Exploiting Data Correlation for Multi-Scale Processing in Sensor Networks

Xiaoning Cui[1, 2, 3] Baohua Zhao[1, 2] Qing Li[2, 3]

[1]Department of Computer Science and Technology, University of Science & Technology of China, Hefei, China
[2]Joint Research Lab of Excellence, CityU-USTC Advanced Research Institute, Suzhou, China
[3]Department of Computer Science, City University of Hong Kong, Hong Kong, China

cxning@mail.ustc.edu.cn bhzhao@ustc.edu.cn itqli@cityu.edu.hk

ABSTRACT

With the emergence of large and multi-scale sensor networks, the technologies of multi-scale processing among various sensors become an essential issue. In this paper, the problem of exploiting data correlation for multi-scale sensor networks is considered, and an architecture exploiting correlation is designed for both intra- and inter-data processing. Our correlation-adaptive scheme follows the characteristics of real sensor data, and fills the gap of the correlation models addressed by most of previous research with inherent support for related data gathering algorithms. A core solution module of this architecture is devised, and theoretical analysis and simulation studies are conducted on real-world datasets. Through the real-world data experiments in terms of accuracy and energy-consumption evaluation, the correlation-adaptive scheme is shown to work well in multi-scale processing sensor networks.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design- *Distributed networks*; C.3 [**Special-Purpose and Application-Based Systems**]: *Microprocessor/ microcomputer applications*; H.2.8 [**Database Management**]: Database Applications- *Data mining.*

General Terms

Design, Experimentation, Performance.

Keywords

Wireless sensor network, Multi-scale processing, Intra-/Inter-data correlation, Data sample, User query, Correlation exploiting architecture, Correlation adjustment function.

Infoscale'07, June 6-8, 2007, Suzhou, China

1. INTRODUCTION

The world is emerging as increasingly pervasive, and is challenged to provide services which integrate people, environment and knowledge. Pervasive computing seamlessly and invisibly pervades information and communication technologies into various environments and conditions, delivering services adapted to individuals and differing contexts of use [2]. The most essential characteristic of a pervasive system is the capability of sensing the physical world via a huge variety of sensors and controlling it through myriad of actuators. These services will have to be greatly based on contextual information and successful exploitation of local knowledge, while coping with highly dynamic environments and unpredictably changing resources. Therefore, the emerging "pervasive sensing world" will, from a networking point of view, face the key issue of efficiently handling sensor data in a multi-tasking environment, which we term as the multi-scale processing problem.

Compared with that of sensing data generation and synthesization, most previous research has focused on optimizing the energy consumption with the consideration of the limited battery supply. In reality, however, both data-centric nature and extreme energy restrictions should be perceived as two main characteristics of wireless sensor networks (WSNs). Based on the inherent attributes of data sensing and processing, in this paper we advocate the use of correlation analysis and data mining techniques to optimize both energy consumption and information accuracy.

The data-centric nature is significant for WSNs, as it is the data values that are concerned rather than the IDs of sensor nodes. Meanwhile, due to the rather limited resource constraint, the sensing and communication overheads of indirections must be avoided as much as possible. Thus, exploiting local data correlation becomes a necessary and efficient way to achieve long-lived sensor networks. Indeed, a WSN will not scale without making use of data correlation information: consider for example a network of n nodes, even with optimal routing, optimal power control, and optimal transmission schedules, the total transport capability scales like $O(n^{1/2})$, i.e., $O(1/n^{1/2})\rightarrow 0$ bit per node when n is large enough [11]. This indicates the great importance of in-network sensor data correlation processing. Moreover, as a sensor network grows large and sensor motes become heterogeneous (i.e., in a multi-scale processing environment), sensors with multiple tasks will obtain more efficient cooperation with the help of

mining miscellaneous (and even just potential) correlation among the data samples. In this paper, we

- describe a multi-scale processing environment in which an appropriate methodology is developed and analyzed for large scale sensor networks;
- design a correlation exploiting architecture for multi-scale processing in WSNs, and devise a core solution module for both simplified cases and real sensed data.

The rest of this paper is organized as follows. In Section 2 we first illustrate the multi-scale processing (MSP) environment of sensor networks, and then discuss the issues involved in MSP. In Section 3, the data correlation problem is presented, and a correlation exploiting architecture is designed with an analysis of a core solution module. The performance of the core solution module is evaluated in Section 4 for both simple case studies and real-world data experiments. Related work is reviewed in Section 5, and conclusions and future work are given in Section 6.

2. ISSUES OF MULTI-SCALE PROCESSING

In this section, the scenario description of multi-scale processing is first illustrated (Section 2.1), and then some applicable methods for multi-scale processing are discussed (Section 2.2) by considering the characteristics of sensor data.

2.1 Scenario Description

A multi-scale sensor network is shown in Figure 1, in which the wide area context is classified into several types, identified by different colors in the picture. Take a weather monitoring system as an example: the wide area context may contain the sun illumination angle, sky condition, canopy structure and etc., the integration of which constitutes complete information of the wide area context. Similarly, sensors with different colors are pre-configured for different functions (probably for different tasks). These sensors do their own sensing tasks as well as cooperating with others, i.e., in-network data fusion, to give a full view of the phenomena in the network.

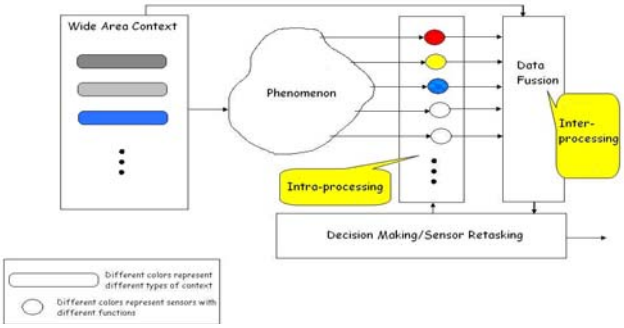

Figure 1. The structure of multi-scale processing

As shown in Figure 1, a sensor's own strategy is determined by the data from both itself (intra-data) and other types of neighbor sensors (inter-data). We aim at using numerical analysis and data mining techniques to deal with the two-step data processing in order to increase information accuracy and substantially save energy.

2.2 Issues Involved

Previous research reveals that sensor data streams exhibit, in addition to those outlined in [1], several key characteristics including inherent uncertainty, intrinsic intra- and inter-data correlation, extremely precious energy, and context-dependent data importance [14]. Concentrating on these key characteristics, the following issues must be addressed in designing a multi-scale sensor system.

Firstly, a sensor network is to give a reflection of the monitoring area. However, the real-world phenomena is consecutive while the data streams generated from sensor readings are discrete, probably with some noise such as sampling errors, faults and external attacks. Uncertainty is therefore inherent under local observation. Data stream modeling, event reconstruction, and noised sampling techniques are required to handle such data uncertainty.

Secondly, the event information captured by adjacent sensors is usually correlated due to the high redundancy of sensor deployment. Thus, different sensors observing the same object can share their information and eventually constitute an integrated view of the object. The intrinsic data correlation then lies in two aspects: intra-correlation (for the correlation of data samples generated by the sensor itself and/or its neighbor sensors with the same task) and inter-correlation (for the data samples generated by sensors with different tasks for correlation reference). Numerical analysis and de-correlation are the main correlated techniques.

Thirdly, energy is always a key concern in sensor networks, which is also influential to the quality of data sampling and in-network processing. The extreme energy constraint calls for methods such as prediction-based data sampling rate adjustment.

Lastly, data samples have different importance under different context. Take the air pollution index (API) by Equation (1) as an example:

$$\mathrm{API} = \max(I_1, I_2, ..., I_n) \tag{1}$$

where I_i (i=1,2,…,n) represents the concentration of various air contamination in the same area. The API indicates the air pollution level and it is harmful for people to live in an area where the value of API is more than 100. Given that there exists an air pollution monitoring sensor network in an area, the API sample values larger than 100 are more important than those smaller ones, and should be endowed with higher sampling rate and higher transmission priority. Decision making theory is required for sensing and communication schedules, especially for a multi-scale sensor network.

3. EXPLOITING DATA CORRELATION

As mentioned in Section 2, our work is to exploit, under application-specific requirements of information accuracy, the correlation of sensor data in order to save energy in multi-scale processing. In this section, we first formulate the data correlation problem in multi-scale sensor networks (Section 3.1), and then come up with the design of a correlation-exploiting architecture (Section 3.2). A core solution module of this architecture is finally devised and analyzed (Section 3.3).

3.1 Data Correlation Problem

We start the data correlation problem with a real-world scenario: the air pollution monitoring sensor network, in which the API-concentration relationship and API-grade relationship are pre-defined as shown in Table 1.

The table reflects theoretical correlation among different kinds of air-borne particulates. However, the concentration of a special kind may vary as the monitoring area changes, which cannot be precisely pre-configured in the correlation structure. Most previous researchers based their ideas on such a static and unchangeable correlation structure as Table 1, with little concern for runtime correlation refinement. In reality, however, the data streams are dynamic and uncertain, perhaps with some special and/or potential correlation which cannot hold accurate pre-definitions and have to be learned during sensor operations. Hence, we formulate the problem from the perspective of data correlation through Definitions 1 and 2.

Table 1. API, concentration, grade & state

TSP	SO_2	NO_x	I_n	Grade & State
1.000	2.620	0.940	500	V, Excellent
0.875	2.100	0.750	400	
0.625	1.600	0.565	300	
0.500	0.250	0.150	200	IV, Good
0.300	0.150	0.100	100	III, Normal
0.120	0.050	0.050	50	II, Not good
-	-	-	<50	I, Bad

Definition 1: *Given measurements $X_1, X_2, ..., X_N$ at time $T_1, T_2, ..., T_N$, the lag k autocorrelation function is defined as:*

$$R_x(k) = \sum_{i=1}^{N-k} X_i \cdot X_{i+k} \quad (2)$$

Although the time variable, T, is not used in the formula for autocorrelation, the assumption is that the observations are equi-paced.

Definition 2: *Given measurements $X_1, X_2, ..., X_N$ and $Y_1, Y_2, ..., Y_N$ at time $T_1, T_2, ..., T_N$, the lag k cross-correlation function is defined as:*

$$R_{xy}(k) = \sum_{i=1}^{N-k} X_i \cdot Y_{i+k} \quad (3)$$

The cross-correlation indicates the simultaneous change between two numerically valued random variables.

The data samples generated from sensors possess both autocorrelation and cross correlation. Consequently, the sensor data correlation problem is divided, separately, into two sub-problems of intra- and inter-data processing. We use Definitions 3 and 4 to clarify the two sub-problems.

Definition 3: *Given a sensor **a** and its homogeneous neighbor set $NS_a=\{sa_1, sa_1, sa_2,..., sa_m\}$,[ψ] the **intra-correlation** is defined as a combination of the autocorrelation of the data sensed by sensor **a** itself and the cross-correlation of the data generated from **a** and its neighbor nodes $sa_i,(i=1,2,...,m)$.*

[ψ] Here the homogeneous sensor nodes represent sensors with the same monitoring task. e.g., nodes sensing sun illumination are homogeneous.

The first sub-problem is thus concerning how to locally exploit intra-correlation to adjust its pre-defined correlation structure to specific circumstances.

Definition 4: *Given a sensor **a** and its heterogeneous neighbor set $ND_a=\{da_1, da_1, da_2,..., da_n\}$,[ζ] the **inter-correlation** is defined as the cross-correlation of the data generated from **a** and its neighbor nodes $ds_i,(i=1,2,...,n)$.*

The second sub-problem is thus concerning how to exploit inter-correlation in a multi-scale sensor network, to make use of the inter-correlation to elaborate the pre-configured correlation, and to eventually obtain an adaptable correlation structure for a specific context.

3.2 Correlation Exploiting Architecture

Our approach to solving the two sub-problems mentioned in Section 3.1 is by designing a correlation-exploiting architecture. In particular, we introduce in this subsection the intra- and inter-data processing modules, respectively.

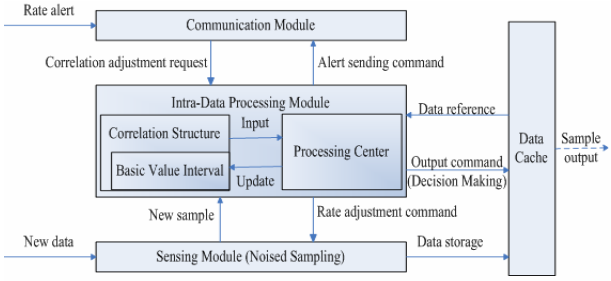

Figure 2. Intra-data processing module and data flow

The intra-data processing module is as illustrated in Figure 2. As shown in this figure, the intra-data processing module is to analyze *intra-correlation* for the sensing module and handle sampling rate alerts from the communication module. The dashed arrow on the right hand side means that the self-sampled data value may or may not be output, depending on the result of the decision-making process. For example, if current sample value is the same as the pre-sampled value or of little importance for describing the current context, the system needs not transmit this value to the sink node (so that the transmission energy can be saved).

The correlation structure (with pre-configured basic value interval) and the processing center are two main components of the intra-data processing module. The basic value interval is derived from statistical results and/or empirical estimation; it represents the criterion of sample values. If a sample value exceeds the lower or upper bound of the basic value interval, or the probability of a certain value is of a striking dissimilarity between the sampling results and that in the basic value interval, it probably implies an event happened in the area.

A prototype correlation structure with the basic value interval and some basic definitions of the data correlations is configured in priori, and sensors initially use this prototype as a reference to their work. As the wide area context changes, the prototype needs update and refinement in real time so that it can provide more

[ζ] Here the heterogeneous sensor nodes represent sensors with different monitoring tasks. e.g., sensor nodes recording temperature and recording humidity are heterogeneous ones.

accurate information for the sensing and communication module, and eventually optimize the entire data sampling process.

The main duty of the processing center (cf. Figure 2) is to manage and maintain the correlation structure. When a new sample comes, the value is first checked according to the basic value interval. Then the processing center makes a judgement with reference to current correlation structure and the sample's *intra-correlation*, including the *autocorrelation* with historical samples from the data cache and *cross-correlation* with neighbor samples, indicated by neighbors' alerts from the communication module., and correspondingly updates the correlation structure and adapts sampling strategy to the wide area context.

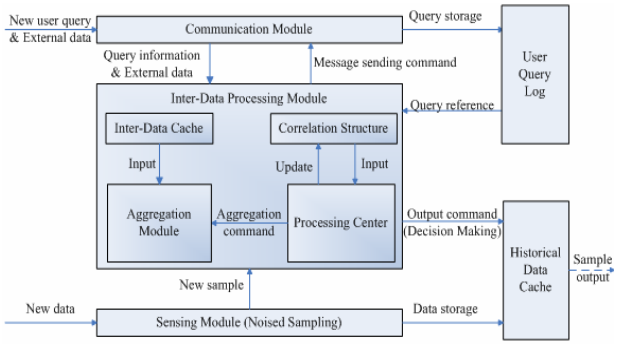

Figure 3. Inter-data processing module and data flow

The inter-processing module as shown in Figure 3 possesses a similar architecture. The dashed arrow means on the right hand side that the value may be output or saved in the historical data cache for future reference.

The real-world context is miscellaneous, so we theoretically assume that the context is described as a combination of various attributes, and heterogeneous sensors collect information related to separate attributes. For example, in a fire watching system, three types of sensors are deployed for sensing the temperature, the concentration of carbon dioxide, and the concentration of carbon monoxide, respectively. The three types of sensors constitute a multi-scale sensor network. Managing interactions among these multi-type sensors is thus the main task of the inter-data processing module.

As shown in Figure 3, the central module consists of an inter-data cache, a correlation structure, an embedded aggregation module, and the processing center. The inter-data cache stores relevant external samples from heterogeneous neighbors, based on the *inter-correlation* (represented by the correlation structure). The aggregation module deals with data fusion of correlated attributes. The processing center plays the role of analyzing user queries[ξ], updating the correlation structure, and managing data aggregation and sample output. As in [15], the user queries can be used as implication of potential and/or special correlation in special cases.

Through the 2-step processing modules, our ultimate goal is to optimize energy consumption as well as information accuracy for a network involving multi-typed sensors.

[ξ] A user query is a set of attributes queried together. e.g., "Attributes {*temperature*,*humidity*} in Area 2 at Time 3:00 pm" is a user query for *temperature* and *humidity* simultaneously.

3.3 Core Solution Module

We now proceed to analyzing, in detail, the functions of the processing center for intra- and inter-correlation (cf. Figures 2 and 3), based on a general algorithmic description.

The prototype correlation structure shown in Figure 4 is initially configured based upon empirical and statistical knowledge. With new data item and new query input, the correlation adjustment function runs according to real-time *intra-/inter-correlation* and updates correlation if necessary.

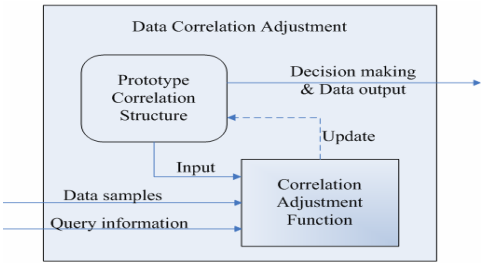

Figure 4. Framework of the core solution module

For intra-data processing, we assume that only a single attribute of the wide area context is sensed by a cluster of sensor nodes, leaving the discussion of multi-attribute sensing to the inter-data processing part. A general description of our correlation adjustment algorithm (Algorithm 1) is given below.

Algorithm 1: Intra-correlation adjustment

Input: raw data samples, represented by a set RawSmp
basic value interval, stored in an internal file BasicValue
intra-correlation structure, stored in a modifiable file
other homogeneous nodes' alert messages AlertMsg

Output: processed data samples, represented by a set OutSmp
alert messages to other homogeneous nodes, AlertMsg

Preconditions: sensors are identified by IDs
The capacity of sensor's data cache is fixed

1. For Sensor i's sensing module

```
(1) while  Retrieving a data sample smp from RawSmp
(2)        if smp belongs to BasicValue
(3)              Cache smp
(4)              Search smp.correlation and Calculate smp.state
(5)              if IsConsistence(smp.state, recent-data.state)
(6)                    Reduce data sampling rate
(7)                    Transmit OutSmp in a lower frequency
(8)                    Send AlertMsg(rateRdc, i)
(9)              else
(10)                   Increase data sampling rate
(11)                   Transmit OutSmp in a higher frequency
(12)                   Send AlertMsg(rateIncr, i)
(13)             endif
(14)       else
(15)               Send AlertMsg(exceedBasicValue, i)
(16)       endif
(17) endwhile
```

2. For Sensor i's communication module

```
(1) while Receiving an alert message from AlertMsg
(2)       on Receiving AlertMsg(rateRdc, j)
(3)             if not IsConsistence(smp.state, recent-data.state)
(4)                   Merge(smp.correlation,recent-date.correlation)
(5)                   Optionally go to sleep
(6)             endif
(7)       on Receiving AlertMsg(rateIncr,j)
(8)             if IsConsistence(smp.state, recent-data.state)
(9)                   Divide(smp.correlation, recent-data.correlation)
(10)            endif
(11)      on Receiving AlertMsg(exceedBasicValue,j)
(12)            Do user/application-specific operation
(13) endwhile
```

In Algorithm 1, the data streams are represented by a set called *RawSmp* since sensors can only capture discrete samples. As a sensor's intra-data processing center interacts with its sensing and

communication modules simultaneously, the algorithmic description is divided into two parts which actually work concurrently.

The first part in Algorithm 1 is for the interaction with the sensing module. When receiving a sample from *RawSmp*, the sample value is compared with the *BasicValue*. An eligible sample is then cached and its correlation is recorded as a reflection of current state of the context. The core function is illustrated in lines (5) ~ (16). If the current sample is consistent with recent ones stored in the historical data cache, indicating slight change during the period, the sensor reduces its sampling rate to save energy. Otherwise a rate increasing alert is sent to neighbor nodes because of a potentially sudden change in the sensing area. Since the sample value may be distorted for several reasons such as external events, noise, faults and attacks, the sample exceeding the *BasicValue* is a signal of these reasons and the operation in line (15) informs its neighbor nodes of the exception.

The interaction with the communication module is described in the second part of Algorithm 1. The functions *Merge* in line (4) and *Divide* in line (9) are the main operations for *intra-correlation*-based correlation adjustment. The function *Merge* combines data value intervals to indicate environmental similarity when neighbors' data exhibit a stable state while current sample and recent historical data are in different correlated value groups (denoting originally weak *intra-correlation*). The function *Divide* runs otherwise. When an alert message *exceedBasicValue* comes, user/application-specific operations are invoked since the reason for exceeding the basic value varies and the user/application requirements are different.

The *inter-correlation* adjustment is illustrated in Algorithm 2, where an *n-scale* sensor network is considered. The *n-scale* indicates the context represented by a combination of *n* correlated attributes. We assume that each attribute is sensed by a certain type of sensors, and sensors of different types have the ability to communicate with each other. The *n* types of nearby sensors coorperate, forming an *n-scale* sensor network.

In Algorithm 2, a sensor's *inter-correlation* structure is generally expressed by a vector $C = (c_1, c_2, ..., c_n)$, where c_{ij} represents the strength of the *cross-correlation* of attributes *i.attr* and *j.attr*. As the real-world context is rather complex, it is difficult to obtain an always-accurate description. User queries, as implication of the strength of *inter-correlation* [15], are quite useful in revealing potential/special *inter-correlation* and hence help adjust the correlation structure. Therefore, the function *CalculateProb* in line (5) calculates the query frequency of each correlated attribute and estimates the probability of each pair of attributes (*i.attr*, *j.attr*) for sensor *i*, the result of which is the basis of the application-specific function *Refine* in line (7). Lines (8) ~ (15) form the basic message handling module; a parameter *THRESHOLD* in this module helps determine the strength of the *inter-correlation* of a given attribute pair. For sensor *i* in Algorithm 2, if the refined *inter-correlation* of sensors *i* and *j* is strong enough, i.e., exceeding the pre-configured *THRESHOLD*, sensor *i*'s current sample is transmitted to sensor *j* on receiving *j*'s invitation. Alternatively, sensor *i* first aggregates the sample received from *j* with its own sample, and then forwards the aggregated data item to the destination. This cooperative scheme is beneficial to in-network processing, so that only the most useful data is transmitted to the most needed users. Since only those highly correlated attributes are aggregated, extra energy consumption of unnecessary aggregation and communication can be avoided.

Algorithm 2: Inter-correlation adjustment

Input: user queries, stored a set UserQuery
probabilities of inter-correlation between neighbors, modifiable inter-correlation structure, stored in a modifiable file
other heterogeneous nodes' invitations & external samples Msg

Output: aggregated samples represented by AggrSmp
invitation messages & internal samples represented by Msg

Preconditions: sensors are identified by the attributes they sense
The capacity of sensor's user query blog is fixed

For sensor node i with current sample smp

```
(1) while Receiving a user query q
(2)     Read (q)
(3)     Cache (q) into UserQuery
(4)     for q in UserQuery
(5)         CalculateProb (q.attr. i.attr)
(6)     endfor
(7)     Refine inter-correlation by probability
(8)     on Receiving external correlated sample from j
(9)         if CorrelationProb (i, j) within THRESHOLD
(10)            Aggregate (smp, j.smp)
(11)        endif
(12)    on Receiving invitation from j
(13)        if CorrelationProb (i, j) within THRESHOLD
(14)            Send (i.smp, i) to j
(15)        endif
(16) endwhile
```

4. PERFORMANCE EVALUATION AND CASE STUDIES

In this section, we evaluate the performance of the embedded correlation adjustment architecture according to accuracy and energy-consumption. First, we introduce the evaluation metrics in Section 4.1. We then consider in Section 4.2 a simplified version of our general problem, followed by experimental results and analysis on some real-world datasets.

4.1 Evaluation Metrics

Energy is a traditional metric in WSNs, based upon the proportion of energy consumption by transmission, receiving, sensing, and sleeping [3].

Accuracy, as defined by Definition 5, is another essential criterion.

Definition 5: *Given a data stream STREAM with a corresponding state set STATE = {(st_1, t_1), (st_2, t_2),…, (st_m, t_m)} and a discrete sample set SAMPLE = {(smp_1, t_1), (smp_2, t_2),…,(smp_n, t_n)}, $n \leqslant m$, the accuracy during time period $T=[t_1,t_m]$ is defined as*

$$accuracy(STREAM, SAMPLE, T) = \frac{num(EQUAL(STATE, SAMPLE.state))}{TOTAL(STATE)} \times 100\% \quad (4)$$

On the right hand of Formula (4), *EQUAL* is a boolean function, showing that at a certain time, if the state of *SAMPLE* is the same as corresponding *STATE*, the value of *EQUAL* is 1; 0 otherwise. *TOTAL* is the total amount of states during *T*, calculated at each sampling point.

4.2 Case Studies

As shown in Figure 4, data samples and user queries are the main sources for correlation adjustment. In the following discussions, we consider a simplified intra-data processing example (Case 1) and a fundamental inter-data processing example (Case 2) of our general problem, in which the two sources are discussed separately.

Case 1: For intra-data processing in Sensor i

Assumptions:
The values of data samples are dynamically changed during some period
Data streams are sampled without noise
Omit query consideration here

Goal: Capture as many state changes as possible with energy consideration.

Settings:
Correlation information: correlation intervals
Correlation adjustment function F
Basic value interval [b_1, b_2]

Estimated by accuracy and energy consumption

Scenario:
20 consecutive integral raw data samples
Basic value interval [0, 9]
Initial correlation intervals [1,5], [6,9]. indicating two states of the context

Rules:
(1) If 3 or more consecutive data items in the same correlation interval occurred, reduce data sampling rate.
(2) If 2 consecutive data items are in different correlation intervals, increase data sampling rate.
(3) Output the alert of data dynamic change when increasing sampling rate continuously. (optional)
(4) If reaching the fastest/slowest sampling rate when increasing/reducing sampling rate, output alert message. (optional)
(5) Follow the rules above after rate alteration.

Adjustment: Follow the adjustment methods in Algorithm 1

In our experiments, the raw data, sensed samples, and output results are given in Table 2 and illustrated in Figure 5. Table 2 shows the sample values and actions to increase or reduce sampling rate adaptively. Notice that an alert is sent at time 3 when the sensor achieves the fastest sampling rate; an even faster rate is needed, however, according to rules (2) and (4) of Case 1 above.

Table 2. Intra-operation table for Case 1

Time	1	2	3	4	5	6	7	8	9	10
Raw	1	2	7	8	7	7	6	7	8	9
Sense	1	2	7	8	7		6		8	
Out	1	2	7	8	7				8	
Action			↑		↓				↓	
Alert			√							
Time	11	12	13	14	15	16	17	18	19	20
Raw	3	5	7	6	6	7	9	7	8	1
Sense		5		6	6	7		7		
Out		5		6	6	7		7		
Action		↑		↑		↓		↓		
Alert										

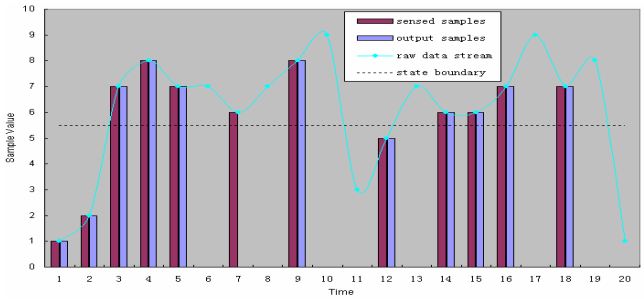

Figure 5. Comparison among raw data and processed results

In Figure 5, the dotted line in the middle is the state boundary which divides the context into two states. The sensed samples constitute an approximation to the raw data stream, and the output samples are a subset of the sensed samples chosen according to the rules of Case 1.

There are 20 samples during the period of time 0 to 20. With an assumption that data streams are sampled without noise, the theoretical accuracy can, following Definition 5, simply be calculated formulas follows:

$$accuracy(raw, sense, t) = \frac{num(EQUAL(raw.state, sense.state), t)}{TOTAL(raw.state, t)} \times 100\% \quad (5)$$

where the parameters *raw* and *sense* represent raw samples of the original data stream and sensed samples, respectively. t = 20 in Case 1. Hence, the accuracy for Case 1 is *18/20*100%=90%*. The errors occur at time 11 and time 20. It is obvious that the discrete samples approximately fit the original data stream and illustrate the trend of state transition.

The correlation interval adjustment follows the methods of Algorithm 1. Suppose an external sampling rate increasing alert (revealing an emergent state transition from neighbors) happens at time 11 when Sensor *i* is still at a low sampling rate. Sensor *i* got a value 3, refers to its recent historical data item value 8, and refines the correlation interval from [1, 5] to [1, 3], [3, 5], and from [6, 9] to [6, 8], [8, 9], depicting the context with more detailed states. This correlation partition calls for Sensor *i*'s attention to the sample value intervals [3, 5] and [6, 8], which is helpful for future sampling rate adjustment. Correlation interval union happens in just the opposite condition, helping save energy under some stable conditions.

Such operations are adaptive and work well even for real-world datasets, as shown in Figures 6, 7 and 8. The daily precipitation datasets in [13] are used to demonstrate the practical capability of our intra-solution module. Figure 6 shows a comparison of original precipitation data stream and output samples with run-time correlation adjustment. These discrete samples fit the real data stream nicely in most of the time, although distortion occurs from time 43 to 48 due to some correlation partition/union errors during the process of correlation update. In Figure 7 (where negative values represent no sensing/output happens at that time), the samples without correlation adjustment have missed several key points such as data items at time 9 and 10, while the samples with correlation adjustment perform much better. The energy cost with different initial correlation configurations are analyzed in Figure 8, indicating a robust energy consumption pattern of the intra-data processing module.

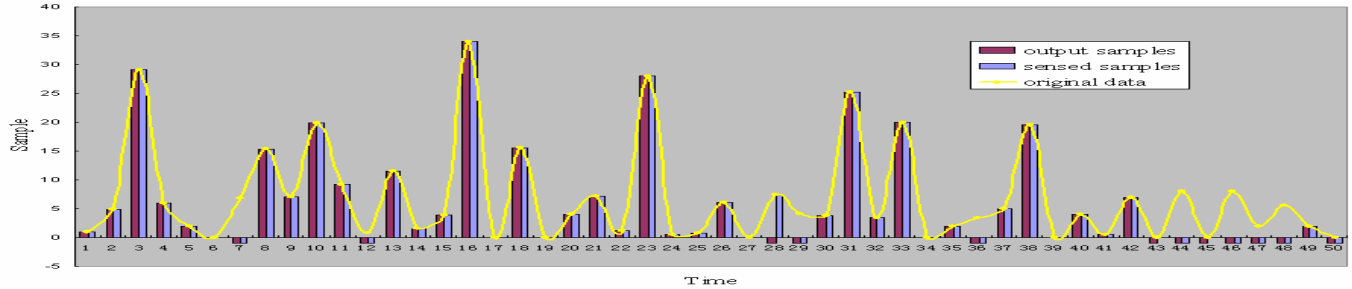

Figure 6. Comparison of original sensed and output samples for the precipitation dataset of Aug. 1st, 1998 in China, with initial correlation interval 10 and maximum sampling rate four times of the minimum sampling rate

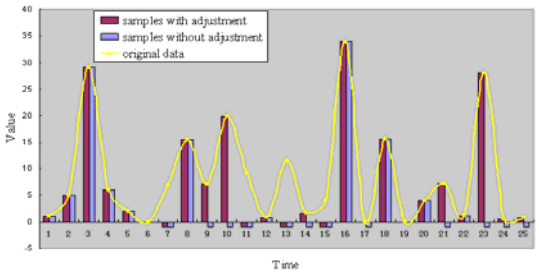

Figure 7. Data processing with/without correlation adjustment

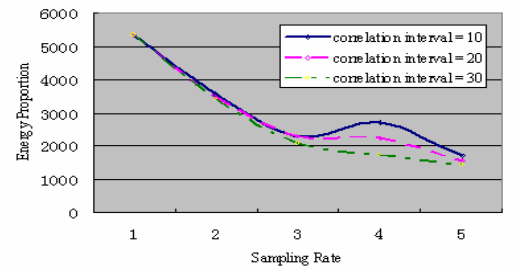

Figure 8. The energy cost with different initial correlation intervals and sampling rates

As to the more general situations, Case 2 depicts the inter-correlation adjustment in our architecture.

Case 2: For inter-data processing in Sensor i

Assumptions:
Data samples are relatively consistent during a certain period, so intra-data processing is omitted here
Data streams are samples without noise

Settings:
Correlation information: boolean matrix C
c_{ij}=1, correlated; c_{ij}=0, irrelevant
Wide area context: an attribute item set A
Correlation adjustment function F
|A| types of sensors in the network

Estimated by probability

Scenario:
3 attributes, 5 continuous queries
An apriori correlation matrix C_{3*3}

Rules & Adjustment:
(1) Threshold = 50%
(2) If probability (i, j) >= 50%, adjust correlation set c_{ij}=1, make more communication with Sensor j.
(3) If probability (i, j) < 50%, adjust correlation set c_{ij}=0, reduce communication with Sensor j.
(4) Send self-sample to j on receiving j's invitation and c_{ij}=1.
(5) Aggregate j's sample with self-sample on receiving j's sample and c_{ij}=1.

We use the notation S_{a1} to denote a sensor for attribute a_1, S_{a2} for attribute a_2, and so on. For a 3-tuple $A = \{a_1, a_2, a_3\}$ and its apriori correlation array C_3= (1, 0, 1), Table 3 records the *QueryLog* of the actual user queries received over a period of time.

It is clear that a_1 and a_2 are simultaneously queried with a frequency of 60%; in contrast, no query occurred for a_1 and a_3 simultaneously according to the recent consecutive queries. So $S_{a1}.c_2$ is changed to 1 and $S_{a1}.c_3$ to 0 based on Rules (2) and (3). Subsequently, Sensor S_{a1} keeps more interaction for attribute a_2 while less for attribute a_3. For instance, if a sample of attribute a_2 comes into Sensor S_{a1}, the sample will be first aggregated with Sensor S_{a1}'s local sample before getting forwarded, so that higher transmission efficiency is achieved. The experiment runs in a similar way, the details of which are omitted here due to space limit.

Table 3. User queries for Case 2

Time	1	2	3	4	5
Query	a1,a2	a1	a3	a1,a2	a1,a2

For the sake of simple expression, we have only discussed and simulated the case in which up to three correlated attributes are requested at a time. However, our proposed methods can be directly applied to cases in which more than three attributes are requested at a time. As there are few formalized methods aiming at evaluating the performance of interactions among multiple sensors, it is part of our future work to develop such applicable estimating methods.

5. RELATED WORK

The issue of designing a new architecture to exploit data correlation in sensor networks was initially addressed by Ganesan et al. [9], where the multi-resolution of data storage, distribution, and adaptation to correlation was analyzed and a hierarchical data handling architecture was presented. The characteristics of sensor data streams were researched in [1, 14], and relevant data models were developed in [8, 14]. A series of data gathering algorithms were devised to address such issues as modeling spatial-temporal correlation [7, 12], using hierarchical architecture[9], optimizing data coding [6, 16], constructing minimum spanning tree, and finding dominating sets [10]. These data gathering strategies focused mostly on saving energy consumption in routing. However, the high cost of accurate sample acquisition was often ignored in those strategies. Recently, much attention has been paid to giving a high-resolution by exploiting correlation in data gathering (e.g., [5]), but, to the best of our knowledge, most of the works were based on a static and unchangeable correlation structure [7], which cannot precisely reflect the dynamic nature of the real-world context. Even a few dynamic models, such as [4], concentrated mainly on centralized correlation processing at the sink node and depended on highly synchronized update on both the sensor network and the sink node.

Theoretically, achieving optimality usually requires global knowledge of real-time sensor data, which may not be available in practice. Our work of designing a locally correlation-adaptive architecture for intra- and inter-data processing can be, as a matter of fact, regarded as complementary to the data aggregation approaches mentioned above.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a correlation-adaptive scheme which, on the one hand, differs from the static and unchangeable correlation structure in most of the previous research, and on the other hand, matches the real sensor data well. In our approach, the problem of data correlation exploitation for multi-scale sensor networks has been considered for both intra- and inter-data processing. Inside the correlation exploiting architecture, a core solution module is devised whose performance is analyzed upon real-world datasets. Our scheme is also complementary and applicable to most existing data gathering algorithms.

Currently, we are in the stage of continuing the development and experiments of our system, with detailed performance results of

the inter-correlation processing to be reported by our subsequent research. In addition, we plan to further investigate the relationships between real-world data samples and various sensor properties, so as to be able to deploy the correlation adjustment scheme in an application-specific way.

7. ACKNOWLEDGEMENTS

The research described here is supported by the National Basic Research Fund of China ("973" Program) under Grant No.2003CB317006, and has been benefited from various discussions among the group members of the Joint Research Lab between CityU and USTC in their advanced research institute in Suzhou, China, particularly Miss Zhi Hui HU, Mr. An LIU, Hai LIU, and Bao Ping LIN.

8. REFERENCES

[1] Babcok, B., and Babu, S., etc. Models and issues in data stream systems. *PODS*, 2002, 1-16.

[2] Carreras, I., Miorandi, D., and Chlamtac, I. A framework for oppotunistic forwarding in disconnected ad hoc networks. *Proc. of MOBIQUITOUS*, San Jose, CA, 2006.

[3] Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *MOBICOM*, 2001, 85-96.

[4] Chu, D., Deshpande, A., Hellerstein, J. M., and Hong W.,Approximate data collection in sensor networks using probabilistic models. *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, Atlanta, Georgia, USA, 2006.

[5] Cristescu, R., and Begerull-Lozano, B. Lossy network correlated data gathering with high-resolution coding. *IEEE Transactions on Information Theory*, *52*, 6(Jun. 2006), 2817-2824.

[6] Cristescu, R., Beferull-Lozano, B., Vetterli, M., and Wattenhofer, R. Network correlated data gathering with explicit communication: NP-completeness and algorithms. *IEEE/ACM Transactions on Networking, 14*, 1(Feb. 2006), 41-54.

[7] Cristescu R., and Vetterli, M. On the optimal density for real-time data gathering of spatial-temporal processes in sensor networks. *IPSN*. 2005, 159-164.

[8] Dekhtyar, A., Ross, R., and Subrahmanian, V. Probabilistic temporal databases, I: algebra. *ACM TODS*, *26*, 1(Mar. 2001), 41-95.

[9] Ganesan, D., Estrin, D., and Heidemann, J. DIMENSIONS: why do we need a new data handling architecture for sensor networks?. *ACM SIGCOMM Computer Communications Review*, *33*, 1(Jan. 2003), 143-148.

[10] Gupta, H., Navda, V., Das, S. R., and Chowdhary, V. Efficient gathering of correlated data in sensor networks. *MobiHoc'05*, Urbana-Champaign, Illinois, USA. 2005, 402-413.

[11] Gupta, P., and Kumar, P. R. The capacity of wireless networks. *IEEE Transactions on Information Theory*, *46*, 2(2000), 388-404.

[12] Guestrin, C., Bodik, P., Thibaux, R., Paskin, M., and Madden, S. Distributed regression: an efficient framework for modeling sensor network data. *(IPSN'04)*. Berkeley, California, USA. 2004, 1-10.

[13] http://ncc.cma.gov.cn/ex-docs/ibm_hubexnavigate.htm

[14] Liu, H. Y., Hwang, S. Y., and Srivastava, J. PSRA: a data model for managing data in sensor neworks. *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, (Jun. 2006), 540-547.

[15] Takahiro, H., Norishige, M., and Shojiro, N. Replica allocation for correlated data items in ad hoc sensor networks. *SIGMOD Record*, 33, 1(Mar. 2004), 38- 43.

[16] Von Rickenbach, P., and Wattenhofer, R. Gathering correlated data in sensor networks. *DIALMPOMC'04*, Philadelphia, Pennsylvania, USA, 2004, 60-66.