CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 0000; **00**:1–15
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Energy-efficient big data storage and retrieval for wireless sensor networks with nonuniform node distribution

Jinhai Xu[1], Songtao Guo[1] *, Bin Xiao[2] and Jing He[1]

[1] *College of Electronic and Information Engineering, Southwest University, Chongqing, 400715 China*
[2] *Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*

SUMMARY

Distributed data-centric storage in wireless sensor networks (WSNs) is considered as a promising big data storage approach, since it contributes to reducing the communication overhead inside the networks. However, most of the existing distributed methods rely on locating systems, which consume more energy, and assume that sensors are uniformly distributed, which is clearly not applicable for the scenarios with nonuniform sensor distribution. To address these issues, in this paper, we propose a big data storage and retrieval algorithm for wireless sensor networks with nonuniform node distribution, which aims at estimating the real distribution and the addresses of sensor nodes. In particular, we consider the data redundancy among neighbor nodes in the proposed algorithm and exploit a simple routing based on the algorithm. Experimental results show that our approach outperforms other approaches in terms of data querying efficiency and data loss rate. Copyright © 0000 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: Wireless Sensor Networks, Nonuniform Distribution, Data Storage and Retrieval, Big Data.

1. INTRODUCTION

Recently, with the development of the Information and Communication Technology (ICT), there has been an explosive growth in the volume of data. Thus, the concept of big data has emerged as a widely recognized trend, which is currently attracting much attention from government, industry, and academia [1]. The big data contains high volume, high velocity, and high variety information assets [2], which are difficult to gather, store, and process by using the existing technologies. Currently, although huge amounts of data have already been generated by the various services, such as social networks, cloud storage, network switches, and so forth, it is an indisputable fact that more and more data will be generated by sensors/RFID devices such as thermometric sensors, atmospheric sensors, motion sensors, accelerometers, and so on.

The data generated by an individual sensor may not appear to be significant, but numerous sensors in wireless sensor networks (WSNs) can produce a significant portion of the big data as they will be used extensively in many military and civilian applications, such as military target tracking, environment monitoring, health monitoring, and observing phenomena. In such applications, a number of important domains of human endeavor are becoming increasingly reliant on these remotely sensed data instead of the deployed sensors in the monitoring field. Therefore, it is necessary and critical to store the large volume and wide variety of the sensed data with less resource

*Correspondence to: Songtao Guo, College of Electronic and Information Engineering, Southwest University, Chongqing, 400715 China. E-mail: stguo@swu.edu.cn

cost and retrieve the big data as quickly as possible. However, in widely distributed WSNs (e.g. in schools, urban areas, mountains, and so forth), how to store and retrieve those sensed big data still remains a challenging and open issue since (i) the limited communication range may pose a challenge for big data storage and retrieval from all sensor nodes; (ii) the big data storage and retrieval will increasingly shorten the lifetime of sensors because each sensor node needs to relay a lot of data generated by tremendous number of surrounding sensors.

To achieve efficient data storage and retrieval in WSNs, there have been some existing approaches. Those schemes for storing and processing data can be classified into three categories: external storage (ES), local storage (LS), and data-centric storage (DCS). In the external storage [3, 4, 5], the sensed data are sent to the sink outside the sensing field without waiting for a user to send a query. Users can directly send their queries to the sink to achieve the interested data. This may consume a lot of energy and cause the unnecessary communication cost and bottleneck since not all of the data in the sink is useful for a user. Moreover, the ES approach may lead to unbalanced energy consumption of sensor nodes, because the sensor nodes close to the sink, namely relay nodes, will always deplete their energy rapidly due to carrying out heavy tasks. In the local storage [6, 7, 8, 9], sensor nodes store the sensed data locally. When a user requests a query, the sink will flood the query within the whole sensor network. The advantage of the scheme is to reduce the route cost from the sensor node to the sink. But it also increases the communication cost due to flooding queries. In the data-centric storage [10, 11, 12, 13, 14], an event name is hashed to a geographic location. All data with the same event name will be stored at the node closest to the geographic location. Hence, the queries with a particular event name can be directly forwarded to the node storing the data with the event name through geographical routing [10], thereby avoiding flooding. Unlike LS and ES, the DCS approach [10, 11, 12] is an energy-efficient data dissemination method. However, the DCS approach in [11] achieves both storage and query by hashing the event and its geographical routing depending on localization systems (e.g., GPS). In this case, if the event has already been known, it makes no sense to find the geographic location to get data through hashing the event. The approach proposed in this paper can avoid the problem by storing some simple information about the time and relative location, i.e., where and when to produce the event.

In this paper, to efficiently store and retrieve the sensed big data, we estimate the distribution of sensor nodes by clustering and achieve the energy balance among intra-cluster members by rotating the cluster head in a round-robin manner. We first handle the data by using an algorithm similar to the rejection hash. The produced data include the meta data, the identifier of sensor node and the time. Then we map the produced data to the clusters according to the distribution estimation. The goal of this processing is to get the information when and which sensor nodes ever generate the same data, and balance the storage load among sensor nodes. Furthermore, we propose a routing algorithm based on cluster, which can route the data to hash position efficiently. The advantages of the algorithm are that (i) it can balance the energy consumption of routing among clusters; (ii) the sensor nodes are not required to equip with localization systems, which can prolong the lifetime of network. In addition, based on the assumption that the proximity of node locations represents the similarity of the data between nodes, we propose the closest interpolation algorithm to reduce the data loss caused by node failures. Besides, we also present an energy-efficient range search algorithm and similarity search algorithm to quickly retrieve the interested data. Extensive experimental results demonstrate that the proposed scheme outperforms the previous data storage and retrieval schemes in terms of query efficiency, data loss rate and fault toleration.

The rest of this paper is organized as follows: Section 2 discusses related works about the data storage for WSNs. Our proposed scheme will be clarified deeply in Section 3. In Section 4, we evaluate the performance of the proposed scheme, and Section 5 draws the conclusions.

2. RELATED WORK

According to the storage position of data in a WSN, the previous data storage solutions can be classified into three categories: external storage, local storage and data-centric storage. In the following, we describe those categories of data storage, respectively.

DOI: 10.1002/cpe

External storage: External storage schemes in [5, 4, 3] rely on the sink, which is located outside of the sensor network, and responsible for storing and accessing all of the data generated by sensor nodes. If many data queries are issued from sensors inside the whole network, the centralized sink may become a bottleneck since sensing data must be sent back and forth between the sensors and sink, and the delay will be larger. The way that all data are sent to sink wastes more energy and reduces the lifetime of the network. In addition, the energy consumption is unbalance within the network, due to that the sensors close to the sink consume more energy. The external storage schemes would only be used in small-scale WSNs with lower data generation rates.

Local storage: In the local storage [6, 7, 8, 9], all sensed data are stored in the sensors generating the data or their nearby nodes. Since the data are dispersed anywhere in the network without any index that can point to the data, the data query must be flooded to all sensors nodes within the network, which will cause high transmission and energy cost. In [6], directed diffusion protocol was proposed to save the cost of data querying. In this protocol, nodes disseminate the messages of queries throughout the sensor network. The events that match a query will flow toward the query nodes along multiple paths. Finally, a query node chooses a high-quality path from multiple paths to route for future data.

Zhang et al.[8][7] proposed an index-based data dissemination to address the problem that a huge amount of sensed data are generated, but only a small portion of data are successfully queried in large scale sensor network. In this scheme, the sensed data are stored at the detecting or nearby nodes, and the location information of these storing nodes are pushed to some index nodes, and the index nodes for one event further form a ring surrounding the location which is determined by the event type. TTDD [9] is a two-tier data dissemination protocol that maintains a grid structure in which each grid has a grid sever. In TTDD, the source detecting a certain event floods the advertisement of the event to the network, and the sink interested in the event can send his queries directly to the source.

Data-centric storage: Most recent researches focus on distributed data storage schemes for WSNs [10, 11, 12, 13, 14]. These schemes map data to geographical locations with hash algorithm. The first DCS proposal in WSNs is Geographic Hash Tables (DCS-GHT) [12]. In DCS-GHT, the set of sensors selected to store data is computed by means of a hash function. This function returns a pair of geographic coordinates fitting in the sensing area.

Maia et al.[13] proposed a distributed data storage protocol for heterogeneous WSNs with mobile sinks, called ProFlex, which consists of three phases: tree construction, important factor distribution, and data distribution. At the end of three phases, each node in the set of storage nodes will store a certain amount of data proportional to its importance factor. GEM [14] constructed a labeled graph spanning the network, which assigns addresses to the sensors rather than geographic coordinates so as to enable nodes performing efficient routing. DIM [15] used a geographic embedding index structure. It recursively divides the plane so as to assign address to sensors, and then hashes metadata to that address. Shen et al. [16] proposed a distributed data storage (SDS) scheme with spatial-temporal similarity, which provides an efficient spatial-temporal and similarity data searching based on location, time and locality-sensitive hash [17]. Albano et al. [18] provided an approach dealing with data storage for non-uniform distribution WSN. This approach first estimates the real network distribution and then disperses data according to the estimated network distribution. The network is divided into square region regularly. Compared to those previous works, we divide the network into regions with general shape by using clustering protocols, which makes the network distribution estimation closer to the actual distribution.

3. DESIGN OF PROPOSED ALGORITHM

Our storage scheme is based on non-uniformly distributed network and inspired by the idea of Locality Sensitive Hashing (LSH) [17], Local Storage (LS) [6, 7, 8, 9], and Similarity Search [19]. We combine LS with DCS schemes, aiming at reducing the transmission overhead, energy consumption in process of routing data, and preventing the flooding query in the whole network. In the meanwhile, we can also retrieve the similar data with the nature that the physical distance

DOI: 10.1002/cpe

between nodes represents the similarity of data stored in the nodes, reducing the search latency and achieving the search efficiency.

In this section, we will propose a big data storage and retrieval algorithm for non-uniform distributed WSNs. In this algorithm, we will first estimate the actual sensor distribution, then accomplish load balancing in terms of storage and routing by using hash function and routing protocol. Hash function makes the amount of data stored in each cluster proportional to the number of nodes in the cluster, and routing protocol makes the energy consumed by forwarding data from other clusters inversely proportional to the number of nodes in the cluster. Finally, we will implement data query with high efficiency.

3.1. Distribution Estimation

As the uneven distribution of sensors in the network may cause unbalanced load among the sensors, we first estimate the distribution of the sensors in the network by a clustering algorithm such as LEACH protocol [20]. In fact, other protocols can work as well. Without loss of generality, we assume that the sensing field is a rectangle and there are N sensors in the field, as shown in Fig.1(a). Each sensor node is equipped with two radio interfaces to adapt to different communication range. One is used to communicate with the nodes in a cluster. The other is applied to communicate with adjacent cluster heads or route messages to other cluster heads directly. The sensor network is divided into n clusters by LEACH protocol. Each cluster is assigned to an integer in the interval $[1, n]$ as its identifier. Moreover, the number represents the address of the cluster so that the cluster can be easily found, and is expressed in fixed-length binary form, such as 16 bits or two bytes. Then, the cluster head acquires the number of sensors belonging to its cluster through beacons, and broadcasts it to other cluster heads in the process of clustering so that each sensor is aware of this number. Cluster head computes the probability that the cluster is selected to store the data from other clusters, and all the probabilities form a distribution vector P, which is taken as an input of hash function stated in section 3.2.1. For example, for a cluster $i \in [1, n]$, and the number of members of cluster i, m_i, the probability of the cluster to be selected is $p_i = \frac{m_i}{N}$ and distribution vector can be given by $P = (p_1, p_2, \ldots, p_n)$. Therefore, the sensed data will be stored in each sensor by the processing of hash function according to the node distribution in the network so that load is balanced over the network.

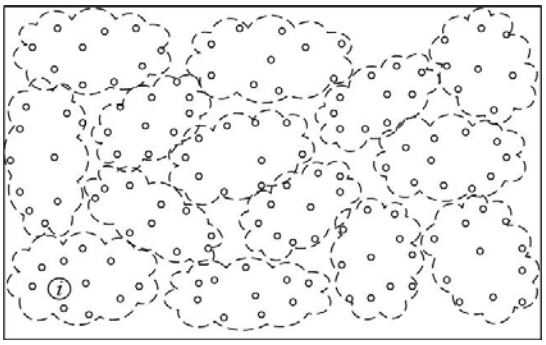
Node addressing: In order to better balance the intra-cluster load, clusters need to rotate cluster heads in a round-robin manner by the order of their identifiers. Node addressing is to assign the proper identifiers for each cluster and sensor nodes. After clustering, each cluster head knows the amount of the sensors in its cluster, and assigns a number/identifier to its cluster members consecutively according to the received signal strength so that data is uniformly stored in the members of the cluster within one hop. Fig.1(b) depicts the process of node addressing. During initialization, for a cluster i, the number of its cluster members is m_i, the identifier of the cluster head is zero, and each cluster member is assigned to a number $s \in [1, m_i - 1]$ according to the received signal strength. The stronger the signal strength is, the smaller the number becomes. After initialization, their identifiers are not changed any more, and the cluster head is elected in a round-robin manner. Besides, when a sensor node is selected as the cluster head, the node takes responsibility for data storage besides forwarding data and balancing load within the cluster. The advantage of this node addressing is that it can route data without GPS or other localization systems.

3.2. Storage and Retrieval

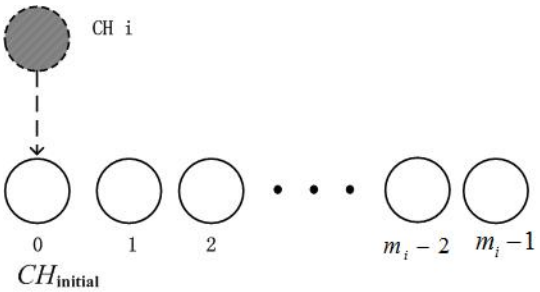
In the multi-dimension space, the distances among nodes can reflect the similarity among the data. The closer the distances, the higher the similarity between their data. Moreover, from the producers point of view, it is most effective to store data locally. Thus, the sensed data are stored in the node generating data so that similarity search and range search can be build according to the similarity of data stored in nodes.

3.2.1. Determination of storage location In previous literature [16],[12], data-centric storage schemes adopt standard hash function to determine the data storage location. But the standard hash

Prepared using cpeauth.cls
DOI: 10.1002/cpe



(a) A WSN with nonuniform distribution



(b) Node addressing

Figure 1. The distribution estimation and node addressing

function maps the data to each zone uniformly, regardless of how many sensors in this zone. This leads to a problem that for the same size of zones, the zone with more sensors is assigned less data. Algorithm 1 describes the hashing module adopted in this paper.

Algorithm 1 Hash algorithm

Require: a meta data k, a vector $P = \{p_1, p_2, \cdots, p_n\}$;
Ensure: a cluster's identifier ID,$ID \in [1, n]$;
  $i = 0$;
  while (true) **do**
    $(ID, z) = Hash(k + i)$;
    $i \quad inc$;
    if $z < p_{ID}$ **then**
      return ID;
    end if
  end while

This algorithm is intuitively a pseudo-random number generator similar to that in [18], which takes the sensor distribution vector P and the meta data k as input and the cluster's identifier (ID) as an output. It uses a strategy similar to the rejection method [21] to generate random numbers following the probability distribution in a limited domain. A dual group (ID, z) is generated uniformly and at random, where $ID \in [1, n]$ is cluster's number/identifier, $z \in (0, 1)$. If $z < p_{ID}$, then ID is returned. Otherwise, other dual groups (ID, z) are generated until $z < p_{ID}$. p_{ID} denotes the probability of the number ID to be returned.

In addition, by this algorithm, we can also analyze the number of events stored in a cluster. For example, for a cluster CH_i, we can calculate the total number n_i of events by using the following

Prepared using cpeauth.cls
DOI: 10.1002/cpe

Table I. The format of packet storage

$SinkId$	t	$SenseId$	$ClstrId$	$DestClstrId$	$SenseData$

equation:

$$\begin{aligned} n_i &= \sum_{j=1}^{j=n} n_{i,j} = \sum_{j=1}^{j=n} m_j \cdot p_i \\ &= p_i \sum_{j=1}^{j=n} m_j = p_i \cdot N \\ &= m_i \end{aligned} \tag{1}$$

where $n_{i,j} = m_j \cdot p_i$ represents the number of events which are produced by cluster j and stored in cluster i. Equation (1) shows that each sensor node in cluster i can be assigned one event on average which guarantees load balance.

To elaborate, after an event is sensed, sensors process the data by using Bloom Filter (BF) [22], where sensors can filter the redundant data and prevent transmitting and routing the same data to the destination cluster determined by Algorithm 1 so as to reduce the consumption of energy in network and improve the utilization of storage space. In the meanwhile, the fixed-length bit vector, which is an output of BF, is taken as the digest information of sensed data. Then, the digest information is packaged by the following tuple table I, where $SenseData$ denotes the digest value of attributions, $ClstrId$ is the identifier of the cluster producing the sensed data, $SenseId$ is the identifier of the sensor sensing the data in cluster $ClstrId$, and t indicates the time when the data is sensed, $DestClstrId$ is the identifier of the destination cluster. Compared to the existing methods, our storage approach is only storing the digest information of the data from other sensor nodes. Moreover, compared to the sensed data, the size of the tuple is very small. Besides, the sensed data is stored locally. Hence, the storage approach can save the storage space and reduce the routing cost from the source node to the hashed position.

Once a node senses data, the node will map its position information to other nodes according to the sensed data. Then by querying, we can know that when and which nodes ever generate the data, so as to facilitate further monitoring. In order to reduce the frequency of storage and save the energy, when the data produced by source nodes have fractional parts, the data are rounded off and their hash values are computed by source nodes. Since the data changes over time, the position information of a sensor node may be stored in many nodes. When the address of a node is stored in the same node, the identifier can be ignored and only the time t needs to be stored, which aims at improving the utilization of storage space. Here, we only take into account the temporal redundancy of the sensed data, instead of its spatial redundancy.

3.2.2. Routing for data storage In this paper, we assume that nodes are not equipped with any localization systems, and geographical coordinates of node positions are unknown. Thus, we propose a cluster-based routing algorithm, similar to the routing algorithm in [23],[24]. In wireless sensor networks, it is difficult to calculate the size of a cluster because the distribution of clusters is irregular and the routing path is random. From section 3.1, we know that the position of nodes consists of the identifier of the cluster and the relative addresses of cluster member nodes, and each cluster head maintains a routing table about neighbor clusters, as shown in Table II. In Table II, the first column represents the neighbor clusters of a cluster, the second column shows the probability of neighbor clusters of cluster i in Fig.1(a) to be selected as the next hop relay cluster, which is calculated by the equation (2):

$$P_{i,j} = 1 - \frac{p_j}{\sum_{j \in V_{i,N}} p_j}. \tag{2}$$

Therefore, when a sensor node has data to transmit, the node sends the data to its cluster head, then the cluster head will first match the destination identifier with its own identifier. If the match

DOI: 10.1002/cpe

Table II. Routing table of neighbor clusters of cluster i in Fig.1(a)

neighbor cluster	**probability** $P_{i,j}$
$Cluster_{i-1}$	0.56
$Cluster_{i+1}$	0.32
$Cluster_{i+2}$	0.28
$Cluster_{i+3}$	0.36

succeeds, the cluster head will store the data in its cluster. Otherwise, the cluster head will match the destination identifier with the identifiers of its neighbor clusters. If the match is successful, the cluster head will directly send the data to the matched neighbor clusters. Otherwise, the cluster head chooses a cluster from its neighbor clusters as the next hop relay according to the probability in routing table as shown in Table II. The process continues until the destination identifier is found. The detailed algorithm is shown in Algorithm 2.

Algorithm 2 Routing Algorithm

1: $V_{i,N}$: the set of the neighbor clusters of cluster i, $CH_j \in V_{i,N}$;
2: $P_{i,N}$: the probability set of cluster $V_{i,N}$, $p_j \in P$;
3: compute $P_{i,j}$ by using the equation (2), $P_{i,j} \in P_{i,N}$;
4: **if** $ID_{CH} == ID_{hash}$ **then**
5: store packets in this cluster;
6: **end if**
7: **if** $ID_{hash} == CH_j, (CH_j \in V_{i,N})$ **then**
8: CH_i sends packets to CH_j to store;
9: return;
10: **else**
11: **while** (true) **do**
12: random z,$j \leftarrow 1$;
13: **if** $z < P_{i,j}$ **then**
14: CH_i sends packets to CH_j to forward;
15: $CH_i \leftarrow$ next hop CH_j;
16: update $V_{i,N}$ and compute $P_{i,N}$;
17: execute step 7;
18: **end if**
19: $j \leftarrow mod(j, Sizeof(V_{i,N})) + 1$
20: **end while**
21: **end if**

The approach proposed in this paper can balance the energy consumption in three aspects: data storage, inter-cluster routing and intra-cluster. In data storage, we scatter the data to the sensor nodes by using the distribution probability so that each sensor node has the same storage workload. The inter-cluster routing selects the cluster with the least cluster members as the next hop so as to balance the energy consumption among clusters. In the intra-cluster aspect, we achieves the balance between the cluster head and cluster members by rotating the cluster head in a round-robin manner so as to avoid some data loss caused by some overburdened sensors.

3.3. Data Retrieval

For multi-attribution query, each node can be represented as a point in multi-dimensional space, and these accessed nodes form a like sphere centered at the queried point when a query is operated.

3.3.1. Range Query A data item denoted by A consists of multiple attributes d. Data item A is represented by $A = \langle d_1, d_2, \ldots, d_n \rangle$. When an acquirer has a range query, $Q = \langle [d_{l,1} - d_{u,1}], [d_{l,2} -$

DOI: 10.1002/cpe

Table III. Format of query packet

$SinkId$	t	$DestClstrId$	Q'	R	$Similarity$

$d_{u,2}], \ldots, [d_{l,n} - d_{u,n}]\rangle$, the acquirer first achieves the mean value of attribute values and attribute ranges, respectively, i.e., $Q' = \langle d'_1, d'_2, \ldots, d'_n\rangle$, $R = \langle r_1, r_2, \ldots, r_n\rangle$, where

$$\begin{cases} d'_i = \frac{d_{l,i}+d_{u,i}}{2}, \\ r_i = \frac{d_{u,i}-d_{l,i}}{2}, \end{cases} \quad i = 1, 2, \ldots, n \tag{3}$$

The acquirer then computes Q' by the same hash function in section 3.2.1, and the data is packaged by the format in Table III. Furthermore, the query packet is routed to the corresponding cluster to get the information about the address of the sensor node producing data Q'. The nodes further route query packet to the source node and judge whether R contained in the packet is zero. If yes, the source node will return the sensing data to the acquirer directly; otherwise, the source node will start a range query by the following process:

1. The search will begin among nodes within one hop of source node, and the source node will find the closest and farthest nodes within one hop, respectively, according to the signal strength;
2. Closest node compares its data with that of source node. If $|d_{i,close} - d'_i| < r_i$ and $|d_{i,farther} - d'_i| < r_i$, the data within this hop will be transmitted directly to the query node and the search will be executed among nodes within next hop.
3. If $|d_{i,close} - d'_i| < r_i$ and $|d_{i,farther} - d'_i| > r_i$, the search is matched by flooding. Otherwise, the search is stopped.
4. The iterative process stops until the condition $|d_{i,close} - d'_i| < r_i$ is not satisfied.

The pseudo-code for the range query is shown in Algorithm 3.

Algorithm 3 Range Query

compute $Q = \langle [d_{l,1} - d_{u,1}], [d_{l,2} - d_{u,2}], \ldots, [d_{l,n} - d_{u,n}]\rangle$;
compute the mean value and range value of each attribute: $d'_i = \frac{d_{l,i}+d_{u,i}}{2}, r_i = \frac{d_{u,i}-d_{l,i}}{2} \quad i = 1, \ldots, n$;
hash the $Q' = \langle d'_1, d'_2, \ldots, d'_n\rangle$, gets the address of the sensor node producing data Q';
route the query packet to the address ;

```
1: if R → 0 then
2:    return the data stored in this node;
3: else
4:    flood the query within one hop ;
5:    if |d_{i,close} − d'_i| < r_i and |d_{i,farther} − d'_i| < r_i then
6:       return the data within this hop;
7:       return step 4;
8:    else
9:       if |d_{i,close} − d'_i| < r_i and |d_{i,farther} − d'_i| > r_i then
10:         match the data one by one;
11:      else
12:         stop search;
13:      end if
14:   end if
15: end if
```

3.3.2. Similarity Search Similar to range query above, a similarity query is processed by the same hash function to get location information and the acquirer further routes the information to the

DOI: 10.1002/cpe

source node. Then the source node floods the query to the sensors within one hop. If the data stored in these sensors satisfy the similarity requirements, i.e., being large or equal to the similarity value obtained by equation (4), it will be returned to the acquirer.

$$similarity = \frac{\sum_{i=1}^{n} B(A_1^{d_i}, A_2^{d_i})}{n} \tag{4}$$

where $B(A_1^{d_i}, A_2^{d_i})$ is a Boolean value function, returning 1 when $A_1^{d_i} = A_2^{d_i}$ and 0 otherwise. The pseudo-code for the similarity search is described in Algorithm 4.

Algorithm 4 Similarity Search

```
compute A = ⟨d_1, d_2, ..., d_n⟩, pre − sim;
hash the A and routes to the address ;
flood the query;
computes the sim with equation (4);
if sim ≥ presim then
    return the data stored in node, continue step 3 ;
else
    stop search;
end if
```

It is noting that not all of the property values in similarity search are equal and each property has an interval range. But part of them are always equal, thus we can calculate the similarity degree according to the number of properties with equal values.

3.3.3. Fault Tolerance When a node in the network fails or is damaged, data in the node may be lost. In order to provide fault tolerance, we can estimate the data approximately using the similarity of data stored in neighbor nodes. By the similarity of the data in neighbor nodes, the estimation can be implemented with the most close interpolation in (5), in which the data stored in failure nodes can be replaced by data in nodes closest to the failure nodes. The pseudo-code for the fault tolerance is depicted in Algorithm 5.

$$data(i) \approx data(j) \qquad j \in \{j | min(dist(i,j))\} \tag{5}$$

where $data(i)$ represents the data of failure node i, and $dist(i,j)$ represents the distance between node i and node j. j belongs the set of neighbors of node i.

Algorithm 5 Fault Tolerance

```
Initialization
V_i: the set of neighbor nodes of node i;
D_{i,j}: the set of the distance from node i to node j , j ∈ V_i;
node j ← minD_{i,j};
node i sends its address to node j ;
if node i fails and a query arrives then
    node j sends its data to the acquirer;
end if
```

4. PERFORMANCE EVALUATION

4.1. Simulation Setup

In this section, we evaluate the performance of the proposed nonuniform node distribution storage and search algorithm in section 3.

DOI: 10.1002/cpe

Table IV. List of Notations

Notation	Definition	Values
N	Number of nodes	400
n	Number of clusters	100
k	Dimension degree	3
Q	Number of queries	100
S_{sumary}	Size of node memory	20
c	Cost of forwarding a unit data	$10^{-6}J$

We compare the proposed algorithm with two storage and search algorithms such as Directed Diffusion(DD)[6] and GHT [12]. The reasons of choosing them as comparison objects instead of the works in [18, 16] are as follow: literature [18] used a geographical routing algorithm called GPSR to dealing with nonuniform data centric storage for WSNs, in which the sensed region is divided into many small square regions, and only the storage load is improved. Literature [16] utilized carpooling routing algorithm to achieve a distributed spatial-temporal similarity data storage, which is built on the basis of artificial division of the sensed area. In fact, it is very difficult to determine the optimal number of subregions and the exact division precision. However, our proposed algorithm does not divide the sensed area into small rectangle regions. Therefore, the algorithms proposed in [18, 16] are remarkably different from our algorithm, which makes it no sense to select them as comparison objects. By contrast, both Geographic Hash Table (GHT) in [12] and Directed diffusion (DD) [6] do not require the division of area and have better performance of search, which is similar to our work.

To this purpose, we have implemented the proposed algorithm in the MATLAB simulator. We assume that 400 sensor nodes are randomly and independently disseminated in a rectangular field $100 \times 100m^2$. These sensor nodes are divided into 100 clusters, each of which has 4 sensor nodes on average. The communication range and the initial energy of each node is $16m$ and $10J$ respectively. The simulation time is $600s$. The cost of forwarding unit data item is $10^{-6}J$. Each data item has 20 bytes and contains the time and the identifier of clusters and sensors. First of all, we divide the network into clusters by using LEACH protocol and analyse the error of data dispersed by using standard hash and our hash. Then, we evaluate the efficiency of range query and similarity search in terms of overhead and latency. The parameters are given in Table IV.

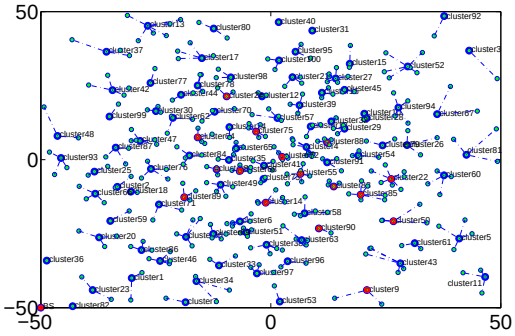
4.2. Error of Mapping Data to Clusters

In this subsection, as shown in Fig. 2(a), we consider non-uniform random distribution scenario with mean 0 and standard deviation 20. Fig. 2(b) depicts the error of mapping data in Gaussian distribution scenario. We represent the error using the Mean Square Error (MSE), a scalar quantifying the standard hash and our hash function. It is shown in Fig. 2(b) that the errors of standard hash function and our hash function decrease as the number of data hashed increases. But the error caused by our hash function is always less than that by the standard hash function. The reason for this situation is that standard hash function maps data to each cluster uniformly, which causes data loss or extra overhead; however, our hash maps the data to each cluster with a certain probability. As the amount of hashed data increases, the error decreases gradually and is close to an extremely small value. Fig.2(b) shows that our approach can disperse the data well regardless of the amount of hashed data.

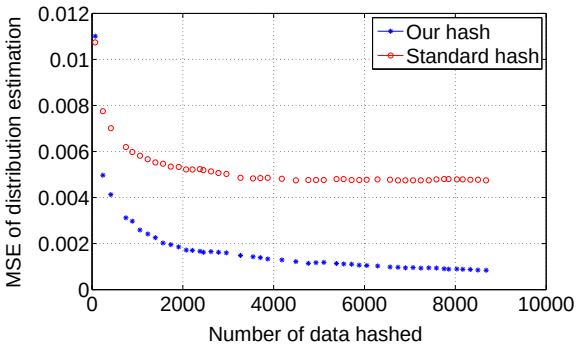
4.3. Query Efficiency

Range query is used to find data within a certain range in the clusters. We use the overhead and latency to test the performance of different methods. The overhead is the products of the number of hops and the cost of forwarding a unit packet within one hop, and the unit is mJ. The latency is time period between a query is issued and a response is received. Fig. 3(a) plots the overhead of range query by the different methods.It is not difficult to observe that (i) DD in [6] has much higher overhead than other algorithms due to its broadcasting for data querying; (ii) The overhead by GHT

DOI: 10.1002/cpe



(a) A Clustered WSN with Gaussian distribution



(b) The error of Gaussian distribution estimation

Figure 2. The error of Gaussian distribution estimation

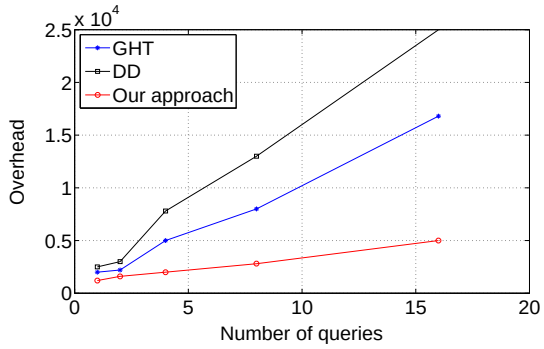
increases greatly as the number of queries increases. This is because GHT sends out more queries than our proposed schemes for every attribute corresponding to the range size. Fig. 3(b) shows that DD's latency increases dramatically because of congestion as querying traffic grows. Although GHT obtains the shortest path, its latency is affected by congestion caused by the increasing traffic. However, our approach achieves better performance, and traffic and congestion are reduced since it does not send too many queries as GHT does.

Fig. 4(a) compares the number of discovered data items with no less than 60 percents similarity in our approach and the number of actual data items in the system. It is shown that our approach can search out about 90 percent of such data items. Fig. 4(b) depicts the discovery rate of three approaches as similar degree grows. DD obtains a perfect performance due to its broadcasting queries. However, GHT only returns the data item with 100 percent similarity, since it only provides the exact matching. However, our approach achieves a discovery rate more than 80 percents while avoiding congestion caused by flooding. Thus, our method is superior to the DD and GHT in terms of overhead, latency and discovery rate.

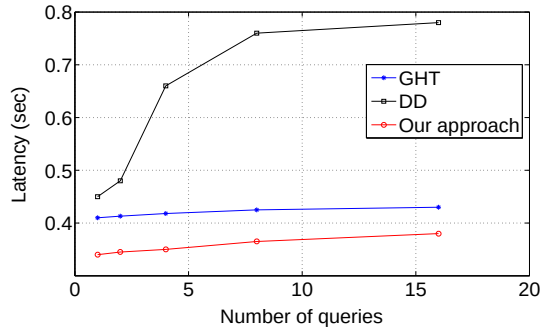
4.4. Data Loss Rate

In this subsection, data loss rate is defined as the number of unsuccessfully resolved queries divided by the total number of queries. This reflects the ability of fault tolerance. Fig. 5 shows the data loss rate with different percentages of failure sensor nodes. As expected, the data loss rate increases nonlinearly with the percentage of failure sensor nodes. The reason is that when the sensor node in charge of forwarding data fails, data item may be lost on the way to the sink node or its destination sensor node to be stored. Therefore, the responses to a query may not return all data satisfying query requirements.

DOI: 10.1002/cpe



(a) Overhead



(b) Latency

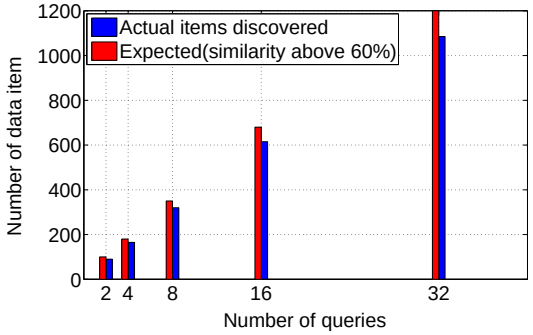
Figure 3. Performance of range querying

4.5. The lifetime of network

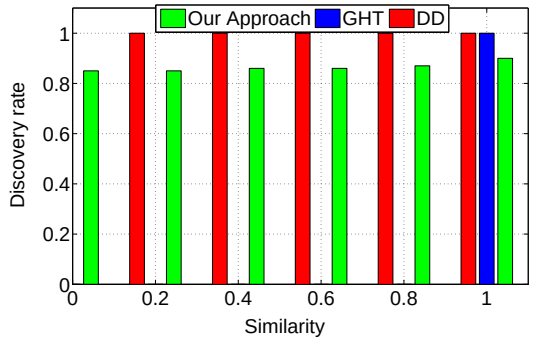
Currently, there is no precise definition on network lifetime. According to the definition given in [25], the network lifetime can be quantified by using the following three kinds of metrics: (1) the time taken from the deployment of the network to the death of the first sensor node; (2) the time when the certain percent of nodes die; (3) the time when all the nodes are dead in the network. In this experiment, the network lifetime is reflected through the changes of the power and number of alive nodes in the network. Fig. 6(a) shows that the total energy of the network almost varies with time linearly by our method and other two techniques, i.e., DD in [6] and GHT in [12]. This is because the number of hops of forwarding packets and the consumption of energy in routing within the same time is approximately equal, respectively. And it is not difficult to observe that (i) DD in [6] has much rapid energy consumption than other algorithms owing to its broadcasting; (ii) The remaining energy by our algorithm is more than that by GHT in [12]. The reason is that GHT sends out more queries than our proposed scheme according to the range size.

Fig. 6(b) depicts the changes of the number of alive nodes with time in the network by three approaches. It is clear that the alive nodes are least in the network by DD method compared to other two algorithms due to the rapid energy consumption of some special sensor does caused by its broadcasting; And our proposed scheme have more alive nodes than the GHT method. This is because the GHT routes the packets to the same destination with the same shortest path, which causes more energy consumption in some special sensor nodes and their earlier death. However, our approach is to choose a path from multiple possible paths so as to disperse the energy consumption among sensors. Thus, our approach is effective to prolong the network lifetime.

DOI: 10.1002/cpe



(a) Number of data items found



(b) Discovery rate

Figure 4. Performance of similarity querying

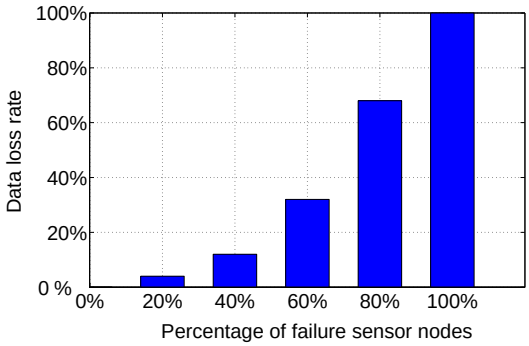
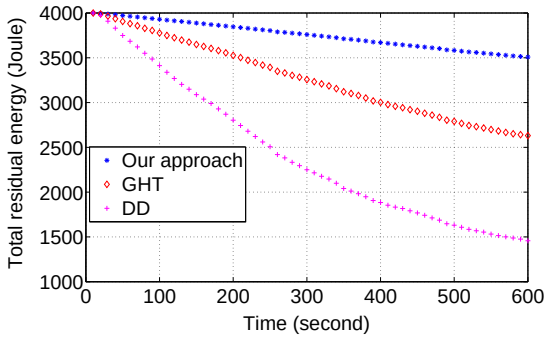


Figure 5. Capability of fault tolerance

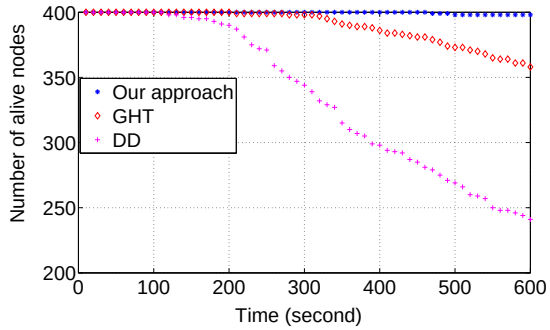
5. CONCLUSIONS

In this paper, we have explored a big data storage and retrieval scheme in WSNs with nonuniform node distribution. In particular, we takes the real network distribution into consideration by node distribution estimation so that the sensed data is uniformly stored in each node. Our storage scheme can reduce the energy consumption in terms of transmission by using the advantage of LS and DCS. And the proposed retrieval algorithm can improve the retrieval efficiency by reducing the data redundancy in closer nodes. In the meanwhile, without the aid of GPS, the proposed routing algorithm can efficiently route the queries or data. The simulation results show that the proposed

DOI: 10.1002/cpe



(a) The energy of network varies with time



(b) The number of alive nodes varies with time

Figure 6. The comparison of network energy efficiency

algorithm outperforms the existing algorithms in terms of data loss rate, data query efficiency and load balance.

Although the simulation results can effectively evaluate the proposed algorithm, a lot of WSN problems are caused by real-world deployment issues. Therefore, our future work is to implement and evaluate the proposed algorithm in a real deployment system of WSNs.

ACKNOWLEDGEMENT

This work was supported by the Fundamental Research Funds for the Central Universities (XDJK2013C094, XDJK2013A018, 2362014XK12), the National Natural Science Foundation of China (No. 61170248, 61373179, 61373178, 61402381) and Science and Technology Leading Talent Promotion Project of Chongqing (cstc2013kjrc-ljrccj40001).

REFERENCES

1. Labrinidis A, Jagadish HV. Challenges and opportunities with big data. *Proc. VLDB Endow.* Aug 2012; **5**(12):2032–2033, doi:10.14778/2367502.2367572. URL `http://dx.doi.org/10.14778/2367502.2367572`.
2. Sagiroglu S, Sinanc D. Big data: A review. *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, 2013; 42–47, doi:10.1109/CTS.2013.6567202.
3. Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. *Communications Magazine, IEEE* Aug 2002; **40**(8):102–114, doi:10.1109/MCOM.2002.1024422.
4. Pottie GJ, Kaiser WJ. Wireless integrated network sensors. *Commun. ACM* May 2000; **43**(5):51–58, doi:10.1145/332833.332838.
5. Yao Y, Tang X, Lim EP. In-network processing of nearest neighbor queries for wireless sensor networks. *Database Systems for Advanced Applications, Lecture Notes in Computer Science*, vol. 3882, Li Lee M, Tan KL, Wuwongse V (eds.). Springer Berlin Heidelberg, 2006; 35–49, doi:10.1007/11733836_5. URL `http://dx.doi.org/10.1007/11733836_5`.

6. Intanagonwiwat C, Govindan R, Estrin D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. *Proceedings of the 6th annual international conference on Mobile computing and networking*, ACM, 2000; 56–67.
7. Zhang W, Cao G, La Porta T. Data dissemination with ring-based index for wireless sensor networks. *Mobile Computing, IEEE Transactions on* July 2007; **6**(7):832–847, doi:10.1109/TMC.2007.1019.
8. Zhang W, Cao G, La Porta T. Data dissemination with ring-based index for wireless sensor networks. *Network Protocols, 2003. Proceedings. 11th IEEE International Conference on*, IEEE, 2003; 305–314.
9. Luo H, Ye F, Cheng J, Lu S, Zhang L. Ttdd: Two-tier data dissemination in large-scale wireless sensor networks. *Wireless Networks* 2005; **11**(1-2):161–175.
10. Liao WH, Chen CC. Data storage and range query mechanism for multi-dimensional attributes in wireless sensor networks. *Communications, IET* Oct 2010; **4**(15):1799–1808, doi:10.1049/iet-com.2010.0036.
11. Ee CT, Ratnasamy S, Shenker S. Practical data-centric storage. *NSDI*, 2006.
12. Ratnasamy S, Karp B, Shenker S, Estrin D, Govindan R, Yin L, Yu F. Data-centric storage in sensornets with ght, a geographic hash table. *Mobile networks and applications* 2003; **8**(4):427–442.
13. Maia G, Guidoni DL, Viana AC, Aquino AL, Mini RA, Loureiro AA. A distributed data storage protocol for heterogeneous wireless sensor networks with mobile sinks. *Ad Hoc Networks* 2013; **11**(5):1588 – 1602, doi: http://dx.doi.org/10.1016/j.adhoc.2013.01.004.
14. Newsome J, Song D. Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information. *Proceedings of the 1st international conference on Embedded networked sensor systems*, ACM, 2003; 76–88.
15. Li X, Kim YJ, Govindan R, Hong W. Multi-dimensional range queries in sensor networks. *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, ACM: New York, NY, USA, 2003; 63–75, doi:10.1145/958491.958500.
16. Shen H, Zhao L, Li Z. A distributed spatial-temporal similarity data storage scheme in wireless sensor networks. *Mobile Computing, IEEE Transactions on* July 2011; **10**(7):982–996, doi:10.1109/TMC.2010.214.
17. Haghani P, Michel S, Aberer K. Distributed similarity search in high dimensions using locality sensitive hashing. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, ACM: New York, NY, USA, 2009; 744–755, doi:10.1145/1516360.1516446.
18. Albano M, Chessa S, Nidito F, Pelagatti S. Dealing with nonuniformity in data centric storage for wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on* 2011; **22**(8):1398–1406.
19. Gionis A, Indyk P, Motwani R, *et al.*. Similarity search in high dimensions via hashing. *VLDB*, vol. 99, 1999; 518–529.
20. Heinzelman WB, Chandrakasan AP, Balakrishnan H. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on* 2002; **1**(4):660–670.
21. Von Neumann J. Various techniques used in connection with random digits. *Applied Math Series* 1951; **12**(36-38):1.
22. Chen T, Guo D, He Y, Chen H, Liu X, Luo X. A bloom filters based dissemination protocol in wireless sensor networks. *Ad Hoc Networks* 2013; **11**(4):1359–1371.
23. Yu J, Qi Y, Wang G, Gu X. A cluster-based routing protocol for wireless sensor networks with nonuniform node distribution. *AEU-International Journal of Electronics and Communications* 2012; **66**(1):54–61.
24. Wu X, Chen G, Das S. Avoiding energy holes in wireless sensor networks with nonuniform node distribution. *Parallel and Distributed Systems, IEEE Transactions on* May 2008; **19**(5):710–720, doi:10.1109/TPDS.2007.70770.
25. Gamwarige S, Kulasekere E. An algorithm for energy driven cluster head rotation in a distributed wireless sensor network. *Proceedings of the International Conference on Information and Automation*, 2005; 354–359.