ISSN 1726-5479

SENSORS & TRANSDUCERS

3/12
vol. 14-2
Special

Physical and Chemical Sensors & Wireless Sensor Networks

International Frequency Sensor Association Publishing

IFSA

Sensors & Transducers

Volume 14-2, Special Issue
March 2012

www.sensorsportal.com

ISSN 1726-5479

Sensors & Transducers Journal (ISSN 1726-5479) is a peer review international journal published monthly online by International Frequency Sensor Association (IFSA).
Available in electronic and on CD.

Sensors & Transducers Journal

Contents

Volume 14-2
Special Issue
March 2012

www.sensorsportal.com

ISSN 1726-5479

Research Articles

Authors are encouraged to submit article in MS Word (doc) and Acrobat (pdf) formats by e-mail: editor@sensorsportal.com
Please visit journal's webpage with preparation instructions: http://www.sensorsportal.com/HTML/DIGEST/Submition.htm

International Frequency Sensor Association (IFSA).

The 6th International Conference on Sensor Technologies and Applications

SENSORCOMM 2012

19 - 24 August 2012 - Rome, Italy

Deadline for papers: 5 April 2012

Tracks: Architectures, protocols and algorithms of sensor networks - Energy, management and control of sensor networks - Resource allocation, services, QoS and fault tolerance in sensor networks - Performance, simulation and modelling of sensor networks - Security and monitoring of sensor networks - Sensor circuits and sensor devices - Radio issues in wireless sensor networks - Software, applications and programming of sensor networks - Data allocation and information in sensor networks - Deployments and implementations of sensor networks - Under water sensors and systems - Energy optimization in wireless sensor networks

http://www.iaria.org/conferences2012/SENSORCOMM12.html

The 3rd International Conference on Sensor Device Technologies and Applications

SENSORDEVICES 2012

19 - 24 August 2012 - Rome, Italy

Deadline for papers: 5 April 2012

Tracks: Sensor devices - Ultrasonic and Piezosensors - Photonics - Infrared - Geosensors - Sensor device technologies - Sensors signal conditioning and interfacing circuits - Medical devices and sensors applications - Sensors domain-oriented devices, technologies, and applications - Sensor-based localization and tracking technologies

http://www.iaria.org/conferences2012/SENSORDEVICES12.html

The 5th International Conference on Advances in Circuits, Electronics and Micro-electronics

CENICS 2012

19 - 24 August 2012 - Rome, Italy

Deadline for papers: 5 April 2012

Tracks: Semiconductors and applications - Design, models and languages - Signal processing circuits - Arithmetic computational circuits - Microelectronics - Electronics technologies - Special circuits - Consumer electronics - Application-oriented electronics

http://www.iaria.org/conferences2012/CENICS12.html

Sensors & Transducers Journal, Vol. 14-2, Special Issue, February 2012, pp. 64-78

Sensors & Transducers

ISSN 1726-5479

http://www.sensorsportal.com

Energy Efficient in-Sensor Data Cleaning for Mining Frequent Itemsets

Jacques M. BAHI, Abdallah MAKHOUL, * Maguy MEDLEJ
Computer Science Laboratory (LIFC), University of Franche-Comté,
Rue Engel-Gros, 90016, Belfort, France
E-mail: {jacques.bahi, abdallah.makhoul, maguy.medlej}@univ-fcomte.fr

Received: 7 November 2011 /Accepted: 20 December 2011 /Published: 12 March 2012

Abstract: Limited energy, storage, computational power represent the main constraint of sensor networks. Development of algorithms that take into consideration this extremely demanding and constrained environment of sensor networks became a major challenge. Communicating messages over a sensor network consume far more energy than processing it and mining sensors data should respect the characteristics of sensor networks in terms of energy and computation constraints, network dynamics, and faults. This lead us to think of a data cleaning pre processing phase to reduce the packet size transmitted and prepare the data for an efficient and scalable data mining. This paper introduces a tree-based bi-level periodic data cleaning approach implemented on both the source node and the aggregator levels. Our contribution in this paper is two folds. First we look on a periodic basis at each data measured and periodically clean it while taking into consideration the number of occurrences of the measures captured which we shall call weight. Then, a data cleaning is performed between groups of nodes on the level of the aggregator, which contains lists of measures along with their weights. The quality of the information should be preserved during the in-network transmission through the weight of each measure captured by the sensors. This weight will constitute the key optimization of the frequent pattern tree. The result set will constitute a perfect training set to mine without higher CPU consumption allowing us to send only the useful information to the sink. The experimental results show the effectiveness of this technique in terms of energy efficiency and quality of the information by focusing on a periodical data cleaning while taking into consideration the weight of the data captured.

Keywords: Sensor networks, Periodic data aggregation, Tree based algorithms, Quality of information, Training set to mine, Frequent Pattern tree (FP_Tree).

---

1. Introduction

Information collection and prediction became an eminent demand nowadays for controlling and predicting natural disasters, preventing failures, improving food production, and improving human well being. Scientific researchers are focusing on monitoring large range of application areas especially in where power or infrastructure limitations make a wired solution costly, challenging or even impossible. Latest researches show that this task is dedicated to spatially distributed autonomous devices. Such devices, or nodes, combined with routers and a gateway constitute a wireless sensor network (WSN). Wireless sensor networks are composed of large distributed number of sensor nodes; each sensor node has a separate sensing, processing, storage, and communication unit. The sensing unit is responsible for gathering data from its environment whereas the processing unit in the form of a microprocessor manages the tasks. Memory is used to store temporary data or data generated during processing. The communication unit communicates with the environment. The distributed measurement nodes communicate wirelessly to a central gateway, which provides a connection to the wired world where collecting, processing, analyzing, and presenting of measurement data is needed. Each sensor collects a considerable amount of raw data which are sent periodically to a central sink (gateway). Each sensor node is powered by a battery. However, sensor nodes are tightly limited in terms of battery, power and memory storage. Thus they are not expected to carry huge amount of data or complex computations. It is important to highlight that a sensor network usually consists of thousands or ten thousands of nodes deployed redundantly in order to ensure reliability and where each single sensor is expected to cooperate with other sensors to provide service. Subsequently, the collected data is partially redundant and is subject to aggregation offering.

The major challenge in a wireless sensor network is improving the lifetime of the network in other word managing efficiently the battery and power consumption. Recent researches focused on such task as it is difficult and cost ineffective to recharge the battery. Energy is mainly consumed during data transmission from the source node to the sink (gateway) making network data transmission one of the core issues to address by reducing energy consumption within the wireless sensor network. Furthermore, data accuracy is another main design concern in wireless sensor networks. In order to avoid any faulty alarms, the distributed measurements nodes communicate wirelessly to a central gateway, providing “interaction between people or computers and surrounding environment” [3]. To achieve data accuracy we strongly believe that only the right information should be communicated through the wireless sensor networks. The authors intrigued by such interesting challenge suggest through this article a multilevel data cleaning and optimized mining algorithms.

This article introduces a periodic multilevel data cleaning algorithm aiming to optimize the volume of data transmitted thus saving energy consumption and reducing bandwidth on the network level. Instead of sending each sensor node’s raw data to a base station, the data is cleaned periodically at the first level of the sensor node then another “data aggregator “ sensor node collects the information from its associated nodes. We shall call this first level “in-sensor process periodic cleaning” approach. A second cleaning algorithm is applied on the level of the aggregator node itself. It is important to note that the weight of each measure (number of occurrences of each measure in the set) is preserved through both described above techniques thus preserving the quality of information provided by each measure. The described approach pioneers in the field of focusing on a periodical data cleaning while taking into consideration the weight of the data captured will result with a cleaned training set of data. This phase will be the pre processing step to get the perfect data set to be mined in an acceptable timeframe.

Data mining is a perfect tool to analyze data, categorize it and summarize the relationships identified. Our approach will adopt this tool on the level of the aggregator to send only useful information to the base station. The weight collected from the first step will constitute the optimization key of a data mining algorithm (FP_Tree). The FP_Tree is a prefix tree for storing crucial and compressed information about support information and is constructed after scanning twice the database. Our approach avoid the scanning by working on a different structure resulted from the output of the data cleaning algorithm applied on the level of the aggregator.

The rest of the paper is organized as follows: The first section presents and accredits data cleaning and aggregation related work and research. While the second introduces our data aggregation scheme: the first level periodic data cleaning algorithm applied on the sensor node level. At the second level, a heuristic method aiming to clean data on the aggregator level and index the data cleaned by a weight significant of its redundancy and quality is introduced. The Third section describes the data mining technique adapted on the level of the aggregator to extract the useful information to be sent to the sink. The fourth section shows the experimental results of our suggested multi cleaning algorithm and its contribution to the network life through optimizing energy consumption. We conclude by emphasizing the added value of our approach and its contribution to the world of wireless sensor network research.

2. Related Work

Limited battery power and high transmission cost in wireless sensor networks make in-network cleaning and aggregation a challenging area for research. Data transmission is the most costly operation in sensors [1], compared with it, the energy cost of in-network computation is trivial and negligible. Reducing the number of packets being transmitted in the network will eventually lead to energy consumption reduction. In order to reduce the number of packets, data cleaning and data aggregation related approaches have been conducted. Based on this, we have presented some network data reduction and aggregation related works that fall into different levels of data cleaning approach: In-network approach between hops or on the level of the sensor itself where each sensor takes up some computation according to the applications (e.g., query processing, data collection, event detection, and so on). Several performance measures like network lifetime, data accuracy, false alarm, high data redundancy, latency and scalability need to be considered concurrently [4, 5]. Zhuang and Chen Hong Kong [2] focuse on the outliers cleaning within multi-hops by including wavelet based outlier correction and neighboring DTW (Dynamic Time Warping) distance-based outlier removal. The cleaning process is accomplished during multi-hop data forwarding process, and made use of the neighboring relation in the hop-count based routing algorithm. On the other hand, data aggregation methods in sensor networks have been reported [11]. Zheng, Chen, and Qiu [12] propose a method to build an aggregation tree model in WSN such that the captured data are aggregated along the route from the leaf cells to the root of the tree. In this scheme, the tree is not built directly on sensors, but on the non overlapping cells which, are divided with equal sizes in the target terrain. A representative sensor in each cell acts in name of the whole cell, including forwarding and aggregation of the sensing data in its cell and the receiving data from the neighbor cells. In light of large-scale and high-density sensor nodes, the scheme cuts down the data transmission overhead from three aspects. Firstly, primary aggregation should be conducted in the cell, based on the observation that the measurement data in one small cell are almost identical. Secondly, aggregation operation in one large-scale network should be directed to avoid the dynamic change of aggregation topology. Finally, using cell-by-cell communication instead of hop-by-hop communication reduces the density of communication and the complexity of the aggregation topology in the network. Greedy aggregation is proposed in [9, 11], where a tree is constructed to indicate the path from each sensor node to the sink. The shortest path linking a node to the sink is used as the initialization of the tree. Then, the shortest paths linking the remaining nodes to the current tree will be incrementally added to enlarge the tree. With this technique, the packets will be aggregated as early as possible and the aggregated packet will be directly routed back to the sink. However, the efficiency of the greedy incremental method is entirely determined by the shortest path. The data transmission is not reliable since once the path is broken, a large region will be disconnected and will not be able to send information to the sink.

All the presented work didn't take into consideration the accuracy of the information affected by the number of similarity between measures. In this paper we shall focus on periodic data collection at the first level of sensor nodes. We consider that at each determined time interval the sensing unit is configured to capture measurements. At this level, our approach consists of comparing measurement captured at an interval of time t with measurements already captured at a previous interval in order to perform some in-sensor processing and evaluate data. We shall call this in-sensor process periodic data cleaning approach. Our aim is to periodically clean the data captured from noisy and redundant measures while maintaining an acceptable level of quality and accuracy of the information that is deduced from the captured measures. The measures' occurrences are called weighted measures in this article and will serve as a parameter passed to all data cleaning levels and subsequently saving accuracy of the purged data. When applying the suggested algorithm, it cleans periodically the data while assigning to each measure its proper weight. The cleaning will be processed in two steps: the source node will constitute the first step whereas a special sensor node called aggregator receiving the data from different source nodes will conduct data cleaning at the second step. Such scheme will form an optimized training set for the classifier, predicting with reasonable accuracy the class of each instance fed. However, in Data Mining the task of finding frequent pattern in large databases is very important but is computationally expensive, especially when a large number of patterns exist. Han & al. [16] propose an FP growth algorithm, which uses an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP_Tree). In his study, Han proved that his method outperforms other popular methods for mining frequent patterns, e.g. the Apriori Algorithm [16] and the TreeProjection [17]. However the FP growth is expensive to built in terms of CPU and time. Our aim is to use the weight collected from the first step as the optimization key for the generation of frequent pattern based on the structure generated on the level of the aggregator.

3. Data Aggregation Schema

This section gives the main definitions and notations, together with our approach that will be used for an efficient and accurate in sensor nodes data reduction. The main focus is the periodic data collection where each sensor takes measurement at regular time interval. We classify our approach as 2 tiers data cleaning approach: the source node will constitute the first tier whereas a special sensor node called aggregator receiving the data from different source nodes will be the subject of data cleaning at the second tier. Fig. 1 illustrates our tree based data aggregation scheme. At the first tier the source nodes exist. The second tier contains the aggregator.

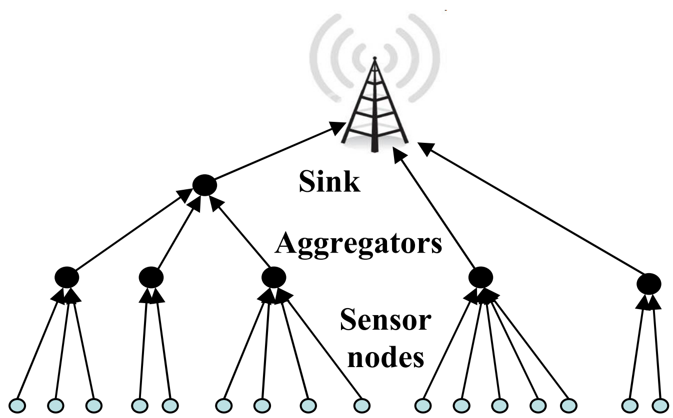

Fig. 1. Tree based data aggregation scheme.

3.1. Definitions and Notations

The set of sensor nodes is denoted by N = {1, 2, …, n}, where n is the number of nodes. Each node is composed of many sensors S that produce a measurable response to a change in a physical condition like temperature or pressure or humidity, etc. Each sensor node takes a vector of measurements M[t] = $(m[t_1],\ldots m[t_{\Pi-1}])\in\Re^{\Pi}$ at regular time interval t during a period ∏. The unit time is called slot, whose length is the time interval between two measurements. After ∏-1 slots, each sensor node Ni will have a vector of measurements Mi as follows:

$$
\begin{array}{c}
M_1 \\ M_2 \\ M_3 \\ \\ \\ M_s
\end{array}
\begin{bmatrix}
m_1[t_1] & m_1[t_2] & \ldots\ldots & m_1[t_{\pi-1}] \\
m_2[t_1] & m_2[t_2] & \ldots\ldots & m_2[t_{\pi-1}] \\
m_3[t_1] & m_3[t_2] & \ldots\ldots & m_3[t_{\pi-1}] \\
 & & & \\
 & & & \\
m_s[t_1] & m_s[t_2] & \ldots\ldots & m_s[t_{\pi-1}]
\end{bmatrix}
$$

3.1.1. Definition1: Substitution between Two Measures

At each interval I and for each sensor s, we associate to each measure $m_s[t_i]$ a function noted Substitution($m[t_i]$, $m[t_j]$) which, define a kind of similarity to unify or not with a measure $m_s[t_j]$ taken a time t_j / j< i.

$$
\mathrm{Substitution}(m[t_i], m[t_j]) = \begin{cases} 1 & \text{if } \|m[t_i] - m[t_j]\| \leq \delta \\ 0 & \text{otherwise.} \end{cases} \tag{1}
$$

where δ is a threshold fixed by the application.

3.1.2. Definition2: Weight of a Measure

Intuitively, redundancy gives more importance to some information which, are represented by many features and may occult less than others that are less present. We define weight of the measure m at a time t the total number of measures captured after the time t and can be unified with m.

$$
\mathrm{Weight}(m[t_i]) = \Sigma^{\Pi-1}{}_{j=t_{i+1}}\mathrm{Substitution}(m[t_i], m[t_j]) \tag{2}
$$

3.1.3. Definition3: Cell's Measure

On the level of an aggregator A, we define a cell's measure cell Ca[i] = A[i](λ_i, m_i) such that the cell contains the received measure mi from the node n_i with its corresponding weight λ_i. A cell Ca[i] is built based on distinct measures and weights existing in the aggregator A. we refer to Ca[i](m) as the measure in a cell while the respective weight is Ca[i](λ).

3.2. First Tier: Periodic data Cleaning

The proposed method calculates on periodic basis the substitution function between the measures already captured and the current measure captured at the current period. If the substitution is equal to 1, which, means that the new measure can be unified with the existing measure on which we are performing the substitution function, the weight of the existing measure is incremented by 1 and the new measure is disregarded. For sake of simplicity and without loss of generality, Algorithm 1 illustrates the first tier with one type of measure. At the end of this algorithm, no redundant measures will exist. Each sensor will send to the aggregator a set of reduced measures associated to their corresponding weight and ready for the second tier data cleaning algorithm.

Algorithm 1. First Tier Data Cleaning.

Input:
New measure $m[t_i]$.
Output:
Reduced set of measurements M.
Initialization
Get first measure $m[t_1]$
$\lambda_1 \leftarrow 0$
For each slot t_i during a period $\prod$ **do**
Get a measure $m[t_i]$
For each existing measure $m[t_j]$ **do**
    If Substitution $(m[t_i], m[t_j]) = 1$ **then**
        $\lambda_j \leftarrow \lambda_j+1$ // λ_j is the weight($m[t_j]$)
        Disregard
    Else $\lambda_i \leftarrow \lambda_i+1$ //λ_i is the weight($m[t_i]$)
        Add $m[t_i]$ to M: $M \leftarrow \{(M \cup (\lambda_i, m[t_i])\}$
        End if
    End for
End For

3.3. Second Tier: Weighted Data Cleaning

We define a special node for each set of nodes which, we shall call "aggregator", such aggregator will receive data from its set of nodes. We assume that the aggregator is more powerful than its set of nodes N. At this stage each aggregator will hold n lists for each type of measurement where n is the number of nodes associated to this aggregator and each list contains measures with their related weights.

Our approach aims at reducing data transmitted from the aggregators to the sink subsequently reducing energy consumption. The obvious idea will suggest looping each list comparing its measures with the remaining lists looking for redundant data. Such approach proved to be costly in terms of data processing since it will scan the whole existing set many times and is attributed a complexity of O(n!). Our approach, illustrated in Algorithm 2, suggests building progressively a dynamic array list as follows:

We define A = Union of all existing lists in the aggregator: $A = (\cup(\lambda_i, m[t_i])|i \in N)$. Then we select a random measure with its related weight from A in order to create the first cell of our dynamic array list by placing the above random value in it. We continue by selecting value (λ_i, m_i) from A and calculating the Substitution function for each selected value m_i with the array list values $\{ (\lambda_j, m_j),$

$j \in$array list values}. The first measure m_j answering Substitution (m_i, m_j) = 1 is observed and the weight of matched values are added. If no match occurs the value is added to the dynamic array list by creating a new cell. Finally, the selected value m_i is deleted from A. As we proceed in the algorithm an array list is built up.

Algorithm 2. Second Tier Data Cleaning.

Input:
N: number of nodes associated to one aggregator A.
K: number of measurements received by the aggregator A.
A= $(\cup_{nk}\lambda, m \mid n \in N, k \in K) = \{(\lambda_{nk}, m_{nk}) \mid n \in N, k \in K\} = \{(\lambda_{11}, m_{11}), (\lambda_{12}, m_{12}), .., (\lambda_{21}, m_{21}), \ldots, (\lambda_{nk}, m_{nk})\}$.
Output:
Final dataset sent to the sink.
Initialization:
We create a cell $C_a[1]$ which contain a random value from the set A.
L←K
T←1 //T is the number of cells created
For i ←2 to L **do**
Remove←False
For j=1 to T **do**
Compute Substitution(C_a[j](m),A[i](m))
If Substitution(C_a[j](m),A[i](m)) =1 **Then**
C_a[j](λ) ← C_a[j](λ) + A[i](λ)
Remove A[i](λ, m) from the set A
Remove←True
End if
End For
If remove ←False **Then**
Build a cell C_a[j+1] to contain A[i](λ, m)
Remove A[i](λ, m) from A
Remove←True
End if
L← length (A)
T← number of cells created for an aggregator.
End For
Built Array list of measures and weights.

3.4. Illustrative Example

Let AM be the set of values related to one type of measures received from different nodes connected to an aggregator A.

$$AM = \{(\lambda_{11}, m_{11}), (\lambda_{12}, m_{12}), .., (\lambda_{21}, m_{21}), \ldots, (\lambda_{nk}, m_{nk})\}.$$

We create the first cell $C_a[1]$ in the array list where we place the first value (λ_{11}, m_{11}).For each (λ_{ij},m_{ij}) we compute Substitution (C_a[1](m), m_{ij}) where m is a measure from AM.

If the function returns 1 it means that these two measures are similar. Then the weights are added to each other and we remove (λ_{ij}, m_{ij}) from the set A, else we create a cell $C_a[2]$ for m_{ij} affected of the weight λ_{ij} as shown in Table 1 for ($\lambda_{12,}$ m_{12}). At the end we remove (λ_{ij}, m_{ij}) from the set A.

Table 1. Array List under creation.

Cells	***C_a[1]***	***C_a[2]***
	λ_{11}, m_{11}	$\lambda_{12,}$ m_{12}

Supposing we are in the case where the measures are not similar we continue as follows:

We move to (λ_{13}, m_{13}), then we check if the similarity is reached with the measure m. If so, the weights are added as follows: $C_a[1](\lambda) = C_a[1](\lambda) + \lambda_{13}$ and $m=m_{13}$ is removed from the set A. Otherwise we continue checking the similarity with the measure existing in the second cell. If Substitution ($C_a[2](m)$, m_{13}) =1 then $C_a[2](\lambda) = C_a[2](\lambda) + \lambda_{13}$ and m_{13} is removed from the set A. If the measure is not similar with any of the existing measure in the array we create a cell $C_a[3]$ for m_{13} affected by its weight λ_{13} before we remove (λ_{13}, $m_{13)}$ from the set A. Instead of looping through the entire set of values in A, we are only scanning the cells progressively created in the dynamic array list while computing the Substitution function and reducing the original set size. If the latter is not verified then we create a new cell containing the measure with its related weight otherwise we are only adding the weight to an existing slot as in Table 2.

Table 2. Sample of the Result Set sent to the aggregator.

Cells	**C_a[1]**	**C_a[2]**
Weight	$\lambda_{11,}$ m_{11}	$(\lambda_{12}+\lambda_{13})$,$m_{12}$

3.5. Generalization

A generalization of Algorithm 1 and Algorithm 2 will be to take into consideration many type of measures and replace the scalar by a vector of scalar as follows:

Replace (m[t1] , λ1) by {(m1[t1], λ11) , (m2[t1]], λ21), ... (mn[t1]], λn1) }

Replace A= $(\cup_{nk}\lambda,m|\ n\in N,\ k\in K\ )= \{(\lambda_{nk},\ m_{nk})|\ n\in N,\ k\in K\} = \{(\lambda_{11},\ m_{11}),\ (\lambda_{12},\ m_{12})\ ,..,\ (\lambda_{21},\ m_{21}),\ \ldots,\ (\lambda_{nk},\ m_{nk})\}$ by A= $(\cup_{nk}(\cup_i\ \lambda_i,m_i|i\in M)|\ n\in N,\ k\in K\ )= \{(\lambda_{ink},\ m_{ink})\ |\ i\in M,\ n\in N,\ k\in K\} = \{\cup(\lambda_{11},\ m_{11}),\ \cup(\lambda_{12},\ m_{12})\ ,..,\ \cup(\lambda_{21},\ m_{21}),\ \ldots,\ \cup(\lambda_{nk},\ m_{nk})\}= \{M_1,\ M_2,\ ...,\ M_n\}$.

As a result, the output of the second tier data cleaning will be an array list containing the list of different type of measures along with their weight.

4. Mining Cleaned Sensor Data

After the first phase of data cleaning, the result set constitutes a perfect candidate for data mining. The purpose of this step is to analyze the data and summarize it into useful information - information that can be used to control natural disaster, improve food quality, improve human well being, etc... At the end of this step, only the useful information is sent to the sink optimizing by this the size of the data sent, thus optimizing the energy. Finding frequent patterns is one of the common tasks of datamining. Methods for mining frequent itemsets have been implemented using a prefix tree structure, known as FP_Tree, for storing compressed information about frequent itemsets. In this section, we present a novel technique adapted on sensor network using the array list built in Algorithm 2 that reduces the need to traverse FP- Trees, thus obtaining significant improvement of performance for FP- Tree based algorithm and allowing them to be implemented in sensor networks as to optimize the energy consumed during the transmission between the aggregator and the sink.

4.1. Frequent Pattern Tree (FP- Tree) Method

The FP- tree is compact representation off all relevant frequency information in a database. Each branch is constructed based on the frequent item set [15]. the nodes along the branches are stored in decreasing order of frequency of the corresponding items, which leaves representing the least frequent items. Compression is achieved by building the tree in such a way that overlapping itemsets share prefixes of the corresponding branches. The FP_Tree has a header table associated with it. Single items and their counts are stored in the header table in decreasing order of their frequency. The entry for an item also contains the head of a list that links all the corresponding nodes of the FP_Tree.

This method needs only two database scans when mining all frequent itemsets. The first scan counts the number of occurrences of each item. The second scan constructs the initial FP_Tree which contains all frequent information on the original dataset. From the FP_Tree, all frequent patterns can be generated.

Let us illustrate by an example the algorithm to build an FP_Tree on a transactional real sensor data measurement which will constitute our database collected by 46 sensors deployed in the Intel Berkeley Research lab. Mica2Dot sensors with weather boards collected timestamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds. Data was collected using the TinyDB in-network query processing system, built on the TinyOS platform. In our experiments, we used a file that includes a log of about 2.3 million readings collected from these sensors. Fig. 2 shows a screen capture of this file. Temperature is in degrees Celsius. Humidity is temperature corrected relative humidity, ranging from 0-100%. Light is in Lux. Voltage is expressed in volts [14].

As example, we will take a snapshot of the database as shown in Table 3.

Table. 3. Example of real data.

Temperature	Humidity	Voltage	Light
23,2126	34,0139	2,48502	121,44
23,046	34,0139	2,71196	397,44
22,4874	34,0139	2,474679	7,36
22,4972	34,0139	2,53812	507,84
21,8112	34,0139	2,65143	104,88
21,8112	34,0139	2,66332	104,88
22,1836	34,0139	2,45421	10,12
22,3208	34,0139	2,45421	10,12
21,7622	34,0139	2,66332	104,88

We start by scanning the transaction database DB once. Collect F, the set of frequent items, and the support of each frequent item. Then, we sort F in support-descending order as FList, the list of frequent items.

	A	B	C	D	E	F	G	H
1	Date	Time	Slot	Node id	Temperature	Humidity	Light	Voltage
2	03/03/2004	00:27,0	12005	1	18,3812	29,9691	1,38	2,63964
3	03/03/2004	01:25,6	12007	1	18,4008	29,8271	1,38	2,63964
4	03/03/2004	01:56,5	12008	1	18,4008	29,7561	1,38	2,63964
5	03/03/2004	02:25,8	12009	1	18,4008	29,8271	1,38	2,63964
6	03/03/2004	02:59,0	12010	1	18,391	29,8271	1,38	2,63964
7	03/03/2004	03:27,7	12011	1	18,3812	29,7561	1,38	2,63964
8	03/03/2004	03:57,7	12012	1	18,3812	29,7916	1,38	2,63964
9	03/03/2004	04:28,7	12013	1	18,3812	29,8271	1,38	2,63964
10	03/03/2004	05:57,4	12016	1	18,3714	29,8981	1,38	2,63964
11	03/03/2004	06:28,1	12017	1	18,3616	29,8981	1,38	2,63964
12	03/03/2004	07:00,1	12018	1	18,3616	29,8981	1,38	2,63964
13	03/03/2004	07:55,2	12020	1	18,3616	29,8981	1,38	2,63964
14	03/03/2004	08:25,3	12021	1	18,3616	29,8981	1,38	2,63964
15	03/03/2004	08:59,9	12022	1	18,3714	29,8981	1,38	2,62796
16	03/03/2004	09:55,3	12024	1	18,3616	29,8271	1,38	2,63964
17	03/03/2004	10:25,3	12025	1	18,3616	29,7916	1,38	2,63964
18	03/03/2004	10:55,8	12026	1	18,3518	29,8981	1,38	2,63964
19	03/03/2004	11:27,0	12027	1	18,3616	29,9336	1,38	2,63964
20	03/03/2004	11:58,4	12028	1	18,3518	29,9691	1,38	2,63964
21	03/03/2004	13:27,0	12031	1	18,3518	29,9691	1,38	2,63964
22	03/03/2004	14:01,4	12032	1	18,3518	29,9691	1,38	2,62796
23	03/03/2004	14:25,9	12033	1	18,3518	30,0401	1,38	2,63964

Fig. 2. Snapshot of real data.

Assume that the minimum support threshold is 2. The algorithm makes the first pass through the database and finds singleton itemsets (items) with enough support as in Table 4.

Table. 4. Header table.

Measures: Head of node links	Frequency
Humidity 34,0139: 9	9
Temperature 21,8112 : 2	2
Voltage 2,45421: 2	2
Voltage 2,66332 : 2	2
Light 104,88: 2	2
Light 10,12: 2	2

The next step will be to create the root of an FP_Tree, T, and label it as null. For each transaction Trans in DB do the following:

Select the frequent items in Trans and sort them according to the order of FList. Let the sorted frequent-item list in Trans be [p | P], where p is the first element and P is the remaining list. Call insert tree([p | P], T). The function insert tree([p | P], T) is performed as follows. If T has a child N such that N.item-name = p.item-name, then increment Ns count by 1; else create a new node N, with its count initialized to 1, its parent link linked to T , and its node-link linked to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree(P, N) recursively.

We then go through each transaction at a time, figure out which of the single items are contained in that transaction, and insert these items into the FP_Tree data structure based on the prefix path. Once the prefix path deviates from the tree, a new branch is created. The hope is that the FP_Tree won't have to keep adding nodes and edges, just increment the counts at existing nodes as shown in Fig. 3.

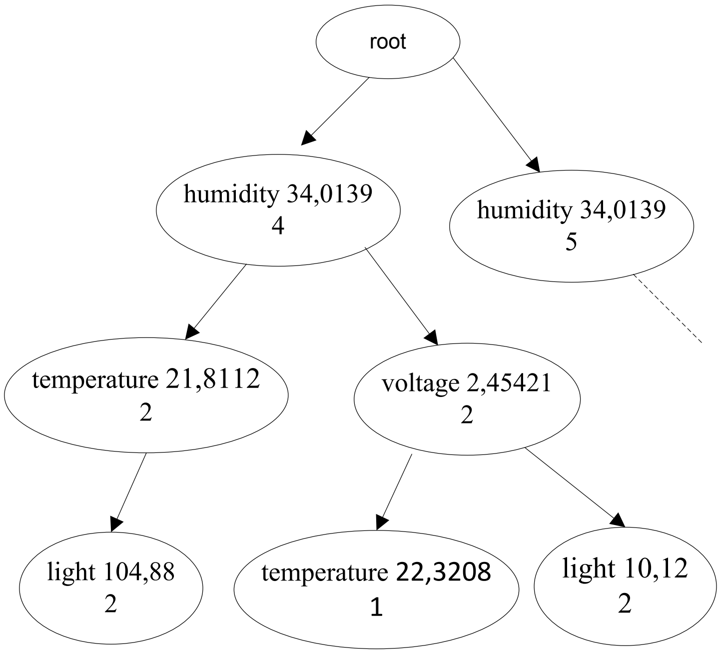

Fig. 3. FP_Tree.

To find the frequent patterns we start with the reference table (from the bottom) and follow the pointers from node to node.

The main disadvantage in this frequency pattern algorithm is that the FP- tree may not fit in memory in addition to the fact that it is expensive to build.

4.2. Our Technique

In the existing method described above, the FP_Tree is traversed in order to construct new conditional FP_Trees after the first FP_Tree is constructed from the original database. About 80% of the CPU time is used for traversing FP_Trees. Thus the main challenge is to reduce the traversal time and speed up the processing.

As per Algorithm 2 already presented, an arraylist of reduced itemsets with their frequency is built thus, allowing us to avoid the full scan of the database. The first scan is omitted as the array list will consist our header table and our database. Next, the prefix tree method is called to mine frequent itemsets for each item in header table. A scan is then done on the arraylist to construct the path of all item pairs. This technique saves traversal time for all items and save the quantity of data sent to the sink by sending only the association rule. As a result, for example we will have the following to be sent to the sink:

If humidity =34,0139 and temperature = 21,8112 then light =104,88 and voltage = 2,65143

This means that we can send only the association rule along with values of two types of measures instead of sending all measures to the sink.

5. Experimental Results

To validate the approach presented in this paper, we developed a C# based simulator that we ran on the readings collected from 46 sensors deployed in the Intel Berkeley Research Lab [13]. Every 31 seconds, sensors with weather boards were collecting humidity, temperature, light and voltage values. In our experiments, we are interested in two sensors measurements: the temperature and the humidity. Each node reads an average of 83000 values of each measurement per day and per field. Our approach consists of a two tiers aggregation: (1) first tier where the aggregation is done on a periodic basis every 31 seconds (2) second tier where the aggregation is done on the level of the aggregator that receives the input from a group of nodes.

5.1. First Tier: Periodic Data Aggregation

At the first tier, data is filtered on a periodic basis where each period is constituted of 31 seconds. At each period, each measure is affected by its weight. The result depends from the threshold delta that we choose to vary between 0.01 and 0.07 based on the variation of measurements. Fig. 4 shows the percentage of data sent to the aggregator. Obviously the data size is disproportional to the threshold data. The goal of this tier is to reduce the size of the data collected by each node while preserving the frequency of each value as to not affect the analysis on the sink level. The experimental results show that a minimum of 5% of the total set for each measure remains. The size of the affected probability for each value is equal to the number of items existing in the message to be sent to the aggregator. The total size of the messages sent to the aggregator is then equal to the total number of measures to be sent in addition to the total number of affected probability. As per the experimental results displayed in Fig. 4, minimum of 10% for each measure is sent to the aggregator.

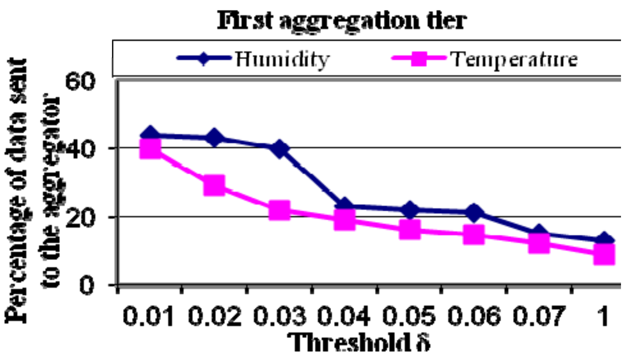

Fig. 4. First Tier Data Aggregation.

5.2. Second Tier: Group Weighted Data Aggregation

At this level, sets of weighted data measures are received by the aggregator. The cleaning on the level of the aggregator can’t ignore the weight of each measure. Weighted data aggregation between sets is performed at the level of the aggregator taking as input the sets received and giving as output one reduced set containing the cleaned measure associated with their weight. The weight of each measure can define the probability of the s data measure existence in this aggregator. Result in Fig. 5 shows that maximum 13% of the data is sent to the sink adding to it 13% related to their respective probability of occurrence. We conclude that only 26 % of the size of messages received by each aggregator A will be the cleaned data generated by Algorithm 2.

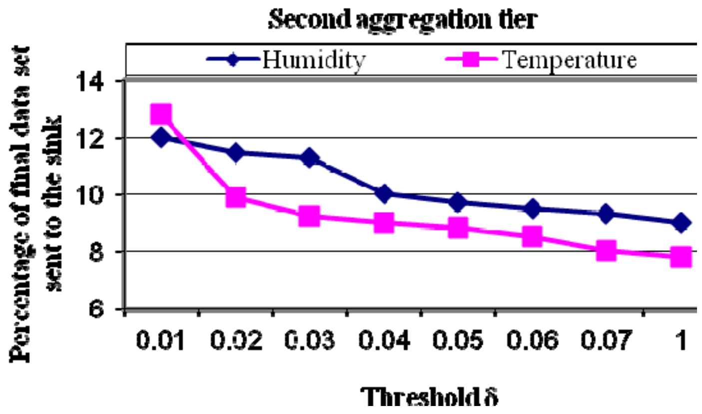

Fig. 5. Second Tier Data Aggregation.

5.3. Energy Study

Sensor nodes that are used to form a sensor network are normally operated by a small battery which has small amount of energy. Therefore, in wireless sensor networks reducing energy consumption of each sensor node is one of the prominent issues to address in the network lifetime, since wireless communications consume significant amount of battery power, sensor nodes should be energy efficient in transmitting data. Protocols can reduce transmitted power in two ways. First where nodes can emit to short distances such as data sinks or cluster nodes. The cluster node can then send the data over a larger distance preserving the power of the smaller nodes. The second is by reducing the number of bits (amount of data) sent across the wireless network. Our approach reduces the overhead by detecting and cleaning redundant measures while preserving the information integrity. To evaluate the energy consumption of our approach we used the same radio model as discussed in [20]. In this model, a radio dissipates Eelec = 50 nJ/bit to run the transmitter or receiver circuitry and βamp = 100 pj/bit/m^2 for the transmitter amplifier. The radios have power control and can expend the minimum required energy to reach the intended recipients as well as they can be turned off to avoid receiving unintended transmissions. The equations used to calculate transmission costs and receiving costs for a k-bit messages and a distance d are respectively shown below in (3) and (4):

$$E_{TX}(\kappa,\ d) = E_{elec} * \kappa + \beta_{amp} * \kappa * d^2. \tag{3}$$

$$E_{RX}(\kappa,\ d) = E_{elec} * \kappa. \tag{4}$$

Receiving is also a high cost operation, therefore, the number of receptions and transmissions should be minimal. In our simulations, we used a measure length k of 64 bits which, corresponds to a packet length. With these radio parameters, when d^2 is 500 m^2, the energy spent in the amplifier part is equal to the energy spent in the electronics part, and therefore, the cost to transmit a packet will be twice the cost to receive.

At the first level, and at the end of the period, each node will contain m messages affected each by a weight λ. The size of the message sent by each node is equal to the number of weight sent in addition to the number of values sent. We consider that each value is equal to 64 bits. The total energy consumed is equal to the sum of the energy consumed by each node when the packet is sent to the aggregator from the source nodes and can be calculated as follows:

$$E_{agg}(\kappa, d) = \Sigma E_{Tx} (\kappa, d) + E_{Px}(\kappa) = \Sigma(E_{elec} * \kappa + \beta_{amp} * \kappa * d^2) + E_{elec} * \kappa. \tag{5}$$

At the second level, the energy consumption will be equal to the energy consumed when the aggregator send the data to the sink in addition to the energy consumed by the sink when receiving the data as shown in (6).

$$E_{sink}(\kappa, d) = \Sigma E_{Tx} (\kappa, d) + E_{Rx} (\kappa) = (E_{elec} * \kappa + \beta_{amp} * \kappa * d^2) + E_{elec} * \kappa. \tag{6}$$

The total energy consumed on the level of the network is calculated as follows:

$$E = E_{agg} (\kappa, d) + E_{sink} (\kappa, d). \tag{7}$$

To evaluate the energy consumption of our approach we compared it to a classical clustering approach, where every node sends all its measures to a cluster head which, in his turn relays all the received data to the sink. Fig. 6 shows that our approach outperforms clustering approaches and minimizes the energy consumption by at least 50%.

Our approach is efficient since the information integrity is fully preserved. All taken measurements appearing in the final set arrived to the sink along with their weight. Therefore, we can consider that our approach decreases the amount of redundant data forwarded to the sink and performs an overall lossless process in terms of information and integrity by conserving the weight of each measure.

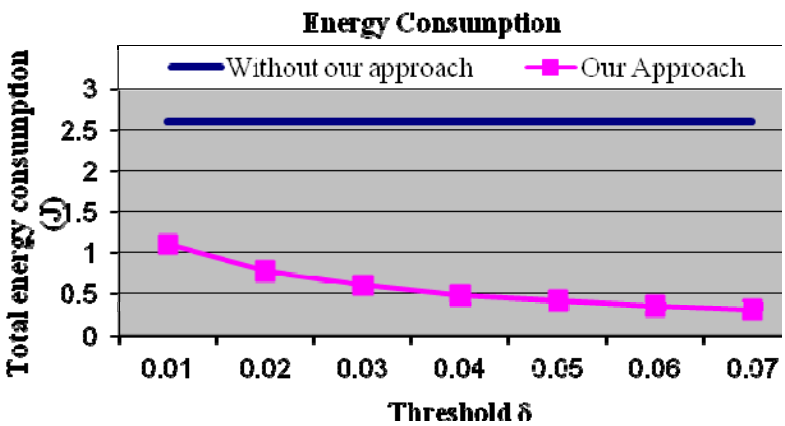

Fig. 6. Energy Consumption.

6. Conclusions

Data aggregation is a well known technique to achieve energy efficiency, in wireless sensor networks, when propagating data from sensor nodes to the sink. The main idea behind is that rather than sending all captured data from sensors to the sink, multiple redundant data are aggregated as they are forwarded by the sensor network. In our approach, we proposed two-tiers weighted periodic data aggregation method. We provided two non complex algorithms that allow at the first level sensor nodes, and at the second level aggregators to identify and reduce duplicate sensor measurements. The experimental results show the effectiveness of our approach in reducing the amount of redundant data;

furthermore, we confirm that the proposed method outperforms existing clustering method in terms of energy consumption. We show the effectiveness of this approach in data mining and how the weighted measures in the training set will serve as an optimization key to generate the frequent patterns.

As a future work, it would be interesting to compare results with other algorithms to mining frequent itemsets and to check if our approach can optimize other datamining algorithm.

References

[1]. G. Pottie and W. Kaiser, Wireless integrated network sensors, *Communications of the ACM*, Vol. 43, No. 5, 2000, p. 51C58.
[2]. Y. Zhuang and L. Chen Hong Kong, In-network Outlier Cleaning for Data Collection in Sensor Networks, in *Proceedings of the 1st International VLDB Workshop on Clean Databases, (CleanDB06),* 2006.
[3]. R. Verdone, D. Dardari, G. Mazzini, and A. Conti, Wireless Sensor and Actuator Networks, *Academic Press/Elsevier,* London, 2008.
[4]. R. Rajagopalan and P. K. Varshney, Data-Aggregation Techniques in Sensor Networks: A Survey, *IEEE Communication Surveys and Tutorials,* Vol. 8, No. 4, December 2006, pp. 48-63.
[5]. X. Li, A Survey on Data Aggregation in Wireless Sensor Networks, Project Report for CMPT 765, Spring 2006.
[6]. Bakhtiar Qutub, N. Pissinou, and K. Makki, Belief based data cleaning for wireless sensor networks, *Wireless Communications and Mobile Computing,* 11, 2011.
[7]. Mehdi Esnaashari and M. R. Meybodi, Data aggregation in sensor networks using learning automata, *Wireless Networks*, 16, 3, 2010, pp. 687–699.
[8]. R E. Elnahrawy and B. Nath, Cleaning and querying noisy sensors, in *Proceeding of the International Workshop of Wireless Sensor Networks and Applications (WSNA),* 2003.
[9]. B. J. Chen, K. Jameison, H. Balakrishnan, and R. Morns, Span: An Energy Efficient coordination Algorithm for Topology Maitnenance in Ad-Hoc Wireless Networks, *Wireless Networks,* 8, 5, 2002, pp. 481-494.
[10]. H. Albert, R. Kravets and I. Gupta, Building Trees Based On Aggregation Efficiency in Sensor Networks, *Ad Hoc Networks*, Vol. 5, No. 8, November 2007, pp. 1317-1328.
[11]. E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, In-network Aggregation Techniques for Wireless Sensor Networks: A Survey, *IEEE Wireless Communications*, Vol. 14, No. 2, April 2007, pp. 70-87.
[12]. Y. Zheng, K. Chen, and W. Qiu, Building Representative-Based Data Aggregation Tree in Wireless Sensor Networks, *Mathematical Problems in Engineering*, Vol. 2010, Article ID 732892, 2010, 11 pages.
[13]. http://db.csail.mit.edu/labdata/labdata.html, Apr. 2004, Last access March 2011.
[14]. Samuel Madden, http://db.csail.mit.edu/labdata/labdata.html
[15]. Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, *Data Mining and Knowledge Discovery*, Vol. 8, Issue 1, January 2004, pp. 53-87.
[16]. J. Han, H. Pei, and Y. Yin, Mining Frequent Patterns without Candidate Generation, in *Proc. of the Conf. on the Management of Data (SIGMOD'00),* Dallas, TX, USA, 2000.
[17]. Agrawal, R. and Srikant, R. Fast algorithms for mining association rules, in *Proc. of the Int. Conf. Very Large Data Bases (VLDB '94),* Santiago, Chile, 1994, pp. 487–499.
[18]. Agarwal, R., Aggarwal, C., and Prasad, V. V. V., A tree projection algorithm for generation of frequent itemsets, *Journal of Parallel and Distributed Computing*, 61, 2001, pp. 350–371.

Guide for Contributors

Aims and Scope

Sensors & Transducers Journal (ISSN 1726-5479) provides an advanced forum for the science and technology of physical, chemical sensors and biosensors. It publishes state-of-the-art reviews, regular research and application specific papers, short notes, letters to Editor and sensors related books reviews as well as academic, practical and commercial information of interest to its readership. Because of it is a peer reviewed international journal, papers rapidly published in *Sensors & Transducers Journal* will receive a very high publicity. The journal is published monthly as twelve issues per year by International Frequency Sensor Association (IFSA). In additional, some special sponsored and conference issues published annually. *Sensors & Transducers Journal* is indexed and abstracted very quickly by Chemical Abstracts, IndexCopernicus Journals Master List, Open J-Gate, Google Scholar, etc. Since 2011 the journal is covered and indexed (including a Scopus, Embase, Engineering Village and Reaxys) in Elsevier products.

Topics Covered

Contributions are invited on all aspects of research, development and application of the science and technology of sensors, transducers and sensor instrumentations. Topics include, but are not restricted to:

- Physical, chemical and biosensors;
- Digital, frequency, period, duty-cycle, time interval, PWM, pulse number output sensors and transducers;
- Theory, principles, effects, design, standardization and modeling;
- Smart sensors and systems;
- Sensor instrumentation;
- Virtual instruments;
- Sensors interfaces, buses and networks;
- Signal processing;
- Frequency (period, duty-cycle)-to-digital converters, ADC;
- Technologies and materials;
- Nanosensors;
- Microsystems;
- Applications.

Submission of papers

Articles should be written in English. Authors are invited to submit by e-mail editor@sensorsportal.com 8-14 pages article (including abstract, illustrations (color or grayscale), photos and references) in both: MS Word (doc) and Acrobat (pdf) formats. Detailed preparation instructions, paper example and template of manuscript are available from the journal's webpage: http://www.sensorsportal.com/HTML/DIGEST/Submition.htm Authors must follow the instructions strictly when submitting their manuscripts.

Advertising Information

Advertising orders and enquires may be sent to sales@sensorsportal.com Please download also our media kit: http://www.sensorsportal.com/DOWNLOADS/Media_Kit_2012.pdf