A Novel Redundant Data Storage Algorithm Based on Minimum Spanning Tree and Quasi-randomized Matrix

Jun Wang[1,2], Qiong Yi[1,2], Yunfei Chen[3], and Yue Wang[1,2]
[1] Department of Communication and Information Engineering
Nanjing University of Posts and Telecommunications, Nanjing, 210003, China
[e-mail: wang_jun@njupt.edu.cn]
[2] Key Lab of Broadband Wireless Communication and Sensor Network Technology (NUPT), Ministry of Education, Nanjing, 210003;
[e-mail: q12010202@njupt.edu.cn]
[3] School of Engineering, University of Warwick, Coventry, U.K. CV4 7AL
*Corresponding author: Jun Wang

Received October 31, 2016; revised January 5, 2017; accepted August 9, 2017; published January 31, 2018

Abstract

For intermittently connected wireless sensor networks deployed in hash environments, sensor nodes may fail due to internal or external reasons at any time. In the process of data collection and recovery, we need to speed up as much as possible so that all the sensory data can be restored by accessing as few survivors as possible. In this paper a novel redundant data storage algorithm based on minimum spanning tree and quasi-randomized matrix—QRNCDS is proposed. QRNCDS disseminates k source data packets to n sensor nodes in the network ($n>k$) according to the minimum spanning tree traversal mechanism. Every node stores only one encoded data packet in its storage which is the XOR result of the received source data packets in accordance with the quasi-randomized matrix theory. The algorithm adopts the minimum spanning tree traversal rule to reduce the complexity of the traversal message of the source packets. In order to solve the problem that some source packets cannot be restored if the random matrix is not full column rank, the semi-randomized network coding method is used in QRNCDS. Each source node only needs to store its own source data packet, and the storage nodes choose to receive or not. In the decoding phase, Gaussian Elimination and Belief Propagation are combined to improve the probability and efficiency of data decoding. As a result, part of the source data can be

This work is supported by National Nature Science Foundation of China 61401234, 61271234, PAPD Project of Jiangsu Higher Education Institutions and Jiangsu government scholarship for overseas studies.

https://doi.org/10.3837/tiis.2018.01.011 **ISSN : 1976-7277**

recovered in the case of semi-random matrix without full column rank. The simulation results show that QRNCDS has lower energy consumption, higher data collection efficiency, higher decoding efficiency, smaller data storage redundancy and larger network fault tolerance.

Keywords: intermittently connected wireless sensor networks, data storage algorithm, minimum spanning tree, network coding, quasi-randomized matrix

1. Introduction

Wireless sensor networks [1-2] are data-centric networks, consisting of a large number of sensor nodes which have perception, processing, storage and communication capabilities. With the emergence of a series of new sensory devices such as video cameras, RFID readers and seismometers, a variety of new sensor network applications have been developed. This include underwater or ocean sensor networks [3], underground sensor networks [4], volcanic eruptions monitoring sensor networks [5-6], and et al. The majority of above networks are deployed in challenging environments. In such scenarios, base stations are usually far away from the sensor networks due to environmental constraints [7]. And sensor nodes including sink nodes may fail due to various reasons such as storage/energy depletion, hardware failures, natural destructions and even animal attacks [8]. The failures of sensor nodes will lead to data loss and communication link's breakdown. Therefore, how to design an effective data storage algorithm to preserve sensory data in intermittently connected sensor networks has become a hot research area in recent years.

This paper focuses on the data storage problem of intermittently connected wireless sensor networks [9] (ICWSNs) deployed in hash environments. Such networks are usually unattended, which means that sink nodes may appear periodically to collect sensory data, or fixed sink nodes are vulnerable to natural disasters and artificial attacks. In order to prevent data loss caused by node failures, redundant fault-tolerant measures [10-15] must be taken to store the sensory data throughout the network. For the purpose of realizing efficient storage and fast collection of sensory data, an energy-efficient redundant data storage algorithm QRNCDS based on MST (Minimum Spanning Tree) and quasi-randomized network coding is proposed in this paper. QRNCDS has following characteristics: 1) The traversal message complexity is reduced to $O(n)$ by utilizing MST rule(source nodes only need to be forwarded n-1 times ; 2) Every sensor node encodes all the received source data packets into an encoded data packet according to quasi-randomized matrix theory, which achieves better network fault tolerance n-k; 3) The data decoding efficiency is improved by utilizing joint decoding method of Gaussian

Elimination and Belief Propagation algorithm. Data Collectors just need to visit *k+10* survived nodes to achieve the decoding probability of 100%.

The rest of this paper is organized as follows. Section 2 reviews the related works. In section 3, we introduce the network model of ICWSNs. Section 4 gives the details and theoretical analysis of the QRNCDS algorithm. The extensive simulations are conducted in section 5. Section 6 concludes the paper and points out potential future research directions.

2. Related Work

In the challenging monitoring regions, sensor nodes may become more unreliable due to the external environment. In order to ensure the availability of sensory data packets, we can take redundant fault-tolerant approaches to storing them, that is to say, each sensory data generated by each source node is stored on multiple different nodes in a redundant manner. When some nodes fail, data collector can recover the sensory data by visiting other survived nodes. Currently, there are two typical redundant methods: one is based on replicas [10-11] and the other one is based on network coding [12-15].

Redundant data storage methods based on replicas duplicate each sensory data into several replicas which incur extra overhead. Especially when the node's failure probability becomes high, in order to ensure the persistency and availability of sensory data, the network needs to store a huge number of redundant replicas, which is a great challenge for nodes with limited storage space and causes great storage waste. In addition, in the data collection process, with the increase in access nodes, the rate of collecting new source data decreases significantly, which is unfavorable to the rapid recovery of data. At the same time, redundant data storage methods based on network coding can overcome this shortcoming by encoding the *k* source data packets into *n* encoded data packets and storing them on every node in the network. In the network-coding based methods, each sensor node can carry multiple source data packets' information by just consuming one storage space, which greatly improves the utilization of nodes' storage space and eases the storage pressure of sensor nodes. The most representative and related algorithms based on network coding are LTCDS-I algorithm [14] and a low visiting cost storage algorithm (LVCDS) [15].

On the basis of LT codes, Aly [14] proposes a new kind of distributed data storage algorithm for WSNs called LTCDS-I. LTCDS-I algorithm is based on a certain degree distribution function (Ideal Soliton distribution or Robust Soliton distribution). Each sensor node stores an encoded data packet by first randomly choosing a degree value *d* from the degree distribution, and then chooses *d* distinct packets from *k* source packets uniformly with probability d / k . The encoded data packet is the bitwise exclusive-or (XOR) of the *d* chosen source packets. LTCDS-I can be easily implemented, but its traversal speed is very slow. In order to transverse the whole network nodes, each source packet must be forwarded over $4.5n \ln n$ times. Besides, its decoding performance is limited to the collecting order of encoded data packets with different degrees. In the early decoding stage, since fewer packets with low degrees are collected, most packets with high degrees cannot be successfully decoded, the decoding efficiency is very low at this time.

And to fully recover all the source packets, collector needs to visit a large number of survived nodes, which will incur high energy cost.

With the purpose of overcoming the drawbacks of LTCDS-I algorithm, Xiao Yilong [15] proposes a new storage algorithm (LVCDS) with low transversal cost. LVCDS preferentially forwards source packets to nodes that have been visited fewer times and have more neighboring nodes according to the directional random walk rule during the traversal process of source packets, which can reduce invalid forwarding of source packets to some extents. And compared with LTCDS-I, LVCDS decreases the communication energy consumption by 55.6%. During the encoding process, LVCDS adopts random coding instead of LT coding, and in the decoding phase, the collector can successfully recover all the source data packets from any k+10 survived encoded data packets, which greatly improves decoding efficiency and reduces decoding cost. However, there are two distinct limitations in this algorithm: 1) The successful decoding ratio is entirely dependent on the rank of the random matrix (the coefficient matrix of all the collected encoded packets).Only when the random matrix is full column rank, can user decode data successfully and recover all the source packets at one time. Otherwise, no source packets can be recovered. 2) Although LVCDS halves the number of forwarding of source packets to $2n \ln n$, its traversal message complexity still remains $O(n \ln n)$.

In order to develop an efficient data storage scheme, authors in [16] first design a system that integrates the Virtual Broking Coding (VBC) data storage scheme in the IoT realm. Then, they propose an algorithm called Dynamic Adaptive Virtual Broking Coding (DA-VBC) that adapts dynamically the packet redundancy level adopted in VBC to the optimal redundancy level, regarding the actual condition of the network, in order to ensure a reliable data storage and data retrieval. But they don't take the harsh environments into account. Compared to [16], we consider that in the harsh environment, efficient storage and rapid collection are two key issues. Based on the combination of the minimum spanning tree and the semi-random matrix, a new energy efficient and fast decoding algorithm, QRNCDS, is proposed to simplify the coding process and improve the recovery rate of the source packet.

In-network storage is an effective technique for avoiding network congestion and reducing power consumption in continuous data collection in wireless sensor networks. Preference [17] presents an efficient approach to update data at storage nodes to maintain data consistency at the storage nodes without decoding out the old data and re-encoding with new data. They studied a transmission strategy that identifies a set of storage nodes for each source sensor that minimizes the transmission cost and achieves ubiquitous access by transmitting sparsely using the sparse matrix theory. In [17] they assume that all sensors know their locations in the sensor field. But in our paper, each sensor node transmits its geographic position information to the central coordinator. After receiving all the nodes' geographic position information, the central coordinator calculates the minimum spanning tree of the network and sends the related result back to all sensor nodes.

According to the above discussion, it can be seen that the redundant data storage strategy is more applicable than non-redundant data storage strategy. However, the redundant data storage strategy based on replicas is extremely wasteful for the storage of the sensor nodes,

especially in harsh environments. Coding-based redundancy storage strategy overcomes the shortcomings of replicas. It saves storage space and greatly improves the persistence of the data. But to further enhance its performance in harsh environments, the following three issues need to be addressed: 1) Most of the existing coding schemes adopt random walk traversal, the efficiency is low. 2) Most of the existing coding schemes need to access $k(1+\varepsilon)(\varepsilon > 0)$ surviving nodes to decode successfully, and the decoding efficiency is low. 3) Simply using the Gaussian Elimination for decoding is not conducive to the restoration of source data. Once the generated matrix of encoded data packets is not full column rank, no one source data can be stored.

To solve the problem of slow traversal speed and the low recovery efficiency of source data packets in the existing coding schemes, we study a redundant quasi-randomized network coding based data storage algorithm-QRNCDS based on the minimum spanning tree. In the process of traverse the whole network node, by applying the minimum spanning tree rules, QRNCDS can reduce the forwarding number of each source packet to *n-1* times. In the coding process, each sensor node will receive data from different source with certain probability according to the quasi-random matrix. So some nodes in the network only store a single source data packet which can facilitate the decoding operation. In order to avoid the situation where the node does not store any data, QRNCDS also sets a cumulative counter for each sensor node, which ensuring the effective storage of each node, as well as improving the probability of data survivability and the efficiency of recovery.

3. Network Model

The intermittently connected wireless sensor network studied in this paper can be represented as an undirected connected graph $G(V,E)$, where $V=\{1,2,3,\cdots,n\}$ is *n* uniformly deployed sensor nodes and *E* is the set of *m* edges. The system model is shown in **Fig.1**.There are three kinds of nodes in the system: source nodes, storage nodes and the sink node. There are *k* source nodes in the area, denoted as $V_s=\{1,2,3,\cdots,k\}$, which generate sensory data (we assume that each data packet generated by each source node has the same unit size).And *n-k* storage nodes, who are responsible for storing sensory data for source nodes. The connections between the sink node and common sensor nodes are intermittent. In this paper, we assume that all the sensor nodes have the same transmission radius *R*. In order to ensure the network's connectivity, *R* must satisfy formula (1) [15]:

$$R \geq (b \ln n / \pi n)^{1/2} \tag{1}$$

Where *b* is a constant and is greater than 1.

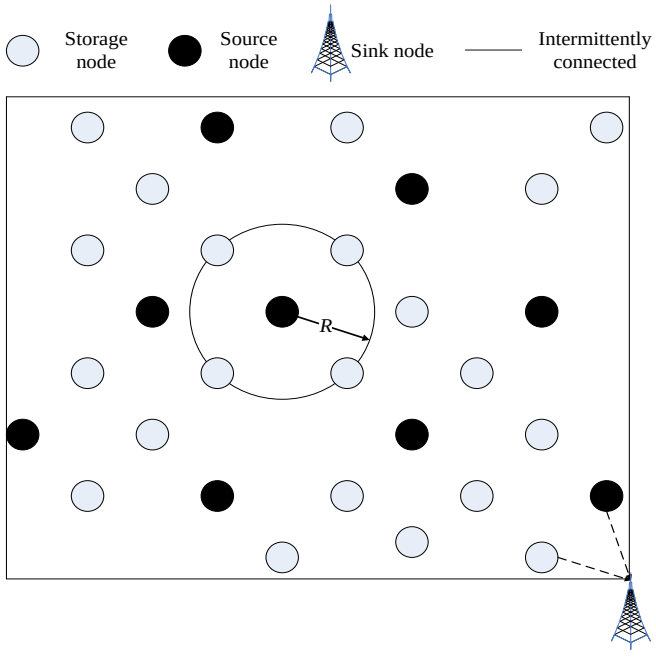


Fig. 1. System model of intermittently connected wireless sensor network

4. Design of QRNCDS Algorithm and Theoretical Performance Analysis

To solve the slow traversal problem and improve the low decoding efficiency of existing algorithms, we design a novel distributed data storage algorithm based on minimum spanning tree rule and a quasi-randomized matrix.

4.1 Quasi-randomized Matrix

The quasi-randomized matrix introduced in this paper is similar to the random matrix defined in literature [15] and its specific definition is as follows:

For a matrix $R_{n\times k}=(r_{ij}), i=1,2,\cdots,n; j=1,2,\cdots,k$, if the matrix has k rows meeting the conditions that in each row only one element's value is"1", the remaining elements' value are all "0", and the positions of the element "1" are all different from each other. And the remaining $n-k$ rows in $R_{n\times k}$ meet the conditions that all the elements are independent of each other and following the 0-1 distribution defined as formula (2), then $R_{n\times k}$ can be called a quasi-randomized matrix.

$$P\{r_{ij}=e\}=\begin{cases} a\ln k/k & for \quad e=1 \\ 1-a\ln k/k & for \quad e=0 \end{cases} \tag{2}$$

Where a is a constant and makes $a\ln k/k \le 1/2$ .According the definition of the quasi-randomized matrix, $R_{n\times k}$ is full column rank. In order to determine the proper value of coefficient *a*, we can get the full column rank probability of sub-matrix $R'_{(k+\varepsilon)\times k}$ (randomly selected) of $R_{n\times k}$ by setting up MATLAB simulation environment. We change the parameter *a* and $\varepsilon(k+\varepsilon \le n)$ to observe the change of the full column rank probability. The experimental results are shown in **Table 1**, where $n=100$, $k=30$. As it can be seen from the table, with the increase of the parameter *a* and ε, the full column rank probability of $R'_{(k+\varepsilon)\times k}$ becomes higher. When $a \ge 2$, the full column rank probability of $R'_{(k+\varepsilon)\times k}$ reaches more than 94%. In this paper, in order to ensure the full column rank of quasi-randomized matrix with higher probability and at the same time make the matrix more sparse, we set *a* as 2.5.

Table 1. FULL COLUMN RANK of $R'_{(k+\varepsilon)\times k}$ with RESPECT to the PARAMETER *a* and ε

	$\varepsilon=4$	$\varepsilon=8$	$\varepsilon=12$	$\varepsilon=16$	$\varepsilon=20$
a=1	0.208	0.425	0.624	0.795	0.868
a=1.5	0.611	0.767	0.840	0.906	0.944
a=2	0.941	0.961	0.985	0.992	0.997
a=2.5	0.993	0.999	1	1	1
a=3	0.995	1	1	1	1

4.2 The Fundamentals of QRNCDS Algorithm

In QRNCDS, source packets are traversed from parent nodes to child nodes layer by layer according to the established minimum spanning tree rooted from source nodes and each node encodes the received packets with given probability according to the quasi-randomized matrix theory. The whole algorithm process can be divided into the following four phases:

1) Initialization phase. Each sensor node transmits its geographic position information to the central coordinator. After receiving all the nodes' geographic position information, the central coordinator calculates the minimum spanning tree of the network and sends the related result back to all sensor nodes. After receiving the information from the central coordinator, each sensor node $u(1 \le u \le n)$ records its neighboring nodes in the

MST(Minimum Spanning Tree) into list $nl(u)$, initializes the encoded packet as 0 ($Y_u = 0(1 \le u \le n)$) and empties the received source packets list $idl(u)$. Each source node $v(1 \le v \le k)$ encapsulates the sensory data into a source data packet $packet_v = (ID_v, X_v)$, where ID_v represents the ID number of the source node and X_v represents the sensory data. To make full use of storage nodes' available storage space, each storage node $i(k+1 \le i \le n)$adds a counter $scount(i)$ to record the number of different source packets that have visited it and sets the counter's initial value to zero($scount(i) = 0$). The MST must be calculated again while the network topology is changed by the intermittently connected environments.

2) Network coding phase. Source data packets are all forwarded layer by layer based on the MST path. At the starting point, each source node $v(1 \le v \le k)$ updates its stored encoded packet as $Y_v = Y_v \oplus X_v$, inserts ID_v into the received packet list $idl(v)$, and forwards $packet_v$ to its neighboring nodes in turn according to the neighboring node list $nl(v)$.

For both source nodes and storage nodes in the network, when receiving $packet_v(1 \le v \le k)$ from its neighboring node $w(1 \le w \le n)$, node $u(1 \le u \le n)$ will first determine whether it is a source node or not. If u is a source node, it inserts $packet_v$ into its forwarding queue directly.

If node u is a storage node, it will first increase its counter value by 1, and then judge whether $scount(u)$ is equal to k or not, if $scount(u) = k$ and the received source packets' ID number list $idl(u)$ is empty, then node u will receive $packet_v$ with the probability of 100% , update Y_u as $Y_u \oplus X_v$, and insert ID_v into $idl(v)$; otherwise, node u will receive $packet_v$ with the probability of $a \ln k / k$. And after that, node u puts $packet_v$ into the forwarding queue.

Before forwarding $packet_v$, node u needs to check its neighboring node list $nl(u)$, if $nl(u)$ has only one node w and w is the previous-hop node of $packet_v$, node u will discard this packet; otherwise, node u will forward $packet_v$ to all the neighboring nodes except the node w according to the neighboring node list $nl(u)$.

The flowchart of network coding phase is shown in **Fig. 2**.

3) Data storage phase. When all network nodes' forwarding queues are empty, which means source packets have visited all the leaf nodes in the tree, the network coding process is finished. At this time, k sensory data have been redundantly stored on the whole network nodes.

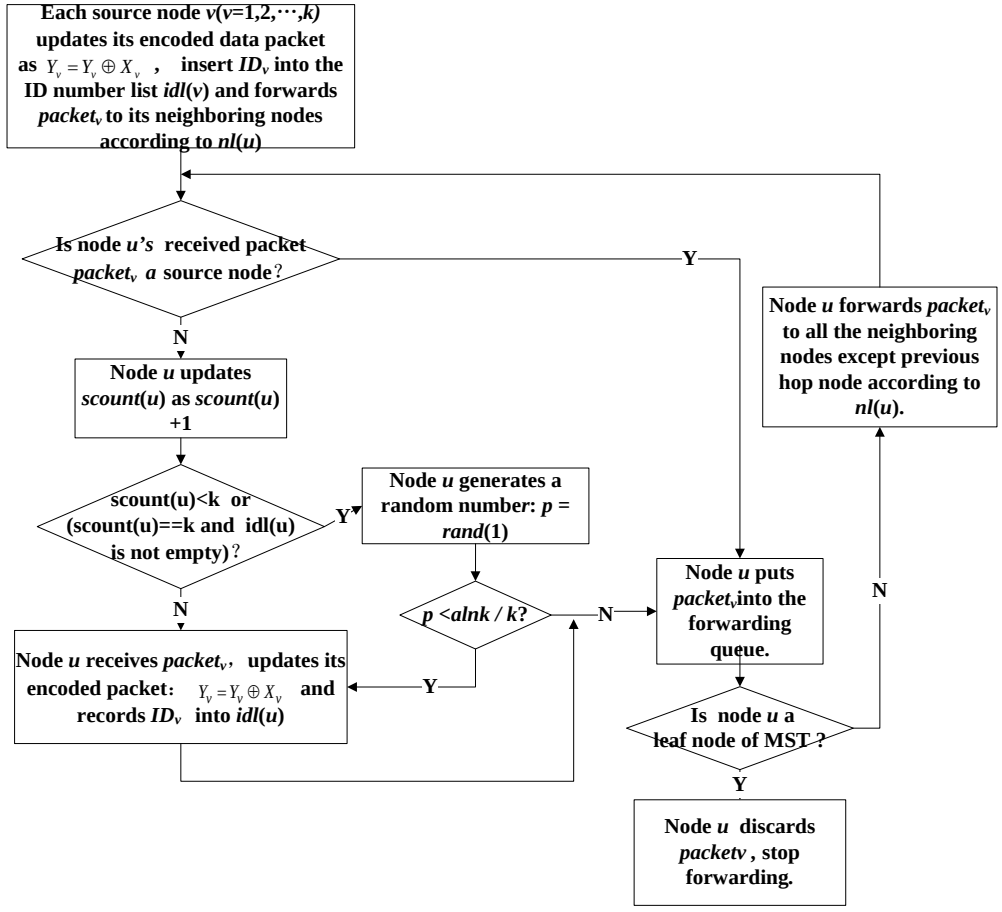


Fig. 2. Flowchart of network coding phase

4) Data decoding phase. At this stage, collector randomly collects $k+\eta$ survived encoded packets and the corresponding received source packets' ID number lists. We denote these $k+\eta$ encoded packets as $Y_i, i=1,2,\cdots,k+\eta$ respectively. The column vector consisting of these $k+\eta$ encoded packets is denoted as $Y=\left[Y_1,Y_2,\cdots,Y_{k+\eta}\right]^T$. Since each encoded packet is the linear combination of k source data packets, we can obtain the coefficient generated matrix $G_{(k+\eta)\times k}$ according to $idl(i), i=1,2,\cdots,k+\eta$. On the basis of $G_{(k+\eta)\times k}$, we can obtain the following linear equations(3):

$$\begin{cases} G(1,1)X_1 \oplus G(1,2)X_2 \oplus \cdots \oplus G(1,k)X_k = Y_1 \\ G(2,1)X_1 \oplus G(2,2)X_2 \oplus \cdots \oplus G(2,k)X_k = Y_2 \\ \vdots \\ G(k+\eta,1)X_1 \oplus G(k+\eta,2)X_2 \oplus \cdots \oplus G(k+\eta,k)X_k = Y_{k+\eta} \end{cases} \Rightarrow G_{(k+\eta)\times k} \cdot \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{bmatrix} = Y \qquad (3)$$

According to the non-homogeneous linear equations' theory, only when $R(G_{(k+\eta)\times k}) = R(G_{(k+\eta)\times k}, Y) = k$ can formula (3) have a unique solution. The detailed decoding steps are as follows:

a) Determine whether the $k+\eta$ collected packets contain *k* source data packets or not, if they already have these *k* source packets, there is no need to decode; otherwise, continue to step b);

b) If the coefficient generated matrix $G_{(k+\eta)\times k}$ is full column rank, and the rank of $(G_{(k+\eta)\times k}, Y)$ is *k*, Gaussian Elimination method is used to decode data; otherwise, skip to step c);

c) Belief Propagation algorithm is used to decode data. First, we perform progressive scanning for the matrix to find the row that contains only one element of 1 and record its position in the matrix as (*i*,*j*), where $1 \le i \le k+\eta, 1 \le j \le k$. Then, we use the *i-th* encoded data packet to recover the *j-th* source data. After this, we go back to $G_{(k+\eta)\times k}$, scan all the elements of the *j-th* column of $G_{(k+\eta)\times k}$, set all elements of 1 to 0 and perform an XOR operation between the corresponding encoded data packet and the recovered *j-th* source data. We repeat these steps until we cannot find any rows that contain only one element of 1.

The pseudo-code of the four phases of the QRNCDS is shown in **Table 2**.

Table 2. PSEUDO-CODE of the four phases of the QRNCDS

1	**/* Initialization stage */**
2	Each sensor node transmits its geographic position information to the central coordinator. After receiving all the nodes' geographic position information, the central coordinator calculates the network's minimum spanning tree and sends the results back to sensor nodes. Each sensor node $u\ (1 \le u \le n)$ records its neighboring nodes in the minimum spanning tree into the list $nl(u)$;The MST must be calculated again while the network topology is changed by the intermittently connected environments.
3	**for** each source node $v = 1\ to\ k$
4	Encapsulate sensory data X_v into the source data packet $packet_v = (ID_v, X_v)$;

```
5       end for
6       for each sensor node u = 1 to n
7         Initialize its encoded data packet as Y_u = 0;
8       End for
9       for each storage node i = k + 1 to n
10      Reset cumulative counter: scount(i) = 0;
11      end for
12      for the network topology is changed
13        Calculate MST again;
14      end for
15      /* Network coding and data storage stage */
16      for each source node v = 1 to k
17        Update its encoded data packet as Y_v = Y_v ⊕ X_v and record ID_v into the ID number list
idl(v);
18        Forward packet_v to all the neighboring nodes in turn according to nl(v);
19      end for
20      for each sensor node u = 1 to n
21        for each packet_v, v = 1 to k that arrives at node u
22          if u is storage node( k + 1 ≤ u ≤ n ), then
23      Update cumulative counter: scount(u) = scount(u) + 1;
24            if scount(u) < k or ( scount(u) == k and idl(u) ≠ φ ) , then
25       p = rand(1);
26              if p < a ln k / k , then
27        Update its encoded data packet as Y_u = Y_u ⊕ X_v and insert ID_v into idl(u);
28              end if
29            else
30        Update its encoded data packet as Y_u = Y_u ⊕ X_v and insert ID_v into idl(u);
31      end if
32       end if
33          Put packet_v into the forwarding queue and check nl(u) before forwarding packet_v:
```

if $nl(u)$ has only one node *w* and *w* is the previous-node of $packet_v$, **then**
$packet_v$ arrives at a leaf node, node *u* discards $packet_v$;
else
Forward $packet_v$ to all the neighboring nodes except the previous node of $packet_v$ in turn according to $nl(u)$;
end if
end for
end for
/* Data decoding stage */
Collector randomly visits $k+\eta$ survived nodes and obtains $k+\eta$ encoded data packets along with the corresponding ID number lists;
If the $k+\eta$ have visited nodes contains *k* source nodes, **then**
Obtain *k* source data directly;
else
if the generated matrix $G_{(k+\eta)\times k}$ of the $k+\eta$ encoded data packets is full column rank, and the rank of $(G_{(k+\eta)\times k}, Y)$ is k then
Utilize Gaussian Elimination method to decode;
else
Utilize Belief Propagation algorithm to decode;
end if
end if

4.3 Theoretical Analysis of QRNCDS Algorithm

1) Applicability

Combining the minimum spanning tree traversal rule with quasi-randomized network coding mechanism, In QRNCDS algorithm, source packets are distributed stored on all network nodes with minimum energy consumption. When a portion of nodes (including source nodes and storage nodes) fail, collector can still collect encoded packets from unfailed nodes and use joint decoding method to recover source data. That's to say, QRNCDS can effectively ensure the persistent of sensory data. But considering that in the initialization phase of QRNCDS, each sensor node needs to communicate with the central coordinator, which incurs extra communication overhead and such overhead increases as the network size becomes larger. Therefore, QRNCDS algorithm is not suitable for

large-scale WSNs.

2) Message Complexity

In the initialization stage of QRNCDS, assuming that the average number of hops between each sensor node and the central coordinator is $\overline{h}$, the total number of transmissions of location information and minimum spanning tree information is $2n\overline{h}$. In the network coding stage, *k* source data packets are transmitted $(n-1)k$ times. Therefore, the total message complexity of QRNCDS is $O\left(2n\overline{h}+(n-1)k\right)$.

3) Storage Cost

Storage cost of each sensor node is consisted of three parts:

a) Neighboring node list storage cost: each node has to store the neighboring nodes' information of the minimum spanning tree. Assuming that the average number of neighbor is $\overline{m}$, the cost of this storage cost is $O(n*\overline{m})$.

b) Encoded packet storage cost: Since every node can at most store one encoded data with the same size of the source data, the encode packet storage cost is $O(n)$.

c) Received source packets' ID number list storage cost: Since every node can at most receive *k* source packets, the cost of the ID number list is $O(k*n)$.

4）Efficiency of Storage Usage

In this paper, we define the efficiency of storage Usage as the average number of source packets that each encoded packet contains. In QRNCDS algorithm, since each storage node have a cumulative counter, when the counter value reaches *k* and ID number list is empty, the node will receive the last arrived source packet with the probability of 100%, so the efficiency of storage usages is greater than or equal to 1, that is to say, there doesn't exist empty storage problem.

5) Computation Complexity

Given that the central coordinator and data collector have powerful computation capability, the computation complexity in both initialization phase and decoding phase will not be considered in this paper. In the network coding stage, each storage node receives source packets with probability $a\ln k/k$, so the average number of source packets that each storage node receives is about $k\cdot a\ln k/k=a\ln k$ .In other words, each storage node conducts "XOR" operation $a\ln k$ times. The computation complexity of QRNCDS is about $O(a(n-k)\ln k)$.

6)Maximum network fault tolerance

The maximum network fault tolerance is defined as the maximum number of nodes that are allowed to fail simultaneously under the premise of recovering all of the *k* source data. Since QRNCDS adopts quasi-randomized network coding mechanism, all the source nodes just store their own sensory data, when *k* source nodes survive and the other *n-k* storage nodes fail, *k* source data can still be recovered. And when a portion of source nodes survive,

collector has to visit at least *k* or more survived nodes to recover original *k* source data. Therefore, the maximum network fault tolerance of QRNCDS is *n-k.*

7）Decoding Efficiency

From the **Table 1**, we can see that when *a=2.5* and $\varepsilon = 8$, full column rank probability of $R'_{(k+\varepsilon)\times k}$ is 99.9%. So the collector can successfully recover almost all the source data packets from any *k*+10 survived encoded data packets, which greatly improves decoding efficiency and reduces decoding cost.

5. Performance Evaluation

To verify the performance of the proposed algorithm, we conduct simulation experiments in the MATLAB environment and compare QRNCDS with three other algorithms: LTCDS-I [14], LVCDS [15] and M-RLC [17]. The simulation settings are as follows: The network size is $100*100$, *n* sensor nodes are uniformly deployed in the square area, and the proportion of the source nodes is 30%. Each sensor node's transmission radius is set to 14 (to ensure the connectivity of the network) and the coefficient *a* of receiving probability is set to 2.5.We set the total number of transmission of source packet as $ft = cn\ln n(c \geq 1)$ (According to reference [14], only when random walk steps reach $O(n\ln n)$, can network coverage reaches 100%). The main performance metrics we investigate are: network coverage performance and decoding performance.

5.1 The Traversal Performance of Source Packets

1) The network coverage

In this paper, we use the definition of network coverage of reference [15], if there is a two-dimensional random graph *T(n, r)*, the network coverage is defined as: the ratio of the number of different vertices visited by the random walk to the total number of vertices in the graph.

In this simulation scenario, the number of sensor nodes is set to 300. **Fig. 3** depicts the network coverage curves of QRNCDS, LTCDS-I and LVCDS respectively, where the horizontal axis represents the coefficient *c* of ft (the number of transmission of source packets), and the vertical axis represents the average network coverage of each source packet. From the **Fig. 3**, it can be observed that QRNCDS achieves the best network coverage performance. Since QRNCDS follows minimum spanning tree rule, no matter how much the coefficient *c* is, *k* source packets can visit all sensor nodes and achieve network coverage with 100%. LVCDS's coverage performance comes in second as expected, since LVCDS adopts directed random walk rule, which greatly reduces the number of invalid forwarding. When *c* is equal to 1, the network coverage can reach 98%, and when *c* is greater than 2, *k* source packets can visit every node. Obviously the network coverage performance of LTCDS-I algorithm based on simple random walk is the worst, only when the number of transmission is greater than *4.5nlnn*(that's to say, c>4.5), can all source packets visit every node.

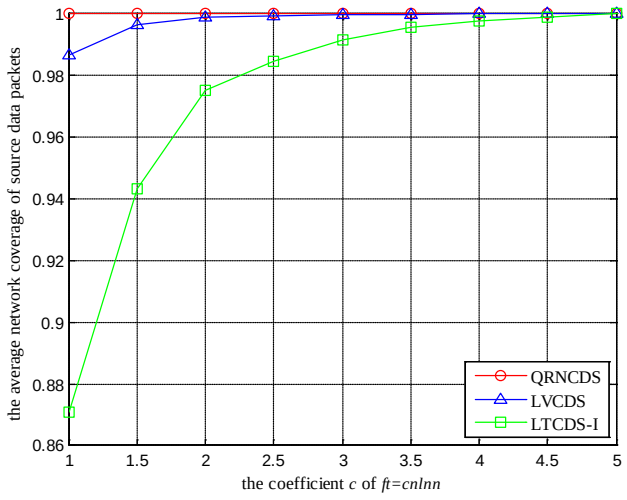


Fig. 3. The experimental result of source packets' network coverage

2) The average number of transmission

We investigate the average number of transmission of source packets required by QRNCDS, LTCDS-I and LVCDS to achieve full network coverage with various network density. **Table 3** shows the results. We observe that with the network density becoming denser, the average number of forwarding keeps increasing in order to ensure that all the network nodes can be successfully traversed by source packets. In QRNCDS, each source packet takes only *n*-1 hops to cover the whole network, the value is about 9.41%~10.59% of LVCDS algorithm and 4.11%~5.17% of LTCDS-I algorithm. Thus it can be seen that adopting QRNCDS algorithm can reduce communication energy greatly. Compared with LVCDS, it reduces energy consumption by an average of 89.73% and compared with LTCDS-I, it reduces energy consumption by an average of 95.08%.

Table 3. THE EXPERIMENTAL RESULTS of AVERAGE NUMBER OF FORWARDING

	QRNCDS(*ft*1)	LVCDS(*ft*2)	LTCDS-I(*ft*3)	*ft*1/*ft*2(%)	*ft*1/*ft*3(%)
n=300,*k*=90	299	3176	7283	9.41%	4.11%
n=400,*k*=120	399	3768	8584	10.59%	4.65%
n=500,*k*=150	499	4748	9920	10.51%	5.03%

n=600,*k*=180	599	5766	11857	10.39%	5.05%
n=700,*k*=210	699	6770	13775	10.32%	5.07%
n=800,*k*=240	799	7910	15623	10.10%	5.11%
n=900,*k*=270	899	8766	17483	10.26%	5.14%
n=1000,*k*=300	999	9442	19307	10.58%	5.17%

5.2 Data Recovery Performance

1)Decoding performance

In this simulation scenario, the total number of sensor nodes in the network is set to 900 and the node failure probability p_f is set as 0.3 and 0.8 respectively. **Fig. 4** and **Fig. 5** show the average recovery ratio of source data packets. From the two figures, we can see that when the node failure probability is small ($p_f = 0.3$, there are adequate survived nodes), the three algorithms can achieve the goal of successful decoding by collecting a certain number of survived packets, where LTCDS-I needs to visit about 390 survived nodes to make the source data recovery ratio reach 1, while QRNCDS and LVCDS only need to visit 280 survived nodes to recover all the original 270 source data, their recovery costs are approximately reduced by 28.21%. When the node failure probability is comparatively high ($p_f = 0.8$ which means very few nodes survive in the network), all the algorithms' successful decoding ratio cannot reach 1. The decoding ratio of LVCDS keeps 0, this can be explained by the fact that LVCDS utilizes Gaussian Elimination method to decode, when it collects less than 270 survived packets, the coefficient matrix cannot be full column rank, so it cannot recover any source packet. For QRNCDS and LTCDS-I, the Belief Propagation algorithm is used during the decoding process, some portion of source data still can be recovered. Therefore, compared with the other two algorithms, QRNCDS is more suitable for working in hash environments. No matter how much the node failure probability is, QRNCDS can always ensure higher decoding ratio.

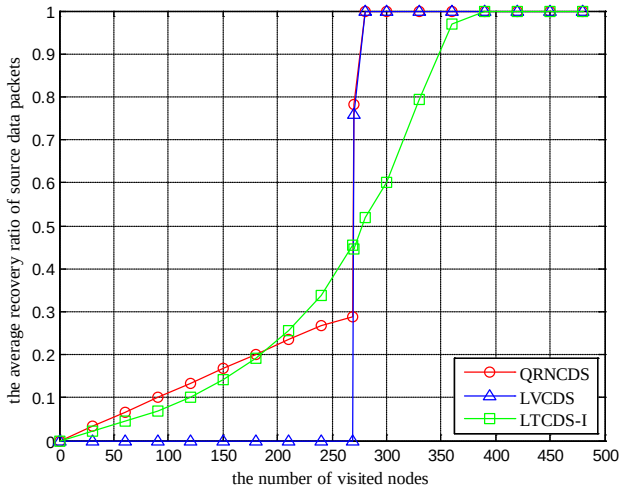


Fig. 4. Decoding performance with $p_f = 0.3$

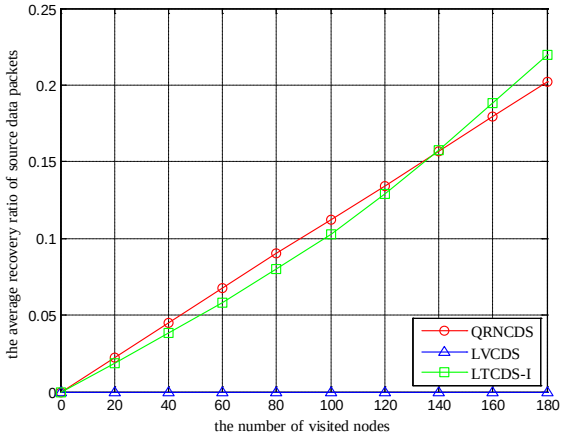


Fig. 5. Decoding performance with $p_f = 0.8$

Next, we conduct experiments to compare the performance of M-RLC proposed in Preference [17] with QRNCDS in our paper.

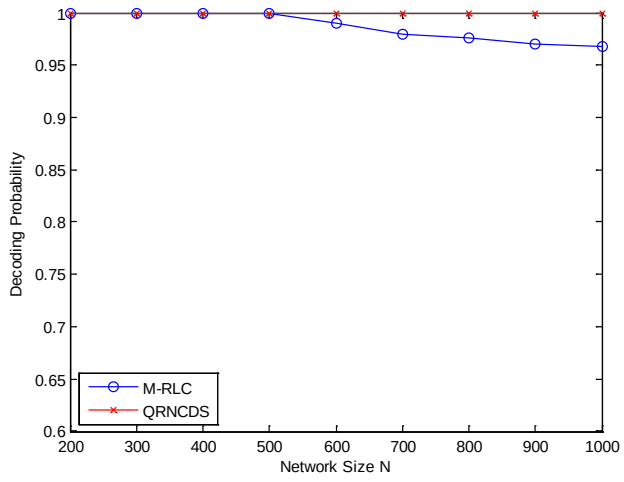


Fig. 6. Decoding probability versus network size

Fig. 6 shows the decoding probability of the data collector as a function of the network size. We can see that our approach almost achieve 100% decoding probability with increasing network size, which is greater than M-RLC [17]. The reason is that QRNCDS adopts the joint decoding algorithm of Gaussian elimination and belief propagation algorithm to improve the probability and efficiency of data decoding.

2) Average Network Fault Tolerance

We test the average number of nodes that are allowed to fail simultaneously when decoding ratio can reach 1. The numerical results are shown in **Table 4**. We can find that QRNCDS and LVCDS have the same average network fault tolerance, its value is approximately *n*-*k*-10, which is more than twice the value achieved by LTCDS-I. With the increase of the number of sensor nodes, the difference among these three algorithms is gradually narrowing, but QRNCDS and LVCDS still outperform LTCDS-I in terms of fault tolerance.

Table 4. THE AVERAGE NETWORK FAULT TOLERANCE of Each ALGORITHM

	QRNCDS(*nft*1)	LVCDS(*nft*2)	LTCDS-I(*nft*3)	*nft*1/*nft*2	*nft*1/*nft*3
n=300,*k*=90	200	200	120	1	1.6667
n=400,*k*=120	270	270	170	1	1.5882
n=500,*k*=150	340	340	220	1	1.5455
n=600,*k*=180	410	410	270	1	1.5185
n=700,*k*=210	480	480	330	1	1.4545

n=800,k=240	550	550	410	1	1.3415
n=900,k=270	620	620	510	1	1.2157

6. Conclusion and Future Work

In this paper, we first study the data storage problem of intermittently connected wireless sensor networks. Then, we analyze the advantages and disadvantages of replicas storage strategy and coding strategy. Considering that existing storage solutions based on network coding have problems of slow traversal and low decoding efficiency, we optimize LT code-based and randomized redundant storage algorithm and design an energy-efficient distributed data storage algorithm QRNCDS which is based on minimum spanning tree and quasi-randomized matrix. Compared with other algorithms, QRNCDS realizes full network coverage of source data packets in linear time, which greatly saves communication costs, and simplifies data encoding and storage process by use of quasi-random network coding and joint decoding method (Source nodes don't need to calculate degree and only stores their own sensory data). QRNCDS can also improve the decoding efficiency of source data packets. The experimental results show that QRNCDS can greatly improve the recovery efficiency of source packets in the case of rapid traversal of the whole network. It is also strongly applicable to wireless sensor networks deployed in harsh environments. However, there are also some shortcomings in this algorithm. QRNCDS is based on high-powered central coordinator to calculate the network's minimum spanning tree and every sensor node needs to know its location information in advance. In the future research, we may consider how to utilize cooperative communication mechanism among neighboring nodes to realize reliable and efficient data storage. Recently more and more machine learning algorithms such as SVM have been applied to the research of wireless sensor networks, but there are some limitations in traditional SVM algorithm, we are very happy to see there are a lot of innovations to improve traditional SVM and machine learning approaches [18-21]. Machine learning approach can be applied to our work in our future research.

References

[1] Ren Fengyuan, Huang Hai and Lin Chuang, "Wireless sensor networks(In Chinese)," *Journal of Software*, vol. 14, no. 7, pp. 1282-1291, 2003. Article (CrossRef Link).

[2] Akyildiz I F, Su W, Sankarasubramaniam Y, et al, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393-422, 2002. Article (CrossRef Link).

[3] Liu Kunpeng, Jiang Weidong, "The research of underwater sensor node coverage model based on perception factor (In Chinese)," *Journal of Nanjing University (natural sciences)*, vol. 51, no. 6, pp. 1203-1209, 2015. Article (CrossRef Link).

[4] Huang Renjie, Song Wen-Zhan, Xu Mingsen, et al, "Real-world sensor network for long-term volcano monitoring: design and findings," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 2, pp. 321-329, 2012. Article (CrossRef Link).

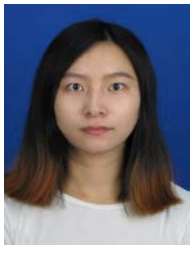
[5] Tan Xin, Sun Zhi, Akyildiz I F, "Wireless underground sensor networks: MI-based communication systems for underground applications," *IEEE Antennas and Propagation Magazine*, vol. 57, no. 4, pp. 74-87, 2015. Article (CrossRef Link).
[6] Cobos M, Perez-Solano J J, Felici-Castell S, et al, "Cumulative-sum-based localization of sound events in low-cost wireless acoustic sensor networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1792-1802, 2014. Article (CrossRef Link).
[7] Liu Peng, Zhang Song, Qiu Jian, et al, "A redistribution method to conserve data in isolated energy-harvesting sensor networks," *Computer Science and Information Systems*, vol. 8, no. 4, pp. 1009-1025, 2011. Article (CrossRef Link).
[8] Shen Chao, "A study of distributed fountain codes based data collection techniques in wireless sensor networks (In Chinese)," *Master's diss. Hangzhou:HangzhouDianzi University*, 2011. Article (CrossRef Link).
[9] XueXinyu, Hou Xiang, Bagai Rajiv, "Data preservation in intermittently connected sensor networks with data priority," in *Proc. of 10th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp.122-130, 2013. Article (CrossRef Link).
[10] Shen Yulong, Xi Ning, Pei Qingqi, et al, "Distributed storage schemes for controlling data availability in wireless sensor networks," in *Proc. of 7th International Conference on Computational Intelligence and Security*, pp.545-549, 2011. Article (CrossRef Link).
[11] Tang Bin, JaggiNeeraj, Takahashi Masaaki, "Achieving Data K-Availability in Intermittently Connected Sensor Networks," in *Proc. of 23rd International Conference on Computer Communication and Networks*, pp.1-8, 2014. Article (CrossRef Link).
[12] Lin Yunfeng, Liang Ben, Li baochun, "Data Persistence in large-scale sensor networks with decentralized fountain codes," in *Proc. of 26th IEEE International Conference on Computer Communications*, pp.1658-1666, 2007. Article (CrossRef Link).
[13] Zhang Wei, Zhang Qinchao, Xu Xianghua, Wan Jian, "An optimized degree strategy for persistent sensor network data distribution," in *Proc. of 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pp.130-137, 2012. Article (CrossRef Link).
[14] Aly S A, Kong Z N, Soljanin E, "Fountain codes based distributed storage algorithm for large-scale wireless sensor networks," in *Proc. of International Conference on Information Processing in Sensor Networks*, pp.171-182, 2008. Article (CrossRef Link).
[15] XiaoYilong, "Random data redundancy method and it application in distributed storage systems (In Chinese)," *PhD diss. Chengdu: School of Computer Science and Engineering*, 2013.
[16] Camila H. S. Oliveira, Yacine Ghamri-Doudane, Carlos E. F. Brito, et al, "Optimal network coding-based in-network data storage and data retrieval for IoT/WSNs," in *Proc. of 14th IEEE International Symposium on Network Computing and Applications*, pp.208-215, 2015. Article (CrossRef Link).
[17] Cheng Zhan, Fuyuan Xiao, "Coding-based storage design for continuous data collection in wireless sensor networks," in *Proc. of Journal of Communications and Networks*, pp.493-501, 2016. Article (CrossRef Link).
[18] Bin Gu, Victor S. Sheng, Shuo Li, "Bi-parameter space partition for cost-sensitive SVM," in *Proc. of 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pp.3532-3539, 2015. Article (CrossRef Link).

[19] Bin Gu, Xingming Sun, Victor S. Sheng, "Structural minimax probability machine," in *Proc. of IEEE Transactions on Neural Networks and Learning Systems*, pp.1-11, 2016. Article (CrossRef Link).
[20] Zhihua Xia, Xinhui Wang, Xingming Sun, et al, "Steganalysis of least significant bit matching using multi-order differences," in *Proc. of Security and Communication Networks*, pp.1283-1291, 2014. Article (CrossRef Link).
[21] Zhihua Xia, Xinhui Wang, Xingming Sun, et al, "Steganalysis of LSB matching using differences between nonadjacent pixels," in *Proc. of Multimed Tools Appl*, pp.1947-1962, 2016. Article (CrossRef Link).

Jun Wang received her B.E. and M.E. degrees in electronics engineering from Nanjing Institute of Communication Engineering, P.R.China, in 1998 and 2001, respectively. He received his Ph.D. degree from the Nanjing University, China in 2012. She is currently working as an associate professor of Nanjing University of Posts and Telecommunications. Her research interests include wireless sensor networks, wireless communications, broad band networking, network protocol and network security.

Qiong Yi received her Master's degree from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2016, in Department of Communication and Information Engineering. Her research interests include wireless sensor networks, data storage, network coding.

Yunfei Chen received his B.E. and M.E. degrees in electronics engineering from Shanghai Jiaotong University, Shanghai, P.R.China, in 1998 and 2001, respectively. He received his Ph.D. degree from the University of Alberta in 2006. He is currently working as an Associate Professor at the University of Warwick, U.K. His research interests include wireless communications, cognitive radios, wireless relaying and energy harvesting.



Yue Wang is currently pursuing Master's degree from Nanjing University of Posts and Telecommunications, Nanjing, China, in Department of Communication and Information Engineering. Her research interests include wireless sensor networks, data storage, network coding and cloud computing.