H-cluster: A Novel Efficient Algorithm for Data Clustering in Sensor Networks

Longjiang Guo[1, 2], Meirui Ren[1, 2], Jinbao Li[1, 2*], Yong Liu[1, 2], Chunyu Ai[3]
[1]School of Computer Science and Technology, Heilongjiang University, Harbin, China, 150080
[2]Key Laboratory of Database and Parallel Computing Heilongjiang Province, Harbin, China, 150080
[3]Computer Science, Troy University, Troy, AL 36082, USA
Email: JBLI@hlju.edu.cn; longjiangguo@gmail.com

***Abstract*—This paper focuses on the problem of data clustering in wireless sensor networks (WSNs). The data time window is a landmark window, from the time WSN starts working up to the current time. The objective is to group sensory data generated by sensor nodes deployed in a two-dimensional physical space by the similarity of sensory data in the multi-dimensional sensory data space. To perform in-network data clustering efficiently, we propose *HilbertMap*, a novel dimensionality reduction technique based on the Hilbert Curves, to map a multi-dimensional data space to a two-dimensional physical space. Through this mapping, the communications for clustering mostly occur between geographically nearby sensor nodes. We have conducted simulation experiments on both real-world and synthetic datasets. Our results show that *HilbertMap* improves the communication efficiency while maintaining a good clustering quality.**

***Index Terms*—Hilbert mapping, sensor networks, data clustering.**

I. INTRODUCTION

Wireless sensor networks (WSNs) have been employed in variety of wide-area environmental continuous monitoring applications, such as continuous monitoring of water distribution systems [1], real-time surveillance of maritime zones [2] and drinking water quality monitoring [3]. These applications require either continuous, real-time monitoring or periodical-based and conditional-based online analytical results about the environment.

As an example application, it is necessary to monitor ingredient of water and determine the quality of water in real time in aquiculture [4]. Culturists expect to know in real time how many types of water quality there exist so that they can control quantity of fish and bivalve mollusks in real time to avoid economic loss [4]. In this application, real time clustering of *d*-dimensional sensory data is very important, where *d* is the dimensionality of sensory data. A wireless underwater sensor network monitoring system[2] can be deployed below ocean water or freshwater that monitors parameters of water such as water temperature, dissolved oxygen, salinity, PH, mercury, cadmium and so on. This wireless underwater sensor network can report the number of types of water quality in real time manner according to clustering results of *d*-dimensional sensory data.

As another example application, researchers develop generic WSNs to enable real time monitoring of water distribution and sewer networks [1]. The project involves three main applications: (1) water conservation; (2) integrated monitoring of hydraulic and drinking water quality parameters; (3) development of systems to enable remote detection of leaks and prediction of pipe burst events. In the second application, environmentalists expect to know in real time how many types of drinking water quality there exist so that they can control quantity of microorganism and chemical medicine in real time to ensure the quality of drinking water. In this application, real time clustering of *d*-dimensional sensory data is also very important. A chemical wireless sensor network monitoring system [3] can be deployed below drinking water to monitor parameters of water such as microorganism, dissolved oxygen, PH, mercury, cadmium and other chemical medicine. This wireless sensor network can real-time report how many types of drinking water quality according to clustering results of *d*-dimensional sensory data.

Regarding clustering in WSNs, there exist some works on clustering sensor nodes such as [5-7] which groups sensor nodes according to their geographic positions mainly for routing, node localization and data aggregation. This paper is different from clustering sensor nodes. The main innovation of this paper is that we propose a clustering algorithm *H-Cluster* to cluster sensory data collected by all nodes in a WSN based on the similarity of the sensory data rather than clustering sensor nodes. This novel clustering algorithm emphasizes the data-centric characteristic of WSNs. It is useful if users need overviews of the areas monitored by WSNs.

Clustering algorithms in data mining have been extensively investigated [8]. Generally, there exist four kinds of clustering algorithms: partitioning algorithms, hierarchical algorithms, density based algorithms and grid based algorithms [8]. However, these algorithms are designed specifically for data mining in general computers. Limitations of WSNs, such as network bandwidth and power supply, are not considered by these works. Therefore, most of the existing clustering algorithms are centralized which are not suitable for WSNs. The detailed reasons are given as follows: a centralized algorithm may need to be run at the base station. This requires all sensors

* To whom correspondences should be addressed. JBLI@hlju.edu.cn
Manuscript received August 15, 2010; revised November 15, 2010; accepted January 15, 2011.

doi:10.4304/jcm.6.2.168-178

to send their data to the sink, and therefore is inefficient, especially for large scale WSNs. First, the data amount of the entire network could be huge, whereas the size of the clustering result might be much smaller. Sending all sensory data to the sink consumes much energy. Also, energy conservation is a primary concern for WSNs and data transmission dominates energy consumption. It has been shown in [9] that the energy for transmitting 1kB of data over 100m is almost enough for executing three million instructions. Second, the available communication bandwidth is very limited in WSNs. The transmission of the huge amount of data as well as noise from the outside world overwhelms the wireless channel and may cause severe data loss, which results in poor clustering quality.

To alleviate the aforementioned problems, we propose a novel and efficient in-network sensory data clustering algorithm called *H-Cluster*. The input of this algorithm is the set of sensory data collected by all the sensors in a WSN from the time of WSN starts working up to current time. The output of this algorithm is a set of cluster features that summarize the clusters of the input sensory data set. The cluster features requires limited storage on each node. For the purpose of energy conservation, we need to constrain these communications to neighboring sensors that are geographically close. We propose *HilbertMap*, a mapping algorithm based on the Hilbert Curves, to map a *d*-dimensional sensory data space into a two-dimensional area covered by a WSN. Through this mapping, sensory data items that are close in the data space are first mapped to sensors that are geographically close. Next, clustering of the sensory data is carried out only through local communications among sensors that are close to each other. The communications for clustering are highly localized and energy consumption is reduced.

Recently, two distributed algorithms Elink [10] and DSIC [11] have been proposed to perform data clustering in sensor networks. The two state-of-the-art algorithms focus on clustering fresh sensory data at the current time, whereas this paper focuses on the accumulated data from the time of WSN starts working up to current time. Elink and DSIC are the algorithms for clustering snapshot data in WSNs. This is the main difference between H-cluster, Elink and DSIC. The main contributions are as follows:

1. We propose *HilbertMap*, a mapping algorithm to map a *d*-dimensional sensory data space virtually generated by a WSN into a two-dimensional area covered by the WSN such that sensory data items that are close in the data space are forwarded to the sensors that are geographically close. *HilbertMap* can be easily implemented on a sensor mote thanks to the low complexity of *HilbertMap*, which is O(*d*), where *d* is the dimensionality of sensory data.

2. We develop a distributed clustering algorithm *H-Cluster* based on *HilbertMap*. Through *HilbertMap*, the communications for clustering mostly occur among geographically adjacent sensors and this in-network processing technique efficiently avoids spending unnecessary energy in finding global clusters.

3. Extensive simulations were conducted to evaluate the performance of *H-Cluster*. The simulation results show that *H-Cluster* consistently achieves the lowest data loss rate and the highest energy efficiency. Due to its low data loss rate, *H-Cluster* achieves better clustering quality.

The rest of the paper is organized as follows. Section II introduces the problem description of this paper. Section III explains the detailed approaches of sensory data relocating for data clustering. Section IV presents in-network data clustering algorithm. Simulation results are presented in Section V. Section VI gives all related work. Finally, Section VII concludes the paper.

II. PROBLEM DESCRIPTION

A. Sensory Data Cluster

Assume that all the nodes in a WSN are synchronized and have knowledge of their locations. Each sensor node is capable of monitoring the same *d* observation attributes. Time is divided into intervals with a fixed length. Each interval is called an *epoch* during which a sensor senses its surrounding environment and generates *d*-dimensional sensory data. The duration of an epoch is application-specific and depends on the change rate of the measured physical attributes. A time point is given by the number of epochs that have elapsed since beginning. *N* sensor nodes are randomly deployed to a rectangular area.

Definition 2.1. (Sensory Data Space) *Let A_k be the domain of the k^{th} monitored attribute in a given WSN for $1 \le k \le d$. $A_1 \times A_2 \times ... \times A_d$ is defined as a d-dimensional Sensory Data Space (SDS) of the WSN, and each $\overrightarrow{S_j^i} = (s_{j1}^i, s_{j2}^i, ..., s_{jd}^i) \in SDS$ is called a d-dimensional sensory data item collected by sensor node j at epoch i, where s_{jk}^i is drawn from A_k for $1 \le k \le d$.*

Definition 2.2. (Similarity) *Given a threshold vector $\vec{\delta} = (\delta_1, \delta_2, ...\delta_d)^{\mathrm{T}}$, two sensory data items $\overrightarrow{S_j^i}$ and $\overrightarrow{S_l^i}$ are similar if $|s_{jk}^i - s_{lk}^i| < \delta_k$ for $1 \le k \le d$.*

The definition of similarity is derived from L1-norm. L1-norm is a typical metric for measuring similarity between two data items [12]. To make the concept of sensory data cluster clear, we partition the *SDS* into non-overlapping *grids*. This is accomplished by partitioning each domain A_k into m intervals with equal length for $1 \le k \le d$. The length of each interval on A_k is $\Delta_k = (Max_k - Min_k)/m$, where $A_k = [Min_k, Max_k]$ $(1 \le k \le d)$. We assume that all the nodes in a WSN have the knowledge about $[Min_k, Max_k]$ of dimension A_k $(1 \le k \le d)$. This assumption is reasonable, because $[Min_k, Max_k]$ of dimension A_k $(1 \le k \le d)$ can be estimated by history data. Each grid $\vec{u}$ is expressed as $\vec{u} = (u_1, u_2, ..., u_d)$ and grid $\vec{u}$ is the intersection of right-open interval $[Min_k + u_k\Delta_k,\ Min_k + (u_k + 1)\Delta_k)$ $(1 \le k \le d)$.

A sensory data item $\overrightarrow{S_j^i} = (s_{j1}^i, s_{j2}^i, ..., s_{jd}^i)$ is *contained in* a grid $\vec{u} = (u_1, u_2, ..., u_d)$ if $s_{jk}^i \in [Min_k + u_k\Delta_k,\ Min_k + (u_k + 1)\Delta_k)$, $(1 \le k \le d)$. *The density of grid $\vec{u}$* is the number of the sensory data items contained in $\vec{u}$, denoted by $Dens(\vec{u})$. A

grid $\vec{u}$ has a unique position, and we can easily convert the position of $\vec{u}$ to a unique number. This unique number is called *grid ID*. There is an one-to-one correspondence between the *grid ID* and *the position of grid* $\vec{u}$. Let $UList$ be a set of grids. The average density of $UList$ is $\frac{1}{|UList|}\sum_{\vec{u}\in UList} Dens(\vec{u})$. A grid $\vec{u}$ is a *dense grid* if $Dens(\vec{u}) \geq \lambda$, where λ is a user-specified density threshold. If each dimension of a *SDS* is partitioned into intervals with length $\Delta_k = (Max_k - Min_k)/m$ and $\Delta_k < \delta_k$, the data items contained in the same grid are similar.

Definition 2.3. (Neighbors) *Two grids* $\vec{u} = (u_1, u_2, ..., u_d)$ *and* $\vec{v} = (v_1, v_2, ..., v_d)$ *are neighbors if there exists an integer* k $(1 \leq k \leq d)$ *such that* $|u_k - v_k| = 1$ *and* $u_j = v_j$ *for* $j \neq k$ *and* $1 \leq j \leq d$.

For any grid $\vec{u}$, it has at most 2*d* neighbors. *The neighbor set* of $\vec{u}$ is denoted as $Nbr(\vec{u})$.

Definition 2.4. (Connected) *Two grids* $\vec{u}$ *and* $\vec{v}$ *are connected if* $\vec{u}$ *and* $\vec{v}$ *are neighbors or if there exists another grid* $\vec{w}$ *such that* $\vec{u}$ *and* $\vec{v}$ *are connected to* $\vec{w}$.

Definition 2.5. (Sensory Data Cluster) *Given a threshold vector* $\vec{\delta} = (\delta_1, \delta_2, ...\delta_d)^{\mathrm{T}}$, *a user-specified density threshold* $\lambda \geq 1$, *and a SDS that is partitioned into* m^d *grids, where* $m > \max\{(Max_k - Min_k)/\delta_k\}$, $(1 \leq k \leq d)$. *A sensory data cluster at epoch* i *is defined as a set of grids* $UList$ *satisfying*

(1). $\forall \vec{u} \in UList$: *all sensory data items contained in* $\vec{u}$ *are collected at epoch* i;

(2). $\forall \vec{u}, \vec{v} \in UList$: $\vec{u}$ *and* $\vec{v}$ *are connected*;

(3). $\frac{1}{|UList|}\sum_{\vec{u}\in UList} Dens(\vec{u}) \geq \lambda$: *the average density of the set of grids* $UList$ *is greater than* λ.

After receiving the user-specified threshold vector $\vec{\delta} = (\delta_1, \delta_2, ...\delta_d)^{\mathrm{T}}$ and the user-specified density threshold λ, each sensor node can partition *SDS* into m^d grids, where *m* satisfies $m > \max\{(Max_k - Min_k)/\delta_k\}$ $(1 \leq k \leq d)$.

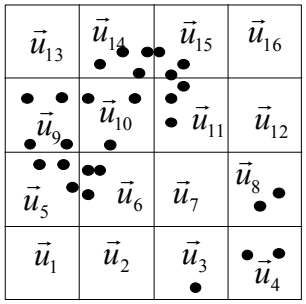

Fig. 1 An example of a cluster

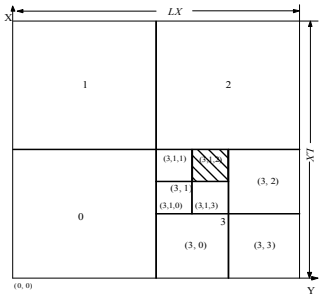

Fig. 2 An example of *HilbertMap*

For example, there is a 2-dimensional *SDS* shown in Fig. 1, where $[Min_1, Max_1]$=[-10, 10] and $[Min_2, Max_2]$=[-20, 20]. In this figure, every black point denotes a sensory data. Given a threshold vector $\vec{\delta} = (\delta_1, \delta_2)^{\mathrm{T}} = (6,11)^{\mathrm{T}}$, a user-specified density threshold $\lambda = 3 \geq 1$. We choose *m*=4, and it satisfies $m > \max\{(Max_k - Min_k)/\delta_k\}$ $(1 \leq k \leq 2)$. Therefore, the *SDS* is partitioned into $m^d = 4^2 = 16$ grids. There is a sensory data cluster $UList = \{\vec{u}_5, \vec{u}_6, \vec{u}_9, \vec{u}_{10}, \vec{u}_{11}, \vec{u}_{14}, \vec{u}_{15}\}$, such that $\forall \vec{u}, \vec{v} \in UList$, $\vec{u}$ and $\vec{v}$ are connected and $\frac{1}{|UList|}\sum_{\vec{u}\in UList} Dens(\vec{u}) = 23/7 \geq \lambda = 3$.

To reduce the storage space, each node only stores data structures of *grid features* and *cluster features* rather than the grids and clusters. G*rid features* and *cluster features* summarize the information of grid and cluster. The concepts of *grid features* and *cluster features* are the core of *H-Cluster*. They are given as follows.

Definition 2.6. (Grid Feature) *Given a grid* $\vec{u}$, *the grid feature of* $\vec{u}$ *is defined as* $GF(\vec{u}) = \{gID, \overrightarrow{LS}, SS, Dens(\vec{u}), IDList\}$, *where gID is the grid ID of* $\vec{u}$, $\overrightarrow{LS}$ *is the linear sum of sensory data contained in* $\vec{u}$, SS *is the square sum of sensory data contained in* $\vec{u}$, $IDList$ *is a set of sensor node IDs* $\{j_1, j_2, ..., j_p\}$ *(it means that the sensory data items in* $\vec{u}$ *come from sensor nodes* $\{j_1, j_2, ..., j_p\}$*) and* p *indicates that the sensory data items in* $\vec{u}$ *come from* p *nodes.*

To reduce the storage space, $IDList$ can be implemented by Bloom Filters [13]. $GF(\vec{u})$ and $GF(\vec{v})$ are called *connected* if and only if $\vec{u}$ and $\vec{v}$ are connected. Two grids features $GF(\vec{u})$ and $GF(\vec{v})$ are called *neighbor*s if and only if $\vec{u}$ and $\vec{v}$ are neighbor grids.

Definition 2.7. (Cluster Feature) *Given a sensory cluster* $UList$, *the cluster feature of* $UList$ *is defined as a set of grid features, denoted by* $CF = \{GF(\vec{u}) \mid \vec{u} \in UList\}$, *satisfying*

(1). $\forall \vec{u} \in UList$, $\exists GF(\vec{u}) \in CF$;

(2). $\forall GF(\vec{u}) \in CF$, $\exists \vec{u} \in UList$;

(3).*There is an one-to-one correspondence between* $UList$ *and* CF.

Two cluster features CF_1 and CF_2 are *connected* if and only if $\exists GF(\vec{u}) \in CF_1$ and $\exists GF(\vec{v}) \in CF_2$, $GF(\vec{u})$ and $GF(\vec{v})$ are connected. The average density of cluster feature CF is defined by $Dens(CF) = \frac{1}{|CF|}\sum_{GF(\vec{u})\in CF} Dens(\vec{u})$.

Obviously, a grid can be described by a grid feature and a cluster can be described by a cluster feature. A cluster feature is a set of connected grid features and its average density is greater than the density threshold λ. In Fig. 1, $CF = \{GF(\vec{u}_5), GF(\vec{u}_6), GF(\vec{u}_9), GF(\vec{u}_{10}), GF(\vec{u}_{11}), GF(\vec{u}_{14}), GF(\vec{u}_{15})\}$ is a cluster feature with λ=3 and $Dens(CF) = 23/7 \geq \lambda$. In the rest of the paper, we only need to deal with the grid features and the cluster features.

B. Problem Definition

The problem studied in this paper is defined as follows: given a WSN with N sensor nodes, a set of *d*-dimensional sensor data $D = \{\overrightarrow{S_j^i}, j = 1,2,...,N;\ i = 1,2,...,NOW\} \subseteq SDS$ collected from beginning epoch 1 up to current epoch

NOW (D is stored in the WSN), a threshold vector $\vec{\delta} = (\delta_1, \delta_2, ...\delta_d)^{\mathrm{T}}$, and a user-specified density threshold λ, design an in-network sensory data clustering algorithm to find k cluster features CF_1, CF_2, ..., and CF_k at current epoch NOW, such that all the sensory data items in any cluster feature CF_j are similar and $Dens(CF_j) \geq \lambda$ $(1 \leq j \leq k)$, where k is unknown in advance and depends on the distribution of D in the *SDS*.

III. Sensory Data Relocating For Clustering

Sensory data relocating is the preparation for clustering. Its purpose is to distribute the data collected by a WSN among sensor nodes, so that the data items that are close in the *d*-dimensional *SDS* are close in the 2-dimensional area covered by the WSN. It guarantees that (1) the *d*-dimensional sensory data in one grid are stored at one sensor node, and (2) the neighbor grids are stored at different sensor nodes that are geographically close in the 2-dimensional area.

Sensory data relocating can be obtained by two mapping algorithms: *FastMap* and *HilbertMap*. *FastMap* is from [14] and HilbertMap is proposed in this paper. The two mapping algorithms are paratactic, their function is that mapping the data items in the SDS into the nodes in the 2-dimensional geographical area coved by a given WSN. This section introduces *FastMap* and *HilbertMap*.

A. FastMap

For self-containedness, we describe our implementation of a modified version of FastMap [14] in this section. The heart of the modified *FastMap* is to project all hypercubes on two perpendicular pivots.

First, we select a pair of hypercubes in *d*-dimensional sensory data space and consider a line that passes through the two hypercubes as the first pivot. To do that, we choose two hypercubes $\vec{u}_1 = (0,0,...,0)$ and $\vec{v}_1 = (m-1, m-1, ..., m-1)$ as first pivot due to $\vec{u}_1$ and $\vec{v}_1$ are farthest pair of hypercubes. The projections of hypercubes on the line ($\vec{u}_1, \vec{v}_1$) are computed by using the cosine law [14] that is for any hypercube $\vec{w}$, the first coordinate of $\vec{w}$ in 2-dimensional geographic area is x_w, where

$$x_w = \frac{D(\vec{u}_1, \vec{w})^2 + D(\vec{u}_1, \vec{v}_1)^2 - D(\vec{v}_1, \vec{w})^2}{2D(\vec{u}_1, \vec{v}_1)}$$

Next, we need select another pair of hypercubes as the second pivot. In order to do this, we project all hypercubes on a (*d*-1)-dimensional hyper-plane that is perpendicular to the line ($\vec{u}_1, \vec{v}_1$), and we choose ($\vec{u}_2, \vec{v}_2$) as second pivot such that the distance between the projections of $\vec{u}_2$ and $\vec{v}_2$ on the (*d*-1)-dimensional hyper-plane is farthest. The distance between the projections of $\vec{u}_2$ and $\vec{v}_2$ on the (*d*-1)-dimensional hyper-plane is defined as $D'(\vec{u}_2, \vec{v}_2)$, and it can be computed from $D(\vec{u}_2, \vec{v}_2)$ as follows [14]:

$$D'(\vec{u}_2, \vec{v}_2) = \sqrt{(D(\vec{u}_2, \vec{v}_2))^2 - (x_{u_2} - x_{v_2})^2}$$

For any hypercube $\vec{w}$, the second coordinate of $\vec{w}$ in 2-dimensional geographic area is y_w, where

$$y_w = \frac{D(\vec{u}_2, \vec{w})^2 + D(\vec{u}_2, \vec{v}_2)^2 - D(\vec{v}_2, \vec{w})^2}{2D(\vec{u}_2, \vec{v}_2)}$$

The time complexity of choosing ($\vec{u}_2, \vec{v}_2$), such that $D'(\vec{u}_2, \vec{v}_2)$ is maximized is O(m^{2d}). Thus Christos et al. proposed a linear random algorithm *choose-distant-objects* [14] that can find ($\vec{u}_2, \vec{v}_2$) such that $D'(\vec{u}_2, \vec{v}_2)$ is approximately maximized. The time complexity of the algorithm *choose-distant-objects* is O(m^d).

Sink can firstly run *choose-distant-objects* to get pivots $\{(\vec{u}_1, \vec{v}_1), (\vec{u}_2, \vec{v}_2)\}$ and $\{(x_{\min}, y_{\min}), (x_{\max}, y_{\max})\}$, where $\begin{cases} x_{\min} = \min\limits_{\vec{w}}\{x_w\} \\ y_{\min} = \min\limits_{\vec{w}}\{y_w\} \end{cases}$, $\begin{cases} x_{\max} = \max\limits_{\vec{w}}\{x_w\} \\ y_{\max} = \max\limits_{\vec{w}}\{y_w\} \end{cases}$ and then flood pivots $\{(\vec{u}_1, \vec{v}_1), (\vec{u}_2, \vec{v}_2)\}$ and $\{(x_{\min}, y_{\min}), (x_{\max}, y_{\max})\}$ to all nodes.

Given a hypercube $\vec{w}$, *FastMap* maps $\vec{w}$ to 2-dimensional coordinate (*x*, *y*), where $\begin{cases} x = \frac{x_w - x_{\min}}{x_{\max} - x_{\min}} \times LX \\ y = \frac{y_w - y_{\min}}{y_{\max} - y_{\min}} \times LY \end{cases}$.

The complexity of the FastMap algorithm is O(m^d).

B. HilbertMap

Let *G* be the set of the grids in a given *SDS* produced by a WSN with node set *S,* and *P* be the set of the positions in a 2-dimensional area covered by the WSN. Based on the *Hilbert Curve*, *HilbertMap* performs the mapping: $G \rightarrow P$, *i.e.*, for each grid's position $\vec{w} = (w_1, w_2, ..., w_d)$ in *G*, *HilbertMap* outputs a position (*x*, *y*) in *P*. This is denoted by $H(\vec{w}) = (x, y)$.

Given a grid's position $\vec{w} = (w_1, w_2, ..., w_d)$ in *G*, *HilbertMap* generates the position (*x*, *y*) in *d* loops. The i^{th} loop works according to w_i. The principle of *HilbertMap* loop by loop is shown as follows.

Loop 1. Assume that the grids in *G* are obtained by partitioning every dimension of the given *SDS* into *m* intervals with equal length. Let integer $j = \lceil \log_4 m \rceil$. The 2-dimensional area is partitioned into 4^j sub-squares each of which has size $\frac{LX}{2^j} \times \frac{LY}{2^j}$, where *LX* and *LY* are the length of the monitored area on *X* and *Y* axes respectively. These 4^j sub-squares are numbered by *Hilbert derived key* [15] from 0 to $4^j - 1$. Since $w_1 \in [0, m-1] \subseteq [0, 4^j]$, w_1 is regarded as a *Hilbert derived key*. Thus, grid $\vec{w}$ is mapped to the w_1^{th} sub-square.

Loop 2. Partition the w_1^{th} sub-square into 4^j sub-sub-squares as in loop 1. These 4^j sub-sub-squares are also numbered by *Hilbert derived keys* from 0 to $4^j - 1$ and the size of each sub-sub-square is $\frac{LX}{2^{2j}} \times \frac{LY}{2^{2j}}$. The $\vec{w}$ can be mapped to the w_2^{th} sub-sub-square in the w_1^{th} sub-square.

The rest *d*-2 loops are similar to loop 2. Finally, grid $\vec{w}$ is mapped into a small square of size $\frac{LX}{2^{d \times j}} \times \frac{LY}{2^{d \times j}}$. The coordinate of the center of this square is the coordinate (*x*, *y*) of the 2-dimensional area in which the $\vec{w}$ should be.

The pseudo-code of *HilbertMap* is in Algorithm 1. The complexity of getting Hilbert coordinate from w_i is O(1) [15], so the complexity of *HilbertMap* is O(d) due to d times of repetition. For example, d=3, m=4, $j=1$, and $\vec{w}=(3,1,2)$. The 2-dimensional area is partitioned into 4 sub-squares. Each sub-square is numbered by a *Hilbert derived key* [15] as shown in Fig. 2. The first dimension coordinate of $\vec{w}$ is 3, so we continue partitioning the 4th sub-square (bottom-right corner) into 4 sub-sub-squares. The procedure repeats 3 times. The coordinate of the centroid of the shaded square is the output of *HilbertMap*.

Algorithm 1 ***HilbertMap***

Input: $\vec{w}=(w_1,w_2,...,w_d)$: grid position

m: the number of intervals partitioning every dimension;

LX, *LY*: the length of the monitored area on *X* and *Y* axes.

Output: (x, y): mapping coordinate in a two-dimensional area covered by sensors.

$x \leftarrow 0$; $y \leftarrow 0$; $pwr \leftarrow 0$; $j \leftarrow \lceil \log_4{}^m \rceil$;

For i =1 To d Do

(a, b)←get Hilbert coordinate from w_i;

$pwr \leftarrow pwr+j$; $x \leftarrow x + a \times \frac{1}{2^{pwr}}$; $y \leftarrow y + b \times \frac{1}{2^{pwr}}$;

Endfor

$x \leftarrow x + \frac{1}{2^{pwr+1}}$; $y \leftarrow y + \frac{1}{2^{pwr+1}}$; $x \leftarrow LX \times x$; $y \leftarrow LY \times y$;

As shown in Theorem 1, *HilbertMap* can guarantee that if two grids are neighbors in a *SDS*, the distance between their mapped positions in the 2-dimensional area are close.

Theorem 1 Assume two grids $\vec{u}=(u_1,u_2,...,u_d)$ and $\vec{v}=(v_1,v_2,...,v_d)$ are neighbors in an *SDS*. Partition every dimension into m intervals with equal length. $H(\vec{u})=(x_u,y_u)$ and $H(\vec{v})=(x_v,y_v)$ are the obtained coordinates in a 2-dimensional area for grids $\vec{u}$ and $\vec{v}$ using *HilbertMap* respectively. $\hat{D}(H(\vec{u}),H(\vec{v}))$ is the Euclidean distance between $H(\vec{u})$ and $H(\vec{v})$ in the 2-dimensional area. $\hat{D}(H(\vec{u}),H(\vec{v}))$ satisfies the following inequality: $\frac{\min\{LX,LY\}}{\sqrt{m^d}} \le \hat{D}(H(\vec{u}),H(\vec{v})) \le \frac{\max\{LX,LY\}}{\sqrt{m}}$.

Proof: Since $\vec{u}=(u_1,u_2,...,u_d)$ and $\vec{v}=(v_1,v_2,...,v_d)$ are neighbors, there exists an integer k $(1 \le k \le d)$ such that $|u_k - v_k| = 1$ and $u_j = v_j$, $j \ne k$, $1 \le j \le d$.

$$\begin{cases} x_u = (\sum_{i \ne k} \frac{a_i LX}{2^{i \times j}}) + \frac{a_k LX}{2^{k \times j}} \\ y_u = (\sum_{i \ne k} \frac{b_i LY}{2^{i \times j}}) + \frac{b_k LY}{2^{k \times j}} \end{cases} \begin{cases} x_v = (\sum_{i \ne k} \frac{a_i LX}{2^{i \times j}}) + \frac{(a_k \pm 1) LX}{2^{k \times j}} \\ y_v = (\sum_{i \ne k} \frac{b_i LY}{2^{i \times j}}) + \frac{b_k LY}{2^{k \times j}} \end{cases}$$

or

$$\begin{cases} x_u = (\sum_{i \ne k} \frac{a_i LX}{2^{i \times j}}) + \frac{a_k LX}{2^{k \times j}} \\ y_u = (\sum_{i \ne k} \frac{b_i LY}{2^{i \times j}}) + \frac{b_k LY}{2^{k \times j}} \end{cases} \begin{cases} x_v = (\sum_{i \ne k} \frac{a_i LX}{2^{i \times j}}) + \frac{a_k LX}{2^{k \times j}} \\ y_v = (\sum_{i \ne k} \frac{b_i LY}{2^{i \times j}}) + \frac{(b_k \pm 1) LY}{2^{k \times j}} \end{cases}$$

Then the Euclidean distance between position (x_u,y_u) and position (x_v,y_v) is $\frac{LX}{2^{j \times k}}$ or $\frac{LY}{2^{j \times k}}$. When k=1, the maximum distance is $\frac{\max\{LX,LY\}}{2^j}$. When k=d, the minimum distance is $\frac{\min\{LX,LY\}}{2^{j \times d}}$. Because $j = \lceil \log_4{}^m \rceil$, $\frac{\min\{LX,LY\}}{\sqrt{m^d}} \le \hat{D}(H(\vec{u}),H(\vec{v})) \le \frac{\max\{LX,LY\}}{\sqrt{m}}$. □

C. Routing Sensory Data to its Destination

For each grid $\vec{u}$ in the *SDS* of a given WSN, it has a position (x, y), computed by *HilbertMap*, in the 2-dimensional area covered by the WSN. Since all the data items in $\vec{u}$ will be routed to the sensor node nearest to (x, y), (x, y) is the *destination* of all data items in $\vec{u}$. Thus each data item has a destination position. The method to route a data to its destination consists of two parts.

Part one (routing). If node j collects a d-dimensional data item $\overrightarrow{S_j^i}=(s_{j1}{}^i, s_{j2}{}^i, ..., s_{jd}{}^i)$ whose destination is (x,y), i.e., $\overrightarrow{S_j^i}$ is contained in a grid $\vec{u}=(u_1,u_2,...,u_d)$, and $H(\vec{u})=(x, y)$, node j uses a geographic routing algorithm GPSR [16] to route $\overrightarrow{S_j^i}$ to the *destination* node s nearest to (x,y).

Part two (storing). To reduce the storage space, each destination node s only stores a *grid feature* $s.GF(\vec{u})=\{gID, \overrightarrow{LS}, SS, Dens(\vec{u}), IDList\}$ for all the data items in a grid, and *IDList* is implemented by Bloom Filters [13]. If destination node s receives $\overrightarrow{S_j^i}$, it performs:

1. $s.GF(\vec{u}).\overrightarrow{LS} \leftarrow s.GF(\vec{u}).\overrightarrow{LS} + \overrightarrow{S_j^i}$,
2. $s.GF(\vec{u}).SS \leftarrow s.GF(\vec{u}).SS + \left(\overrightarrow{S_j^i}\right)^2$,
3. $s.GF(\vec{u}).Dens(\vec{u}) \leftarrow s.GF(\vec{u}).Dens(\vec{u}) + 1$, and
4. Hash j into $s.GF(\vec{u}).IDList$.

IV. In-Network Data Clustering Algorithm

Given a set of d-dimensional sensory data items $D=\{\overrightarrow{S_j^i}, \quad j=1,2,...,N\} \subseteq SDS$ collected by a WSN at epoch i, assume that the grid features of all the sensory data in D are already stored in their destination nodes by using the relocating method in Section III. The distributed clustering algorithm is to cluster the data in D by dealing with grid features. The algorithm consists of two phases. In the first phase, we merge connected grid features to the local cluster features of D at each destination node. In the second phase, we combine the connected local clusters to obtain the global cluster.

A. Generating Local Cluster Features

The first phase of the distributed clustering algorithm is to merge connected grid features to the local cluster features of D. When a clustering query is issued, it is flooded within the network. After receiving the clustering query, all the destination nodes that hold grid features

execute the distributed algorithm. Assume that node s is an arbitrary destination node that maintains grid features GF_1, GF_2, …, and GF_q of D. s may maintain a number of grid features. The reason is that *HilbertMap* can map different sensory data items, *e.g.*, $\overrightarrow{S_{j1}^{i}}$ in grid $\vec{w}$ and $\overrightarrow{S_{j2}^{i}}$ in grid $\vec{u}$, to different positions (x_w, y_w) and (x_u, y_u), but a destination node s may be the only node closest to (x_w, y_w) and (x_u, y_u), so s needs to maintain two grid features for $\vec{w}$ and $\vec{u}$. Node s generates local cluster features through the following two steps:

The first step is to convert each grid feature GF_i to cluster feature CF_i by $CF_i \leftarrow \{GF_i\}$.

The second step is to merge connected cluster features among CF_1, CF_2, …, and CF_q to local cluster features. The generation of local cluster features is equivalent to finding all the connected components in a graph T=(V, E), where V is the set of the converted cluster features, and there exists an edge <CF_i, CF_j>∈E if and only if CF_i and CF_j are connected. A connected component represents a local cluster feature. The generation of local cluster features is given in Algorithm 2.

Algorithm 2 ***Generating_Local_Clusters***

Input: CF_1, CF_2, …, and CF_q of D.
Output: Local cluster features of D.
Step 1. Construct a graph T=(V, E).
$V \leftarrow \{CF_1, CF_2, \ldots, CF_q\}$
$E \leftarrow \{<CF_i, CF_j> \mid CF_i$ and CF_j are connected.$\}$
Step 2. Merge connected cluster features.
1.Get an edge <CF_i, CF_j>∈E
2. If $Dens(CF_i \cup CF_j) \geq \lambda$ Then
(1)$CF \leftarrow CF_i \cup CF_j$ //Merge CF_i and CF_j
(2)To construct new graph T_{new}=(V_{new}, E_{new})
$V_{new} \leftarrow \{ V - \{ CF_i, CF_j \} \} \cup \{CF\}$
$E_{new} \leftarrow E - \{<CF_i, CF_j>\}$
For edges <CF_i, CF_k> and <CF_j, CF_l>∈ E_{new}
Convert <CF_i, CF_k> to <CF, CF_k>
Convert <CF_j, CF_l> to <CF, CF_l>
Endfor
(3) $T \leftarrow T_{new}$;
Step 3. If E is not empty then repeat step 2 on T;
Else, node s gets all local cluster features in V.

Time complexity: The Algorithm 2 checks |E| edges to find a pair dense cluster feature <CF_i, CF_j>∈E, and combines the edge, then |E| is reduced by 1 until E is empty. The total number of dense cluster feature in E is n, the total time complexity is n.|E|.

B. Finding Global Cluster Features

The second phase of the distributed clustering algorithm is to combine the connected local clusters located on different nodes to get the global clusters. The process of getting global clusters consists of four basic steps. The first step is to send clustering requests from those nodes which hold cluster features. The second step is to receive, process, cluster requests, and return reply information. The third step is to receive and process reply information. The fourth step is the terminated condition of the algorithm. The pseudo code of finding global clusters is presented in Algorithm 3.

Algorithm 3 ***Finding_Global_Clusters***

Input: Local cluster features of D.
Output: Global cluster features of D.
Step 1. Sending clustering requests.
For each node s who has $s.CF_j$ with $Dens(s.\ CF_j) \geq \lambda$
1. Node s computes neighbors of CF_j as following:
$Neighbor(s.CF_j) = \bigcup_{GF(\vec{u}) \in s.CF_j} Nbr(\vec{u})$;
2.For all the grid features in $Neighbor(s.CF_j) - s.CF_j$, compute their positions in the 2-dimensional area by *HilbertMap* (H for short) as follow:
$L(s.CF_j) = \{(x_{\vec{u}}, y_{\vec{u}}) \mid \vec{u} \in Neighbor(s.CF_j) - s.CF_j; H(\vec{u}) = (x_{\vec{u}}, y_{\vec{u}})\}$
3.Node s sends clustering request packet $\{(x_s, y_s), s.CF_j\}$ to all nodes whose position are in $L(s.CF_j)$, where (x_s, y_s) is the location of node s. The nodes in $L(s.CF_j)$ are not far from node s, which is guaranteed by Theorem 1.
Step 2. Receiving, processing clustering requests， and returning reply information.
For each node t which receives a clustering request from node s, t performs the following actions:
1.Call *Generating_Local_Clusters* to merge cluster features in node t and $s.CF_j$.
2.If (there exists one $t.CF_i$ in node t, it can merge with $s.CF_j$) and $Dens(t.CF_i) > Dens(s.CF_j)$
then $Rply(t,s,j) \leftarrow \{<NULL, 1>\}$;
3.If (some cluster features $t.CF_{i_1}$, $t.CF_{i_2}$ ... $t.CF_{i_q}$ can merge with $s.CF_j$) and $Dens(t.CF_{i_k}) \leq Dens(s.CF_j), k = 1,2,...,q$
then $CF \leftarrow (\bigcup_{k=1}^{q} t.CF_{i_k}) \cup s.CF_j$; $Rply(t,s,j) \leftarrow \{<CF, 0>\}$;
4. If (there does not exist any cluster features, who can merge with $s.CF_j$), then $Rply(t,s,j) \leftarrow \{<NULL, 2>\}$
5. Node t sends $Rply(t,s,j)$ to s.
Step 3. Receiving and processing reply information.
For each node s which receives a reply $Rply(t,s,j) = \{<CF, state>\}$, s carries out the following operations:
Case 1: $state = 0$, $s.CF_j \leftarrow s.CF_j \cup CF$
Case 2: $state = 1$, $s.CF_j$ is *annexed* by $t.CF_i$. Thus, $s.CF_j$ cannot be merged by any cluster feature.
Case 3: $state = 2$, clustering request is rejected.
Step 4. If (all the state of reply equals 1 or 2) or cluster features exceed storage requirements, then s sends cluster features $\{s.CF_j \mid Dens(s.CF_j) \geq \lambda\}$ to the sink and this algorithm stops on node s; Else repeat Step 1.

The structure of clustering reply sent by node t to node s for $s.CF_j$ is denoted by $Rply(t,s,j) = \{<CF, state>\}$, where CF is the combination result and $state$ is a flag. If $state = 0$, it means cluster features at node t and $s.CF_j$ can be combined and $s.CF_j$ annexed cluster features at

node t. If $state = 1$, it means cluster features at node t and $s.CF_j$ can be combined and cluster features at node t annexed $s.CF_j$, in this case CF is null. If $state = 2$, it means cluster features at node t and $s.CF_j$ cannot be combined, i.e.，node t rejects the clustering request from node s. In this case, CF is null.

There is an extreme case of $Dens(t.CF_i) = Dens(s.CF_j)$, the clustering reply sent by t to s is $Rply(t,s,j) = \{< CF, 0 >\}$ and the clustering reply sent by s to t is $Rply(s,t,i) = \{< CF, 0 >\}$, where $CF = t.CF_i \cup s.CF_j$. This will cause duplicate clustering results in WSNs when a node receives reply structure and does combination of two cluster features according to reply structure. However, this will not affect the final clustering results as the combination of two cluster features is the union of two sets, and it is duplicate insensitive.

The annexation terminates when all the clustering requests from node s are rejected or $s.CF_j$ is combined by other cluster features or cluster features exceeds storage requirements, thus s will not send any clustering request, and s will send cluster feature $s.CF_j$ with $Dens(s.CF_j) \geq \lambda$ to the sink. Otherwise s repeats Step 1.

Finally, when the sink receives all the cluster features, it merges the cluster features that can be merged. The final cluster feature set $\{CF_j \mid Dens(CF_j) \geq \lambda\}$ at the sink is the clustering result.

Time complexity: The Algorithm 3 is a distributed algorithm. It starts from those nodes who have dense cluster features. The annexation is a spreading process among sensor nodes. The time of annexation is $O(\sqrt{N})$. Then, annexed node send the cluster features to the sin through $O(\sqrt{N})$, so the total time complexity is $O(\sqrt{N})$.

C. Extending to Large Scale Networks

In a sensor network with *N* nodes, it is possible to have an extreme situation where sensory data items from a source node *s* have to be routed to a very distant destination node in the network resulting in a communication cost of $O(\sqrt{N})$ at every epoch. To solve this problem, a network can be logically partitioned into $M \times M$ sub-networks. The in-network sensory data clustering algorithm can be applied in each sub-network. In this way, the communication cost can be reduced from $O(\sqrt{N})$ to $O(\frac{\sqrt{N}}{M})$. Clustering results of each sub-network can be delivered to the sink, and the sink can finish the combination of clustering results.

V. EXPERIMENTAL EVALUATION

A. Experimental Setup

We implemented our distributed clustering algorithm based on *HilberMap* (called *H-Cluster*) and *FastMap* (called *F-Cluster*) as well as the centralized clustering algorithms (called *C-Corner* and *C-Center*). *FastMap* [14] can also map a *d*-dimensional grid into a 2-dimensional physical position such that the neighbour grids are stored at different nodes that are close in the 2-dimensional area. For *H-Cluster*, *F-Cluster,* and *C-Corner*, the sink is placed at the bottom-left corner of the network. For *C-Center*, the sink is placed at the center of the network.

Simulator. We choose ns-2 (ns-allinone-2.28[17]) as our network simulator. Table I lists all parameters.

MAC Protocol. The IEEE 802.15.4 is adopted.

Routing protocol. We implemented a greedy routing protocol which is similar to GPSR in *ns*-2. In order to save energy, nodes in our protocol do not send messages periodically to report their positions.

TABLE I
SIMULATION PARAMETERS USED IN NS-2

Parameters	Values
Sleeping power in watts	0.035W
Transmition power in watts	0.66W
Initial energy per node in joules	10000J
Reception power in watts	0.4W

Topology, radio range, and node density. We used CMU's version of *setdest* tool in *ns*-2 to randomly generate five scenarios. Nodes were uniformly deployed with network sizes ranging from 100 nodes to 500 nodes. Each node has a radio range of 10*m*. For the results presented here, each node has an average of 6 nodes within its normal radio range.

Synthetic dataset. The synthetic dataset is obtained from a synthetic data generator developed by us. The generator controls the structures and the sizes of the data sets, such as the number of records and the number of attributes. The type of each attribute is double and the range values is [0, 1].

Real Dataset: real sensory data collected from 54 Mica2 motes by the Intel Research Lab at Berkeley [18] during 600 epochs. This dataset was collected within an epoch of about 31 seconds. We limit our discussions to four of the attributes sensed by these motes, i.e., temperature, humidity, light, and voltage.

B. Mapping Quality

We compare the mapping qualities of *HilbertMap* and *FastMap* [14]. *HilbertMap* and *FastMap* are implemented in C++ on a PC, which has a 1700MHz processor and 512M RAM.

To evaluate mapping quality, we use the stress function [14] defined by Formula 5.1 and the average number of hops from source nodes to destination nodes as measurements. The stress function is given as follows:

$$stress^2 = (\sum_{\vec{u},\vec{v}} (\hat{D}(H(\vec{u}),H(\vec{v})) - D(\vec{u},\vec{v}))^2) / (\sum_{\vec{u},\vec{v}} (D(\vec{u},\vec{v}))^2) \quad (5.1)$$

where $D(\vec{u},\vec{v})$ is the Euclidean distance between $\vec{u}$ and $\vec{v}$ in the *SDS* and $H(\vec{u})$ is the result of *HilbertMap* on $\vec{u}$. $\hat{D}(H(\vec{u}),H(\vec{v}))$ denotes the Euclidean distance between $H(\vec{u})$ and $H(\vec{v})$ in the 2-dimensional area.

Small $stress^2$ implies mapping algorithms can preserve the distance between two sensory data items in an *SDS* as same as possible after they are mapped to a physical space. Since the grids are obtained by partitioning every dimension into *m* intervals with equal length, we can

obtain m^d grids. Therefore, the important factors that affect the stress function are parameter m and the dimensionality d of an *SDS*.

Fig. 3 shows the *stress*^{2} for the two mapping algorithms, as a function of the number of intervals on each dimension. When $4 \le m < 64$, the *stress*^{2} of *HilbertMap* is less then *FastMap*, i.e., *HilbertMap* is better than *FastMap*; when $m > 64$, the *stress*^{2} of the two mapping algorithms are the same. From Theorem 1 we know that if two grids $\vec{u}$ and $\vec{v}$ are neighbors, then the Euclidean distance between their mapped positions based on *HilbertMap* satisfies the following inequation:

$$\frac{\min\{LX, LY\}}{\sqrt{m^d}} \le \hat{D}(H(\vec{u}), H(\vec{v})) \le \frac{\max\{LX, LY\}}{\sqrt{m}}$$

When m increases, if $\vec{u}$ and $\vec{v}$ are neighbors, $\hat{D}(H(\vec{u}), H(\vec{v}))$ decreases in a polynomial manner. This can reduce stress function value to some extent.

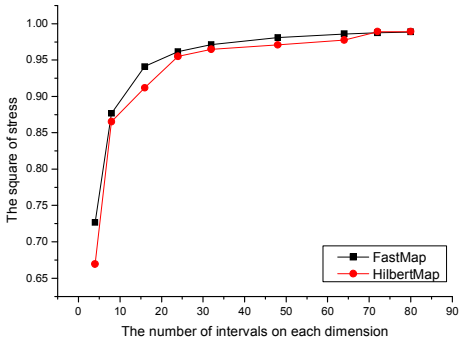

Fig. 3 *stress*^{2} vs. m for d =3

Fig. 4 shows the *stress*^{2} for the two mapping algorithms as a function of the dimensionality of an *SDS*. When the dimensionality d increases, the *stress*^{2} of *HilbertMap* is less than *FastMap*, i.e., *HilbertMap* is better than *FastMap*. When d increases, if $\vec{u}$ and $\vec{v}$ are neighbors, $\hat{D}(H(\vec{u}), H(\vec{v}))$ decreases in an exponential manner. This can reduce stress function value to a great extent as d is an exponent of m.

Fig. 5 illustrates that the average number of hops from source nodes to destination nodes using different clustering algorithms. While N increases, for *C-Center* and *C-Corner*, the average number of hops from source nodes to destination nodes increases as well. The reason is that when the sink is placed at the center or the corner of the network, the average number of hops is $O(\sqrt{N})$. Therefore, for *C-Center* and *C-Corner*, the average number of hops from source nodes to destination nodes increases when network size N increases. In this implementation, 200 is set as the threshold for partitioning the network into sub-networks. When network size N ($N \le 200$) increases, for *H-Cluster* and *F-Cluster*, the average number of hops from source nodes to destination nodes also increases. For network size N ($N > 200$), in the experiment we partitioned sensor network into 2×2 sub-networks, and *H-Cluster* and *F-Cluster* were applied in each sub-network. In this way, the average number of hops from source nodes to destination nodes decreases as discussed in *C* of Section IV.

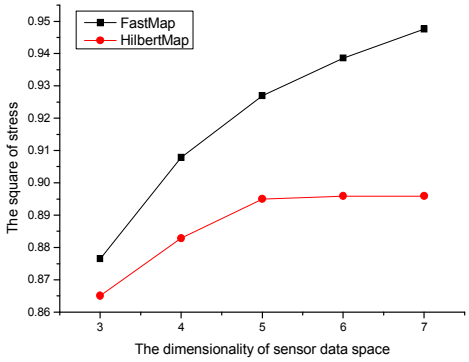

Fig. 4 *stress*^{2} vs. d for m=8

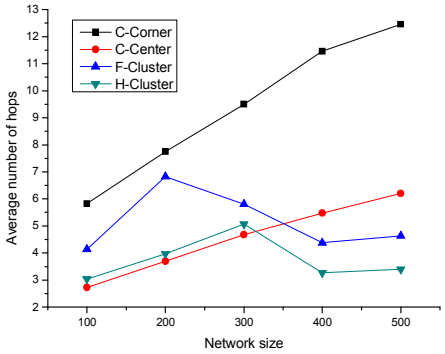

Fig. 5 Average hops vs. N

C. *Sensory Data Loss Rate*

In the simulation, we found that when a huge amount of sensory data is delivered, some data may be lost due to limited bandwidth. We define sensory data loss rate as the ratio of the amount of lost sensory data over the total amount of data sent by sensor nodes. It is obvious that if data loss rate is high, some information is lost and this may greatly affect clustering quality.

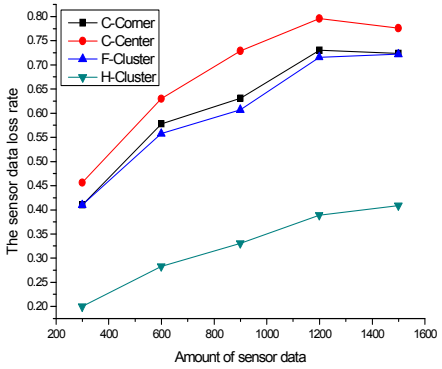

Fig. 6 The amount of data vs. the data loss rate

First, we investigated the effect of amount of sensory data on sensory data loss rate. We used CMU's version *setdest* to generate a topology with 300 nodes. The number of sensory data generated by each node varies from 1 to 5. Fig. 6 shows that sensory data loss rate goes

up as the amount of sensory data increases. *H-Cluster* has the smallest loss rate. The sensory data loss rate of *F-Cluster* is smaller than that of *C-Center* and *C-Corner*. The reason is that sending all data to a single destination node results in a bottleneck and data collisions. In *H-Cluster* and *F-Cluster*, there are more destination nodes compared with the centralized algorithms. Since *H-Cluster* has less hops between source and destination than *F-Cluster* does, the sensory data loss rate of *H-Cluster* is smaller than that of *F-Cluster*.

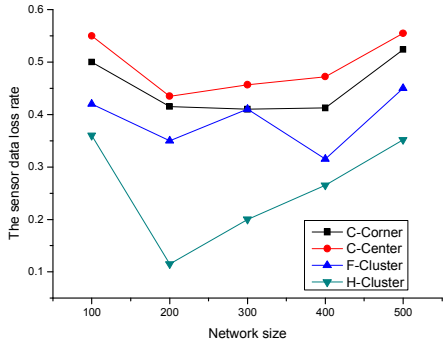

Fig. 7 The sensory data loss rate vs. *N*

Next, we studied the effect of network size on sensory data loss rate using different topologies. In each topology, every sensor node only generated 1 sensory data item. From Fig. 7 we find that sensory data loss rate is irregular, but for every fixed topology, the loss rate of *H-Cluster* is the smallest. The sensory data loss rate of *F-Cluster* is also smaller than that of *C-Center* and *C-Corner*. The data loss rate first goes down and then grows up as the network size is increased. This because if the network size is small, the destination nodes are more close, when data arrive at destination nodes would result in a bottleneck and data collisions. If the network size appreciably extends, it can properly relax collisions. However, if the network size is large, the amount of sensory data is huge, the data loss rate grows up as the network size is increased.

D. Energy Efficiency

The primary energy consumption in a WSN is for radio communication [9]. In our simulations, due to limited bandwidth some sensory data is lost. It results in such situation in ns-2: when sensory data loss rate increases, the energy consumption reduces; otherwise when sensory data loss rate decreases, the energy consumption increases. Therefore, using the total amount of energy spent on communication as the primary yardstick for measuring the cost of a clustering algorithm is not appropriate. We define energy efficiency as follows: Energy efficiency=(1-sensory data item loss rate)/(the total amount of energy spent on communication). If a clustering algorithm spends small energy to process much more sensory data, i.e., if the energy efficiency is large, then the algorithm is better.

First, we investigated the effect of network size on energy efficiency. We used CMU's version *setdest* to generate five topologies. The number of epochs is 30. Fig. 8 shows that the energy efficiency decreases as network size increases because of the increase in the total amount of energy spent on communication. It is shown in Fig. 8 that *H-Cluster* has the largest energy efficiency. The energy efficiency of *F-Cluster* is better than that of *C-Center* and *C-Corner*.

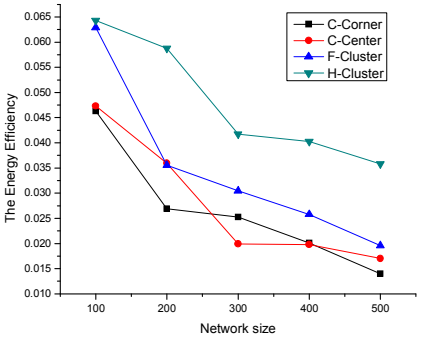

Fig. 8 *N* vs. the energy efficiency

Next, we studied the effect of the amount of sensory data on the energy efficiency. We chose a topology with 500 nodes. The number of epochs varies from 10 to 30 with an increment of 5. Fig. 9 shows that the energy efficiency decreases as the amount of sensory data increases because of the increase in the total amount of energy spent on communication. Another reason is that the energy efficiency is correlative with sensory data loss rate. When the amount of data increases, the sensory data loss rate also increases, and the energy efficiency reduces. From Fig. 9 we know that *H-Cluster* has the largest energy efficiency.

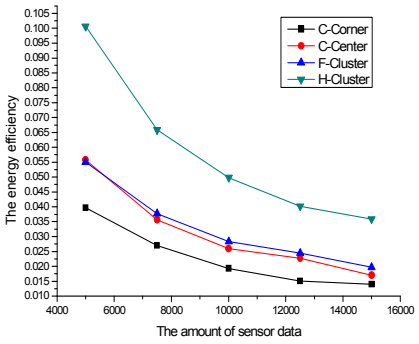

Fig. 9 Data vs. the energy efficiency

E. Clustering Quality

Clustering quality is an important criteria for evaluating clustering algorithms. We regard the clustering result of the centralized clustering algorithm CLIQUE [19] in the ideal situation (called *C-Ideal*) as an optimal clustering result. In the ideal situation, all sensory data can be delivered to the sink and there is no lost sensory data. To evaluate a clustering algorithm's clustering quality, we use the similarity between the results obtained by this algorithm and the results obtained by *C-Ideal* as a measurement.

Assume that we find *cn* clusters *C-ideal*={$C_{\text{c-ideal}}(1)$, $C_{\text{c-ideal}}(2)$, …, $C_{\text{c-ideal}}(cn)$} using *c-ideal*. $C_{\text{c-ideal}}(i)$ denotes the i^{th} cluster found by *C-Ideal*, where *i* denotes the number of clusters. Similarly, when we execute an algorithm such as

C-Corner, *C-Center*, *F-Cluster,* or *H-Cluster*, we may find *cm* clusters *C-algo*={$C_{\text{algo}}(1)$, $C_{\text{algo}}(2)$, …, $C_{\text{algo}}(cm)$}. $C_{\text{algo}}(j)$ denotes the j^{th} cluster found by an algorithm *algo*, where *algo* denotes an algorithm such as *C-Corner*, *C-Center*, *F-Cluster,* or *H-Cluster*. Algorithm 4 *ReassignClusterNumber* can adjust cluster numbers to *cm* clusters found by algorithm *algo,* and it can guarantee that $C_{\text{c-ideal}}(i)$ and $C_{\text{algo}}(i)$ have the same maximum number of sensory data items. As a supplement, if *cm*<*cn*, *cn-cm* empty clusters $C_{\text{algo}}(cm+1)$, $C_{\text{algo}}(cm+2)$, …, $C_{\text{algo}}(cn)$ are added to *C-algo*. If *cm*>*cn*, *cm-cn* empty clusters $C_{\text{c-ideal}}(cn+1)$, …, $C_{\text{c-ideal}}(cm)$ are added to *C-ideal*.

Algorithm 4 ***ReassignClusterNumber***

Input: *C-ideal*={$C_{\text{c-ideal}}(1)$, $C_{\text{c-ideal}}(2)$, …, $C_{\text{c-ideal}}(cn)$} and *C-algo*={$C_{\text{algo}}(1)$, $C_{\text{algo}}(2)$, …, $C_{\text{algo}}(cm)$}

Output: {$C_{\text{algo}}(1)$, $C_{\text{algo}}(2)$, …, $C_{\text{algo}}(cm)$} are reassigned cluster numbers and $C_{\text{c-ideal}}(i)$ and $C_{\text{algo}}(i)$ have the same maximum number of sensory data items.

1.*Clu-ideal*←*C-ideal*; *Clu-algo*←*C-algo*;

2.**While** $Clu-ideal \neq$ NULL or $Clu-algo \neq$ NULL

(1). Find $C_{\text{c-ideal}}(x)$ and $C_{algo}(y)$ such that

$$|C_{\text{c-ideal}}(x) \cap C_{algo}(y)| = \max_{1\le i\le cn} \max_{1\le j\le cm} |C_{\text{c-ideal}}(i) \cap C_{algo}(j)|$$

(2). $C_{algo}(y)$ is reassigned cluster number x.

(3). $C_{\text{c-ideal}}(x)$ is dropped from $Clu-ideal$.

(4). $C_{algo}(y)$ is dropped from $Clu-algo$.

Endwhile

TABLE II
CLUSTER RESULTS OF CENTRALIZED ALGORITHM

	C-Ideal	***C-Corner***	***C-Center***
NCF	30	30	30
CMAX	*n*=5025 *r*=0.621	*n*=2594 *r*=0.620	*n*=2658 *r*=0.621
CMIN	*n*=12 *r*=0.622	*n*=5 *r*=0.628	*n*=5 *r*=0.616
MAXR	*n*=273 *r*=0.759	*n*=156 *r*=0.763	*n*=152 *r*=0.754
MINR	*n*=531 *r*=0.371	*n*=222 *r*=0.370	*n*=216 *r*=0.372
TNSD	36196	18441	18976

TABLE III
CLUSTER RESULTS OF DISTRIBUTED ALGORITHM

	F-Cluster	***H-Cluster***
NCF	30	30
CMAX	*n*=2379 *r*=0.622	*n*=2638 *r*=0.621
CMIN	*n*=3 *r*=0.606	*n*=7 *r*=0.609
MAXR	*n*=147 *r*=0.752	*n*=151 *r*=0.760
MINR	*n*=268 *r*=0.365	*n*=296 *r*=0.370
TNSD	18674	19588

The clustering algorithm with the best clustering quality will maximize the following parameter:

$$Sim(algo) = \frac{1}{\max\{cn, cm\}} \sum_{i=1}^{\max\{cn,cm\}} |C_{\text{c-ideal}}(i) \cap C_{algo}(i)| \quad (5.2)$$

where $Sim(algo)$ denotes the average similarity of the clustering results obtained by the algorithm *algo* and *C-Ideal*. $|C_{\text{c-ideal}}(i) \cap C_{algo}(i)|$ denotes the similarity of cluster *i* found by *C-Ideal* and cluster *i* found by algorithm *algo*. We run *C-Ideal* on the Intel Lab Sensory dataset and found 30 clusters. Then, we run *C-Corner*, *C-Center*, *F-Cluster,* and *H-Cluster*, and show their clustering results in Table II and Table III.

In Table II and III, CMAX is the cluster with the maximum number of sensory data items. CMIN is the cluster with the minimum number of sensory data items. MAXR is the cluster with the maximum radius. MINR is the cluster with the minimum radius. TNSD is the total number of sensory data items. NCF is the number of discovered clusters. *n* is the number of sensory data items in a cluster. *r* is the radius of a cluster.

Finally, we calculated *Sim*(*C-Corner*), *Sim*(*C-Center*), *Sim*(*F-Cluster*), and *Sim*(*H-Cluster*), and they are shown in Table IV. From Table IV we know that our distributed clustering algorithm outperforms the centralized algorithms. The main reason is that clustering quality is relevant to sensory data loss rate. *H-Cluster* has a low sensory data loss rate, thus this can guarantee that its average similarity is as maximum as possible. The reason of low sensory data loss rate is that using *HilbertMap* the communications for clustering mostly occur among close sensor nodes.

TABLE IV
THE AVERAGE SIMILARITY

Sim(C-Ideal)	***Sim(C-Corner)***	***Sim(C-Center)***
1206.53	526.89	536.46
Sim(F-Cluster)	***Sim(H-Cluster)***	
533.54	599.97	

VI. RELATED WORK

Data mining for WSNs is a new emerging research area. Much work has been done on discovering frequent patterns and detecting outlier. For discovering frequent patterns in WSNs, [20] explored the use of in-network data mining techniques to discover frequent event patterns and their spatial and temporal properties. [21] studied the problem of mining frequent value sets from a large sensor network. Regarding detecting outlier in WSNs, [22] studied the problem of unsupervised outlier detection in WSNs and developed an in-network algorithm with a communication load proportional to the outcome. [23] proposed a framework to compute an approximation of multi-dimensional data distributions in order to identify either distance-based or density-based outliers. [24] studied localization of mobile objects in WSNs and proposed a probabilistic semi-supervised data mining approach to reduce calibration effort and increase tracking accuracy. [10] proposed a spatial clustering that partitions the network into a set of spatial regions. However,

existing works on mining in WSNs do not consider the problem of in-network real time clustering of sensory data.

In this paper, we propose a mapping algorithm based on the *Hilbert Curves*, *HilbertMap*, to map a *d*-dimensional sensory data space into a two-dimensional area covered by a WSN. Through this mapping, sensory data items that are close in the data space are first mapped to sensors that are geographically close. *HilbertMap* is a hash function essentially. Hashing has been well-studied in the area of sensor networks[25, 26]. *HilbertMap* is quite different from these hashing functions. First, these works are targeted towards answering range queries, and *HilbertMap* is targeted towards sensory data clustering. Second, these works do not guarantee that sensory data items that are close in the data space are first mapped to sensors that are geographically close, and *HilbertMap* ensures this (see theorem 1) and saves much energy while doing sensory data clustering.

VII. CONCLUSIONS

In this paper, we investigate the problem of sensory data clustering in WSNs. We propose a distributed clustering algorithm, *H-Cluster*. This algorithm relies on our proposed mapping algorithm, HilbertMap, to map multi-dimensional sensory data items to positions in a 2-dimensional area. This mapping algorithm preserves data locality. Therefore, it makes data communications for clustering highly localized in the network. We also design and implement two centralized clustering algorithms as well as *F-Cluster*, and compared the performances of these algorithms in ns-2. Simulation results show that *H-Cluster* outperforms the others in terms of both energy efficiency and clustering quality.

ACKNOWLEDGEMENT

This work was supported by NSFC grant No. 60803015 and 61070193. It partly supported by grants No. 20080430902, No. 2008RFQXG107, No. LRB08-021, No.1154Z1001, and No.GC09A109.

REFERENCES

[1] "Center for Environmental Sensing and Modeling, http://censam.mit.edu/research/res2/index.html," 2008.
[2] "Wireless Underwater Sensor Networks, http://www.sintef.no/content/page1____14699.aspx," 2008.
[3] A. Ailamaki, C. Faloutsos, P. Fischbeck, M. Small, and J. M. VanBriesen, "An Environmental Sensor Network to Determine Drinking Water Quality and Security," *SIGMOD RECORD*, vol. 32, pp. 47-52, 2003.
[4] C. Giansante and S. Pelini, "The Use of Geographic Information Systems in Sea and Freshwater Ecosystems," *Veterinaria Italiana*, vol. 43, pp. 507-512, 2007.
[5] S. Bandyopadhyay and E. J. Coyle, "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks," presented at Proceedings IEEE INFOCOM 2003, The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), San Franciso, CA, USA, 2003.
[6] O. Younis and S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach," presented at Proceedings IEEE INFOCOM 2004, The 23nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), Hong Kong, China, 2004.
[7] H. Chan, M. Luk, and A. Perrig, "Using Clustering Information for Sensor Network Localization," presented at International Conference on Distributed Computing in Sensor Systems (DCOSS 2005), Marina del Rey, CA, USA, 2005.
[8] J.Han and M.Kamber, *Data Mining-Concepts and Techniques*. San Fransisco, USA: Morgan Kaufmann Publishers, 2001.
[9] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "The Design of an Acquisitional Query Processor For Sensor Networks," presented at Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD 2003), San Diego, California, USA, 2003.
[10] A. Meka and A. K. Singh, "Distributed Spatial Clustering in Sensor Networks " presented at International Conference on Extending Database Technology (EDBT 2006), Munich, Germany, 2006.
[11] J. Yin and M. M. Gaber, "Clustering Distributed Time Series in Sensor Networks," presented at the 8th IEEE International Conference on Data Mining (ICDM 2008), Pisa, Italy, 2008.
[12] Y. Endo, R. Murata, H. Toyoda, and S. Miyamoto, "L1-Norm based Fuzzy Clustering for Data with Tolerance," presented at IEEE International Conference on Fuzzy Systems (ICFS 2006), Vancouver, BC, Canada, 2006.
[13] B.H.Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, pp. 422-426, 1970.
[14] C. Faloutsos and K.-I. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets," presented at Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data(SIGMOD 1995), San Jose, California, 1995.
[15] J.K.Lawder, "Calculation of Mappings Between One and n-dimensional Values Using the Hilbert Space-filling Curve," School of Computer Science and Information Systems, Birkbeck College, University of London, London Research Report BBKCS-00-01 August 2000.
[16] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," presented at Proceedings of the sixth annual international conference on Mobile computing and networking(MOBICOM 2000), Boston, MA, USA, 2000.
[17] "The Network Simulator ns-2 http://berkeley.intel-research.net/labdata," 2008.
[18] S. Madden, "Intel Lab Sensor Data. http://berkeley.intel-research.net/labdata," 2003.
[19] R. Aggrawal, J. Gehrke, D. Gunopulos, and P.Raghawan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," presented at In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1998), Seattle, WA, 1998.
[20] K. Römer, "Distributed Mining of Spatio-Temporal Event Patterns in Sensor Networks," presented at International Conference on Distributed Computing in Sensor Systems Euro-American Workshop on Middleware for Sensor Networks (EAWMS / DCOSS 2006), San Francisco, USA, 2006.
[21] K. K. Loo, I. Tong, B. Kao, and D. Cheung, "Online Algorithms for Mining Inter-Stream Associations From Large Sensor Networks," presented at In Proceedings of the Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2005), Hanoi, Vietnam, 2005.
[22] J. W. Branch, B. K. Szymanski, C. Giannella, R. Wolff, and H. Kargupta, "In-Network Outlier Detection in Wireless Sensor Networks," presented at International Conference on Distributed Computing Systems (ICDCS 2006), Lisboa, Portugal, 2006.
[23] S.Subramaniam, T.Palpanas, D.Papadopoulos, V.Kalogeraki, and D.Gunopulos, "Online Outlier Detection in Sensor Data Using Non-parametric Models," presented at In Proceedings of International Conference on Very Large Data Bases (VLDB 2006), 2006.
[24] R. Pan, J. Zhao, V. W. Zheng, J. J. Pan, D. Shen, S. J. Pan, and Q. Yang, "Domain-Constrained Semi-Supervised Mining of Tracking Models in Sensor Networks," presented at Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2007), San Jose, California, USA, 2007.
[25] Y.-C. Chung, I.-F. Su, and C. Lee, "Supporting Multi-Dimensional Range Query for Sensor Networks " presented at 27th International Conference on Distributed Computing Systems (ICDCS 2007). , Toronto, ON, Canada, 2007.
[26] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-Dimensional Range Queries in Sensor Networks," presented at Proceedings of the 1st international conference on Embedded Networked Sensor Systems (SENSYS 2003), Los Angeles, California, USA 2003.