted well with gamma distribution and had lowest standard deviation. These findings motivated us to investigate further on the parameters of cluster network topology that can produce data with the gamma distribution and the lowest standard deviation in real world WSN. Additionally, we found that statistical characteristic of multiple data types in real world WSN, i.e. the distribution and its parameters, can be influenced by other parameters of cluster network topology. We identified two of them here which are the sampling period and the distance between each sensor nodes.

In this section, we obtained real world data from the SensorScope project [1], [10] and used them to analyze the real world probability model. A data set called "luce" was used because it contained GPS data for each sensor node. This enabled us to approximate the distance between pairs of sensor nodes. The data of 97 sensor nodes were provided in their website [10]. The sensor nodes are scattered around the area of 278m by 409m. The sensor data provided by the

project were *temperature* and *relative humidity* obtained from Sensirion's SHT75 devices. We formed fives clusters of five sensor nodes based on the provided Global Positioning System (GPS) locations. We formed fives clusters of five sensor nodes based on the provided Global Positioning System (GPS) locations. We selected one sensor node as a cluster head while the rests became cluster's members and each member required only one hop to transmit residue values to the cluster head.

We varied two parameters, which are the sampling period and the distance between nodes in each cluster, and collected relevant data from all sensor nodes to determine the corresponding probability model for each cluster. The sampling periods are classified into three intervals which are 30 seconds (*30s*), 5 minutes (*5m*) and 30 minutes (*30m*). The distance (*d*) between each sensor node in the cluster can be divided into three groups: 10-15 meters (*c10*), 20-30 meters (*c20*) and 50-70 meters (*c50*). Using the *chi-square goodness-of-fit* test as showed in (7) [15], the results of the best fitted distribution among normal, Laplacian and gamma distributions are shown in Table III where g and l mean that the data are fitted with gamma and Laplacian distributions, respectively.

$$\chi^2 = \sum \frac{(observed_{value} - \exp ected_{value})^2}{\exp ected_{value}} \qquad (7)$$

The results in Table III show that when the shorter sampling periods and the smaller distance between sensor nodes in a cluster are applied, the multiple data types in each cluster are fitted well with gamma distribution. In contrast, when the larger sampling frequency and the larger distance of each sensor node in cluster are applied, the data are fitted well with Laplacian distribution. This is due to the change in spatial correlation of sensor data in different cluster formations.

TABLE III: MODEL SOF MULTIPLE DATA TYPE SOVER DATA SAMPLING PERIOD AND RADIUS OF CLUSTER

Cluster no.	30s			5m			30m		
	c10	c20	c50	c10	c20	c50	c10	c20	c50
no.1	g	g	g	g	g	l	l	g	l
no.2	g	g	g	g	l	g	g	g	l
no.3	g	g	g	g	g	g	g	l	l
no.4	g	g	l	l	l	l	g	l	l
no.5	g	g	l	g	l	l	g	g	l
no.1	g	g	g	g	g	l	l	g	l
no.2	g	g	g	g	l	g	g	g	l
no.3	g	g	g	g	g	g	g	l	l
no.4	g	g	l	l	l	l	g	l	l
no.5	g	g	l	g	l	l	g	g	l

V. DESIGN GUIDELINE FOR ENERGY-EFFICIENT DATA COMPRESSION

Based on the simulation results in the previous sections, there are three important findings that can help maximize the energy saving when applying arithmetic coding in clustered WSNs. First, the results in Fig. 2 indicated that the energy saving of energy-critical node is higher when the sequences fed into encoder have a small standard deviation (σ) for all distributions of sequence (S) over all three models (M). This provides an important insight into the consideration of the routing strategies in multi-hop WSNs with AC. Typically, the small σ of residue values should be found in small clustered routing more than in non-clustered routing due to higher spatial correlation within the same geographical area. Thus, implementing the AC in cluster topology should performs better in energy saving.

Second, the energy saving of energy-critical node can be maximized when the standard deviation of feeding sequence (S) is the same as the standard deviation (σ) used in the probability model (M) of encoder as illustrated in Fig. 3 by energy saving of 1-1, 2-2, and 5-5 pairs in the x-axis. This provides an insight information that the probability model of AC encoder/decoder should be adapted according to the standard deviation of feeding sequence.

Third, to correctly encode and decode compressed data, all wireless sensor nodes need to use the same probability model (M) for both encoder and decoder in the same communication cluster. To maximize the energy saving, the distance (d) between each sensor node in a cluster should be pre-configured in the range of 10-15 meters and each sensor node should require only one hop for sending their residue values to a cluster head. This kind of cluster should yield the residue data set of multiple data types that are fitted with gamma distribution (S_{gam}) as observed in Table. III. If an AC encoder with gamma model (M_{gam}) is used to compress these sequences (S_{gam}), the energy saving should be maximized as shown in Fig. 2. A simple approach is to implement a static probability model of gamma distribution all the time for all arithmetic coding in clustered WSN. However, this may not maximize the energy saving. The reason of this can be observed in the 8th column of Table. III. Even though the cluster is pre-configured to provide gamma sequences S_{gam} as defined by the range (d) and the single hop parameters, it is possible that the distribution of data sequence could be changed into Laplacian distribution when the sampling period is longer. This provides an insight that the probability model (M) of AC encoder/decoder should also be adapted according the distribution types of feeding sequence (S).

Finally, to achieve the maximal energy saving, we propose to implement an adaptive probability model in the arithmetic coding base on the presented optimal parameters in this study. However, the sensors nodes will require higher cost of communications in order to update the table of probability model. This is because the cluster head, which is the only node that has knowledge of frequency of occurrence of all residue values in the cluster, has to update the table of probability model to all of its members in the cluster. This

implementation problem is further investigated on a real WSN system presented in the next section.

VI. Implementation of Adaptive Arithmetic Coding with Low Cost Updating Procedure

In this section, a simple approach is proposed to solve the high cost of communication issue described in the previous section. The detail of our low cost updating procedure is presented as follows. First, the sink node uses *chi-square goodness-of-fit* test as expressed in (7) [15] to find the best fitted distribution model for residue data received from each cluster. The $observed_{value}$ defined in (7) is a number of occurrence of a residue value while the $expected_{value}$ is the probability of occurrence of that residue value based on probability density function (*pdf*) of a given distribution model. Note that the sink node is not typically an energy- constrained node in WSN which enables it to perform the test.

Then, the sink node sends only standard deviation (σ) and type of distribution that provides the lowest χ^2 score to each cluster head. Subsequently, each cluster head forwards these information to all sensor nodes in its cluster. The sensor nodes calculate the table of data model (similar to Table I) for AC using the *pdf* of fitted distribution model and standard deviation (σ) parameter. Using this updating procedure, the data model used by AC encoder/decoder in our implementation can be adapted according to the standard deviation and the type of distribution of a feeding sequence. This will allow the data compression to achieve the maximal energy saving in multi-hop clustered WSNs. A real implementation of energy-efficient data compression in a small clustered wireless sensor network with the proposed simple updating procedure is described in the following subsections.

A. Hardware and Software Setup

A real WSN system used in this investigation is depicted in Fig. 4. The system consisted of four wireless sensor nodes which were based on Tmote Sky platform [9]. Three of the nodes formed a small cluster in which one of them was selected as a cluster head, while the other two were cluster's members. Each cluster's member required only one hop to send its residue values of sensor data to the cluster head. The distance (d) between each sensor node and its cluster head in a cluster of the system was set at 10 meters in an outdoor environment with clear line-of-sight. All three Tmote Sky nodes in the cluster were equipped with Sensirion's SHT15 sensors [16] which allowed each node to measure temperature and relative humidity data. Note that we obtained the SHT15 sensors later and soldered them on the empty footprints of the Tmote Sky boards. The last sensor node, which was operating as a sink node, was connected to a laptop computer via a Universal Serial Bus (USB) interface. All Tmote Sky nodes were loaded with TinyOS [8] which were programmed to run as sensor nodes, cluster head node, and sink node, respectively. Note that TinyOS is an open source embedded operating system which is specifically designed for event-driven low- power wireless sensor devices [8]. To implement the adaptive arithmetic coding algorithm and proposed updating procedure, new software components in the TinyOS were developed using NesC (Network Embedded System C) language which is an extended version of the C programming language [17]. The laptop in Fig. 4 was running a Linux Operating System based on an Ubuntu distribution.

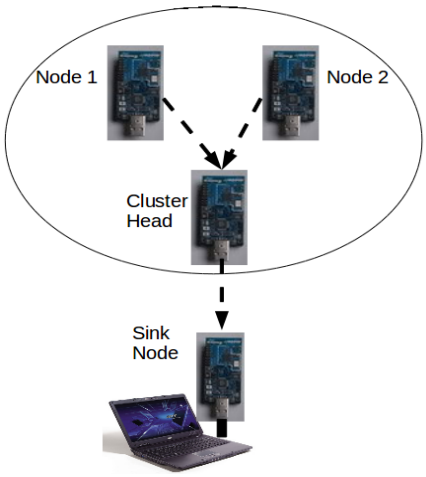

Fig. 4. WSN system

B. Node Programming

1) *Sensor Node*: The roles of wireless sensor nodes including cluster head in this real system are the same as those described in Section II-A. Fig. 5 shows a block diagram of a NesC program called ACoderC that implemented the wireless sensor nodes in the clustered network of Fig. 4. The sensor node's program is built from the basic components such as temperature sensor (TemperatureC), humidity sensor (HumidityC), over-the-air messaging modules (AMReceiverC, AMSenderC, and ActiveMessageC), timer module (TimerMil- liC), and light emitting diode (LED) module (LedsC). Each block represents a component in a form of either a module or a configuration which are basis of TinyOS programming [17]. The notations of different boxes in the diagram are as follows. A single box is a module, while a double box is a configuration [18]. The dashed border lines on a box indicate that the component is a generic component which can be instantiated more than once [18].

Basically, a module contains the actual implementation of the NesC code, while a configuration describes all wirings among multiple components to form a new component which enables the new component to perform its operations [17]. The arrows connecting these blocks in Fig. 5 are called "wires" which represent interfaces of communication between components. The label on each arrow represents the name of interface that components use to communicate between them. The components that have wires with arrow's heads pointing into them provide interfaces, while the component at the origins of the arrows is the one that use the interface. An interface contains the routines that a component providing the interface must implement, and that can be called by components using the interface. The routines can be of two types: command and event. The command can be invoked by clients of the interface to perform an operation. On the other hand, the event is returned when a given condition is raised (typically when an operation is complete, a

resource becomes ready etc.). The provider of the interface must send a signal corresponding to the event, and the client of the interface must implement an event handler for that signal. Further details of NesC programming for TinyOS is beyond the scope of this work and can be found in [17].

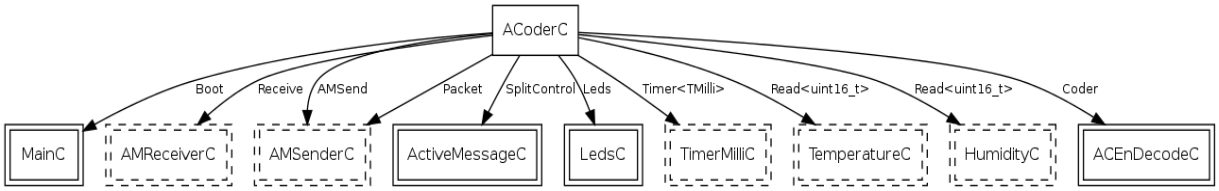

Fig. 5. Block Diagram of NesC program implemented in Sensor Node

All components and interfaces shown in the block diagram of Fig. 5 are common to several applications described in the tutorial documents of TinyOS [18]. The new component and interface for our arithmetic coding algorithm are *ACEnDecodeC* component and *coder* interface. The ACEnDecodeC component consists of two functions: encoder and decoder based on arithmetic coding described in Section II-B. The coder interface consists of two commands: encode and decode. In our case, ACoderC, which is the application designed for sensor node, is wired to ACEnDecodeC with coder interface. Therefore, the ACoderC can use encode and decoder commands to call the encoder and the decoder functions when the node needs them.

2) *Sink Node*: The sink node in our implementation requires the abilities to be able to communicate over both radio and serial port (via USB). This node's functionality is similar to the *BaseStation* application which is described in one of the tutorials of TinyOS [18]. The new sink node application must allow the compressed sensor data to be forwarded from the radio interface to the serial interface. However, due to the limited processing capability of the micro controller on the Tmote Sky platform, the tasks of determining standard deviation σ, performing chi-square goodness-of-fit test to determine the type of distribution model, and simple updating procedure to the sensor nodes, which are described in the beginning of Section VI, are moved to the laptop using the normal C language programming on the Linux operating system. The pseudo-code of the sink node is listed as follows.

```
main()
    {
      initial_model_paramameter();
      initial_ACmodel();
      create_expect();
      number_value=0;
     for (;;)
      {
        read_sf_packet();
        acdecoder();
        number_value++;
        create_observe();
        if (number_value >= UPDATE_PERIOD){
            sigma_change=check_sigma();
            if (sigma_change == 1){
              sigma_current=sigma_new;
              create_expect();
              update_flage = 1;
        }
        dist_change=check_dist();
        if (dist_change == 1){
           dist_current=dist_new;
           update_flage = 1;
        }
```

TABLE IV: THE RESULT OF REAL WSN SYSTEM WHEN THE SAMPLING PERIOD AND UP DATE PERIOD ARE VARIED

Case no.	Sampling period	Updating period	Total no. of communication between sink and cluster head	No. of update	% Compress
1	30 sec.	2.5 min.	1500 times	24 times	53.90%
2	30 sec.	15 min.	1500 times	5 times	55.02%
3	3 min.	15 min.	480 times	6 times	54.04%
4	3 min.	30 min.	480 times	2 times	53.80%

```
      If (update_flage == 1){
         prepare_packet();
         update_ACmodel();
         write_sf_packet();
      }
      update_flage = 0;
      number_value=0;
    }
```

The algorithm of the sink node's pseudo-code can be explained as follows. In the initial step, *initial_model_parameter()* function defines the initial values of sigma, sigma threshold, and distribution types (*sigma_current, threshold, and dist_current*). Then *initial_ACmodel()* function initializes ACmodel using *sigma_curre*nt and *dist_current* values defined by the first function. In order to generate the expected value used to

calculate *chi-square goodness-of-fit* as expressed in (7), *create_expect()* function creates an initial table and keeps initial expected values for normal, Laplacian, and gamma distribution with initial sigma value.

Next, inside the infinite for-loop, *read__sf__packet()* function reads encoded data from the serial port (USB interface) in which those data are decoded by *acdecoder()* function to obtain the residue values of each sensor node. Then, *create_observe()* function counts the occurrence of each residue value and creates the table of observed values. This table is used for calculating the chi-square *goodness-of-fit* as described in (7). The next part of code inside the if-condition is the updating procedure. The statistical characteristics of the next period are predicted by statistical characteristics of the current period. For example, the sigma and the type of distribution of residue data occurred in the first half of an hour is used as the predicted model of residue data occurred in the second half of an hour. When the number of residue data (*number_value*) is more than the number defined in *UPDATE_PERIOD* variable, the sigma of the residue value is determined and kept in *sigma_new* variable. If the difference between *sigma_new* and *sigma_current* is greater than a *threshold* value, the *sigma_change* variable is set to change status or 1. The type of distribution is also determined using *chi-square goodness-of-fit* as expressed in (7). If either the sigma or the type of distribution is changed, the AC model will be updated and the new sigma or new type of distribution is sent to all sensor nodes.

VII. Performance Evaluation of Real Implementation

Based on the energy consumption model described in Section II-C, the total energy saving of the clustered WSN with data compression can be determined by two factors: the communication energy saved by compression rate and the energy consumed by performing data compression algorithm. However, in the real implementation with the Tmote Sky platform it is not feasible to measure the energy consumption when a node performs arithmetic coding directly. To evaluate the energy-saving performance of the proposed algorithm, this study resorts to approximate the compression ratio of the sensor data that the system can achieve. Another indication that can determine the energy-efficiency of the proposed algorithm is the number of times that the sink node sends updating parameters to all wireless sensor nodes. The lower value of this number means that the adaptive arithmetic coding requires lower cost and energy to update data model among sensor nodes. The number of times the sink node sends updating parameters to the cluster head in Fig. 4 is used as a representation of updating cost in this evaluation.

The best energy saving system should minimize the number of updates while maximize the data compression ratio of the energy-critical node or the cluster head in the clustered wireless sensor network. To investigate the performance of the real system based on the four Tmote Sky nodes, four different experiments (cases) were conducted by varying the *sampling period* described in Section IV and *update period* described in Section VI-B2. The measurement results from the small clustered are shown in Table IV.

When comparing the results of Case 1 and Case 2, in Table IV, at the same sampling period of 30 seconds, Case 2 which had the longer updating period of 15 minutes provided a better data compression ratio than Case 1 with 2.5 minutes of updating period. Case 2 also had the lower number of updates than Case 1. This means that Case 2 has lower updating cost while achieving a better compression ratio by approximately 1%. In other words, Case 2 could save more energy on both transmit data and updating cost. The reason why a system with shorter updating period requires higher number of updates is that it is sensitive to sudden fluctuation of small number of residue data which reflects in the change of standard deviation (σ). When the longer updating period is used, the total number of accumulated residue data is larger which results in a more stable value of standard deviation. That is the standard deviation of this case was not less affected by the fluctuation of residue values receiving at the sink node.

The observation on the updating periods of the first two cases provides an insight into the updating procedure proposed in Section VI-B2. The updating period should be long enough in order to eliminate the impact from sudden fluctuation of residue values. The proper updating period can minimize the number of updates which in turn can reduce the energy consumption. A similar conclusion can be made on the measurement results of the last two cases in Table IV. At the same sampling period of 3 minutes in Case 3 and 4, the compression ratios were comparable while the updating periods for both cases were long enough at 15 and 30 minutes. Note that Case 4 had the lowest number of updates while it maintained similar data compression performance as in Case 3.

VIII. Conclusions

A proposed data compression in clustered wireless sensor networks using adaptive arithmetic coding with low updating cost is presented in this article. The real implementation of wireless sensor network was based on the energy-efficient framework developed from simulation study and actual sensor data in the first part of this work. Using adaptive arithmetic coding, the proposed system could achieve approximately 54% data compression with low updating cost of relaying distribution model from the sink node to other sensor nodes in the network. The simple updating procedure suggested in this work is based on the prediction of the statistical model of the next updating period using the statistical characteristic of the current updating period. However, the results shown in the last section might not be the optimal one. We expect that a better and more complex prediction procedure could be developed to improve energy efficiency of the clustered wireless sensor network.

References

[1] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Out-of-the-box environmental monitoring," in *Int. Conf. on Info. Proc. in Sens. Netw. (ISPN)*, 2008, pp. 332–343.

[2] K. C. Barr and K. Asanovic´, "Energy-aware lossless data compression," *ACM Trans. Computer Systems.*, vol. 24, pp. 250–291, 2006.

[3] Z. Xiong, A. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 80–94, sep. 2004.
[4] J. Chou and D. Petrovic, "A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks," in *Proc. IEEE INFOCOM*, 2003, pp. 1054–1062.
[5] F. Marcelloni and M. Vecchio, "An efficient lossless compression algo- rithm for tiny nodes of monitoring wireless sensor networks," Comput. J., vol. 52, no. 8, pp. 969–987, 2009.
[6] Y. Liang and W. Peng, "Minimizing energy consumptions in wireless sensor networks via two-modal transmission," SIGCOMM Comput. Commun. Rev., vol. 40, no. 1, pp. 12–18, 2010.
[7] P. Sornsiriaphilux, D. Thanapatay, K. Kaemarungsi, and K. Araki, "Performance comparison of data compression algorithms based on characteristics of sensory data in wireless sensor networks," in *Int. Conf. on Info. & Commu. for Embed. Syst. (ICICTES)*, Thailand, Jan 2010.
[8] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," in *Ambient Intelligence*. Springer Verlag, 2004.
[9] "Sentilla Company." [Online]. Available: http://www.sentilla.com/moteiv-transition.html
[10] SensorScope Project. [Online]. Available: http://sensorscope.epfl.ch
[11] Y. Zhu, R. Vedantham, S.-J. Park, and R. Sivakumar, "A scalable correlation aware aggregation strategy for wireless sensor networks," *Information Fusion*, vol. 9, pp. 354–369, 2008.
[12] E. Bodden, M. Clasen, and J. Kneis, "Arithmetic coding revealed-a guided tour from theory to praxis," Sable Research Group, McGill University, *Tech. Rep. Sable Technical Report No. 2007-5*, may 2007.
[13] K. Sayood, *Introduction to data compression*, 3rd ed.US: Morgan Kaufmann, 2006.
[14] F. Huang and Y. Liang, "Towards energy optimization in environmental wireless sensor networks for lossless and reliable data gathering," in *IEEE Int. Conf. on Mobile Adhoc and Sensor Systems (MASS)*, oct 2007, pp. 1–6.
[15] NIST/SEMATECH, *E-Handbook of Statistical Methods*. NIST: an agency of the U.S. Commerce Department, 6/23/2010. [Online]. Available: http://www.itl.nist.gov/div898/handbook/eda/section3/eda35.htm
[16] "SHTxx Sensor." [Online]. Available: http://www.sensirion.com/
[17] P. Levis and D. Gay, *TinyOS Programming*, 1sted. Cambridge University Press, Apr. 2009. [Online]. Available: http://www.worldcat.org/isbn/0521896061
[18] "Tinyos tuturial website." [Online]. Available: http://docs.tinyos.net/

Tossaporn Srisooskai was born in Nakhon Si Thammarat, Thailand on November 28. 1980. He received a B.Eng. degree in industrial engineering, from Burapha University, Thailand, in 2002.
In 2002-2005, he was Engineer at Pigeon Industries (Thailand) Company Limited, Thailand. In 2005-2009, he was Senior Engineer at Union Technology Company limited, which is currently Hitachi Global Storage Technologies (Thailand) Company Limited, the hard disk drive manufacturer. In 2010, he was Teaching Assistant at School of Information, Computer and Communication Technology, Sirindannhorn International Institute of Technology, Thammasat University. Thailand.
Currently, he is pursing the Master's degree in information and communication technology for Embedded System under TAIST-Tokyo Tech scholarship program at Department of Electrical Engineering, Faculty of Engineering, Kasetsart University, Thailand. This scholarship program is the cooperation of Tokyo Institute of Technology (Tokyo Tech), National Science and Technology Development Agency (NSTDA), Sirindhorn International Institute of Technology and Kasetsart University.

Kamol Kaemarungsi was born in Bangkok, Thailand in 1972. He graduated from the King Mongkut's Institute of Technology at Ladkrabang (KMIT'L) with a Bacheler's degree in electronics in 1994. He received a Master's degree in telecommunications from the University of Colorado at Boulder in 1999 and a Doctoral degree in information science from the University of Pittsburgh, Pennsylvania, USA in 2005.
Currently, he is a researcher in Embedded System Technology Laboratory at National Electronics and Computer Technology Center (NECTEC) under the National Science and Technology Development Agency (NSTDA), Thailand. His research interests include wireless sensor networks, location determination systems, and ultra-wideband systems.

Poonlap Lamsrichan was born in Chaiyaphume, Thailand on 29 July 1972. He received the B.Eng. (first class honor) and M.Eng. in Electrical Engineering, from Chulalongkorn University, Thailand, in 1994 and 1996, respectively.
In 1997-2002, he was a researcher at Chulalongkorn University. He received the Doctor of Engineering (Telecommunications) from Asian Institute of Technology (AIT) in 2006. Currently, he is an assistant professor at Electrical Engineering Department, Faculty of Engineering, Kasetsart University, Thailand.,

Kiyomichi Araki was born in Nagasaki, Japan on January 7. 1949. He received a B.E degree in electrical engineering from Saitama University, in 1971, and M.E. and Ph.D degree in physical electronics both from Tokyo Institute of Technology, Japan, in 1973 and 1978 respectively.
In 1973-1975, and 1978-1985, he was a Research Associate at the Tokyo Institute of Technology. In 1985-1995, he was an Associate Professor at Saitama University. In 1979-1980 and 1993-1994, he was a visiting research scholar at University of Texas, Austin and University of Illinois, Urbana, respectively. He is currently a Professor at Tokyo Institute of Technology. His research interests are information security, coding theory, communication theory, electromagnetic theory, circuit theory, and microwave circuits, etc.
Prof. Dr.Araki is a member of IEEE, IEICE, IEE of Japan and Information Processing Society of Japan. He served as a chair of Japan Chapter of IEEE MTT-S and a chair of Japan National committee of APMC in 2006.