EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Delay and energy-efficient data collection scheme-based matrix filling theory for dynamic traffic IoT

Xuemei Xiang[1], Wei Liu[2], Tian Wang[3], Mande Xie[4*], Xiong Li[5], Houbing Song[6], Anfeng Liu[1] and Guoping Zhang[7]

Abstract

Data collection is the basic functions of the Internet of Things (IoT), in which the sensed data are concentrations from sensor nodes to the sink, with a timely style, so the smart response can be done for emergency. The goal of multi-modal sensor data fusion is to obtain simple and accurate data to enhance system reliability and fault tolerance. Energy efficiency and small delay are the most important indicators which govern the performance of IoT. Convergecast is a low-latency data collection strategy based on effective time division multiple access (TDMA), in which each sensor node generates a packet, and *m* packets can aggregate to a packet. However, in most practical networks, sensor nodes do not necessarily generate packets during each data collection cycle, but instead generate packets from time to time. In the previous convergecast strategy, each node was fixedly allocated a slot, which increased the delay and wasted energy. A delay and energy-efficient data collection (DEEDC) scheme-based matrix filling theory is proposed to collect data in a randomly generated WSNs with minimum delay and energy consumption. The DEEDC scheme uses a clustering approach. For each cluster, the number of slots required for transmission is calculated by matrix filling theory, not the number of nodes that actually generate data. This ensures that data can be collected in a network with randomly generated data (number of slots ≤ number of nodes), thereby avoiding the allocation of slots for each node and the acquisition of redundant data to lead to the wastage of time and energy. Based on the above, a mixed slot scheduling strategy is proposed to construct energy and delay-efficient, collision-free schedule scheme. After extensive theoretical analysis, by using the DEEDC scheme, the delay is reduced by about 50~80%, and the energy consumed is reduced by about 40~57%.

Keywords: Multi-modal, Clustered routing, Matrix filling, Low latency, Energy efficient

1 Introduction

The Internet of Things (IoT) is the latest Internet evolution, with billions of Internet devices. Such as cameras, RFIDs, in addition to wearable, vehicles, to smart meters, medication pills, industrial, and signs machines [1–5]. An important part of the IoT is wireless sensor networks [6, 7]. Widely followed by researchers due to their large-scale, self-organizing, and dynamic characteristics [8–10], and has been widely used in industry, traffic information, military, environmental monitoring, etc. [11–15]. As the technology matures, the processing capabilities and sensing capabilities of sensors become more and more powerful. In addition, the number of types of perception increases and the accuracy of perception is higher. In addition, its size has become smaller and the price is cheaper [16–20]. This has made the application range of wireless sensor networks more widely expanded, including smart home, intelligent agriculture, and other fields [21–25]. With the development, sensing device has also developed rapidly; the current Internet is experiencing a trend from centralization to marginalization, Cloud computing [26, 27], Edge computing [28, 29], and Fog computer [30–32] which correspond to the new computational model proposed for such development [26, 28, 33–35]. With the rapid rise of artificial intelligence technology [36, 37], the combination of artificial intelligence and Internet of Things (IoT) has made it a longer development [38–40], which has become the focus of researchers.

* Correspondence: xiemd@zjgsu.edu.cn
[4]School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China
Full list of author information is available at the end of the article

Broadly speaking, multi-modal sensor data fusion not only refers to the synthesis in different modalities but also includes different features and the same type of data fusion in the same mode.

Sensor nodes are responsible for monitoring events within the perceived range, sensing surrounding data. Then send them to the sink node via multi-hop routing [41–45]. To realize the enormous potential of wireless sensor networks (WSNs), it is important for sink to integrate the sensed data by many sensor nodes, and distill the high value information for each application needs [46–48]. Based on economic considerations, sensor nodes are simple in construction, small in size, and powered by batteries, which can be used with limited energy [3, 5, 7, 43, 49, 50]. Therefore, energy efficiency has become an important indicator [51, 52]. On the other hand, the main application of WSNs is monitoring [22, 49]. When a monitored event occurs, the packet is forwarded to the sink so that the system can quickly process the event, because the delayed data transmission may cause serious loss [21, 44]. For example, in the monitoring of critical facilities, important buildings, industrial sites, geological hazards, and fires, delayed data routing can lead to the destruction of important facilities, and people and objects in geological disaster areas cannot be transferred, resulting in serious losses. Energy-efficient and fast data collection has become an important research content of WSNs [4, 5, 21, 53–55].

Convergecast data collection is an effective method widely used [4, 5, 21, 53–55]. The Convergecast method uses two important methods. The first is the data fusion mechanism. It adopts a data aggregation method in which a class of m data packets can be organized into one after encountering each other [4, 5, 21]. This data aggregation method is often used in applications such as monitoring the highest temperature, average temperature, and minimum temperature [4, 5, 21]. For example, when monitoring crops, only the three temperatures mentioned above need to be known. The second method is the time division multiple access (TDMA) mechanism [4, 5, 21]. In the TDMA method, a slot for performing data operations is accurately assigned to each node. When a node is performing a data operation, it is awake, and at other times, it goes into the sleep state to save energy. Therefore, for Convergecast, the working process of each node can be divided into two stages. First, the node does nothing except receive data. In the second phase, the node performs data transmission, that is to say, the received data is fused into one data packet and sent out, and then the data is no longer received [5].

In many studies, the state transition of a node does not consume energy. Therefore, it is a reasonable way to use the TDMA method to arrange the nodes' slots on demand. However, in practice, the state transition of a node requires energy consumption. Therefore, Xu et al. [56] suggest that the node be in a sleep state for more time and reduce the possibility of node state transitions. If the interval between the two data operations of the node is long, let the node sleep first and then awake, which can save energy. However, if the interval between two data operations of the node is relatively short, it may be more energy-saving to keep the node in the awake state. Xu et al.'s research proposes a scheduling strategy that requires only two state transitions in a data collection process. The first phase is the collection of data in the cluster, at which point each member of the cluster sends data to the head. For cluster member nodes, they only need one slot to send data, and all the data operations are completed. For the cluster head, in the first stage, after awake, it starts to receive the data. In the above process, the cluster head node is always awake. The second phase is the data transfer between the cluster heads: When the first phase is completed, each cluster head has received all data. At this time, inter-cluster data relay starts from the far sink area. To save energy, the cluster head node near the sink area first goes into the sleep state, and waits until the inter-cluster data relay is needed. In the method proposed by Xu et al., the node only needs to convert the state twice.

In the study of Xu et al., a structure of an unequal cluster network is used. In such a network structure, the radius of all clusters in the network is the same. In Li et al. [5], we believe that the performance of data collection (e.g., latency and energy utilization) is not optimal in a network structure with equal cluster radius. Therefore, we use a network structure with unequal cluster radii. Different from most previous studies, in our proposed strategy, the cluster radius is smaller in the far sink region. With such a network structure, the node can only switch state once. This is because the nodes in the far-sink area have a smaller cluster radius, and they enter the inter-cluster relay process earlier. When the node near the sink area completes the data collection work in the cluster, the data packet in the distance has just arrives. Therefore, the number of state transitions can be reduced from twice to once, and the delay of data collection is significantly reduced.

Although the current research has achieved good results, it is based on the assumption that each node generates a packet in each round of data transmission. However, in actual situations, some nodes do not generate packets every time. As a simple example, when monitoring forest fires, nodes only need to send data packets when the danger is detected and only need to send sporadic packets in other times. Obviously, it is not appropriate to assign a slot to each node in such a network. According to the previous strategy, at least n slots are needed in a cluster with n nodes. However, if the data packet is generated irregularly, if the proportion of nodes generating the data packet is only ε, then the theoretical number of slots to be allocated is εn. In this case, the strategy of still allocating slots for all nodes is wasteful, making the data collection delay large, and let node awake, which does not need to transmit data, also consume energy and reduce network lifetime.

In a network where packets are randomly generated, in theory, the slots required can be greatly reduced, but finding a suitable collection strategy is an extremely hard problem. The difficult point is that the TDMA scheduling strategy is based on a centralized scheduling, and the state of each node in the network is obtained before scheduling. In addition, TDMA requires that the network parameters for each data collection be the same, ensuring that the scheduling slots are fixed. The network randomly generated by the data does not meet the above requirements.

First, the nodes that generate the packets and the number of generated packets are all changes. Therefore, it is necessary to re-allocate the slot for each node before each round of scheduling to achieve good results, but this is not practical. The reason is that the scheduling algorithm does not know in advance which nodes generated the data and which did not. In a network in which nodes randomly generate packets, the number of slots required for each round is different, which we cannot know in advance. If you assign a fixed slot to each node, it can cope with all the situations, but the performance is not good. At present, there has not been a research on a network generated randomly for data packets.

On the other hand, many studies have shown that there is a certain correlation between data packets in time and space. When the collected data packets reach a certain amount, some missing data packets can also be recovered, so that the information volume of the network is not lost.

We have proposed a delay and energy-efficient data collection (DEEDC) scheme-based matrix filling theory. By skillfully utilizing the matrix filling theory, the DEEDC scheme allows the entire network to collect only a subset of data samples in each round, while other data can be recovered from the acquired data. Suppose there are n nodes, to collect m packets (according to the matrix filling theory, and $m \leq n$). Thus, in the DEEDC scheme, only m slots need to be arranged for data collection. This reducing the number of time slots is required, and ultimately reducing the delay. In the case where a node randomly generates a data packet, it is impossible to have exactly m cluster member (CM) nodes generating packets each time in a cluster. Our method is to find the threshold k ($k > m$). If more than k packets are generated in the cluster, the cluster head (CH) node will only receive k of them.

This paper proposes a DEEDC scheme-based matrix filling theory to collect data with energy effective and low delay style.

1. A DEEDC scheme-based matrix filling theory is proposed to collect data with energy effective and low delay style for dynamic traffic WSNs. Although the collection strategy proposed in the past can also be applied to networks that randomly generate data packets, it will generate a lot of delay and energy waste. The difference between the DEEDC scheme and the previous data collection strategy lies in the fact that by using the matrix filling theory, it is adapted to ensure that no data is lost in a network that dynamically generates data. Thus, each cluster head allocates a minimum number of collected data packets that can recover all the data.
   As far as we know, the DEEDC scheme is the first strategy that can adapt to dynamic data generation, and the number of slots, delay, and energy consumption required are much smaller.
2. A clustered data routing protocol suitable for networks that generate data from time to time is given for DEEDC scheme. Here, the unequal clustering network structure is used, which is the result of our previous research. Then, for the specific situation in this article, we give the data routing strategy in detail.
   The data routing strategy consists of two processes. The data collection phase in the cluster is the first, which contains two sub processes: the first is to allocate a slot for the cluster member; the second is to send the data to the cluster head. In the first step, the competing MAC protocol is used. Each cluster member with data generation competes with the channel, reporting data generation to the cluster head. After receiving the message of the member node, it is determined which packets to be received by the cluster head node. The ID and the slot information of the node are broadcasted by broadcasting. The second stage is inter-cluster data routing. Through the above two stages, data can be efficiently collected in a wireless sensor network that randomly generates data.
3. After rigorous and comprehensive theoretical analysis, the DEEDC scheme can adapt to wireless sensor networks that dynamically generate data. The DEEDC scheme reduces the node delay by about 50~80%, and reduced the energy consumption by at least 40%, up to 57%.
   The remaining chapters are organized as follows: the research background and related work is located in Section 2. The relevant model is in the third section. In Section 4, the DEEDC scheme proposed in this paper is introduced in detail. In Section 5, the DEEDC scheme was analyzed in detail and compared with the previous scheme. In Section 6, we summarize the full text. Finally, we

give the meaning of the abbreviations appearing in the paper in Chapter 7.

2 Background and related work

The prosperity of WSNs provides an inexhaustible driving force for the development of IoT [57]. In WSNs, the sensor node collects peripheral data and forwards it to the sink node by using other sensor nodes as relays. The sink node receives the data packets generated in the network and by analyzing these data packets, timely controlling the special changes and making adjustment measures to achieve the purpose of automation.

In the above process, two main issues need to be taken seriously. The first is the energy, followed by the delay [10].

Sensor nodes are battery-powered and deployed in special areas with complex environments. It is conceivable that it is very troublesome to replace the battery for them [58]. In previous studies, the energy consumed by sensor nodes mainly considered the following: (1) due to packet transmission, including send and reception; (2) switching state [59]. In order to achieve the goal of reducing energy consumption, sensor nodes typically do not remain awake all the time, but constantly switch between sleep and awake states. Although state switching consumes a portion of the extra energy, it is small compared to the energy consumed for listening. It generally achieves the goal of reducing energy consumption, so it is often used. (3) To keep listening [60]. The node is awake but no packets need to be transmitted. In this case, it listens for the status of other nodes and receives it directly after sensing the packet. The energy consumed to keep listening is greater than remains asleep. Therefore, in the ideal case, the node wakes up just when it needs to transmit packets, and stays asleep at other times. That is, almost all of the energy is used to transmit packets. To this end, relevant scholars have also made a lot of research.

Delay is another factor that requires our attention in addition to energy consumption [61]. Real-time performance is a very important feature for WSNs. For example, in dealing with forest fires, the sooner the sink node receives the danger, the faster the rescue response will be. Timely rescue can keep losses to the smallest possible extent. However, since the WSNs are wirelessly transmitted, which means that the generation of delays is inevitable. The delay of a single hop mainly includes [60] (1) the time from the generation of the packet to the time it was officially sent. In a duty cycle controlled network, the receiver may be in a sleep state. The sender must wait, and the transmission interruption causes a delay [60]. (2) Delay caused by communication. In order to increase the reception rate of packets, the sender and the receiver often need to establish a connection in advance. Especially in the network to which the Send-Wait protocol is added. All of the above operations result in an increase in the delay of the node. (3) Delay in transmitting data packets. Nodes always transmit packets at a certain rate. The more packets, the greater the delay. Increasing the transmission rate of the node reasonably or reducing the amount of data transmitted can achieve the purpose of reducing the delay. (4) Delay in data retransmission [62]. Since the transmission is wireless, if the packet is not successfully received, this means that the sender node needs to transmit the same packet again, causing a delay.

2.1 Existing research related to delay

Delay as a performance indicator that seriously affects WSNs is naturally the focus of researchers [63]. In order to reduce the delay, the methods commonly used by researchers include the following.

1. In a network with duty cycle control, the receiver may fall asleep, causing a part of the delay. Eliminating the duty cycle control obviously reduces the delay directly to zero, but at the expense of increasing the energy consumption. Therefore, researchers tried to control the sender so that when the packet needs to be sent, it can wait for wake-up receivers as soon as possible. Reasonable control mainly includes finding out the hotspot time and increase the probability that the potential receiver will wake up during this period. All possible recipient nodes are scheduled to wake up evenly during the period during which the sender node may send a data packet [64].
2. Increase the transmission radius to make the hop count smaller. However, as the transmission radius increases, the reception rate of the node necessarily decreases; conversely, if the previous reception rate remains unchanged, the transmission power must be increased. Of course, related research indicates that the reception rate can be guaranteed even if the transmission power is not changed, for example, using opportunistic routing. However, this approach sacrifices other performance.
3. The shortest routing scheme [65]. In this scheme, once the node is deployed, it first sets its own hop count to infinity. The aggregation node then starts broadcasting, which contains the message "I am at a distance of 0 from the sink". All nodes that have received this message have a distance of 1 from the sink node. Subsequently, these nodes continue to broadcast the message "My distance to the sink node is". Those nodes that received this message set the distance to $k + 1$. In particular, if node A receives two different broadcast data, and the hop count information contained therein is i and j,

respectively, it is set to $min(i, j)$. Through the above method, the minimum hop count is successfully found, and the data packet transmission is performed along the found path, which reduces the delay.
4. By increasing the reliability of the transmission, which is mainly for networks with low reliability [62]. Assuming that the reliability of the node transmission is not high, so that the retransmission again causes a large amount of delay. From this perspective, improving reliability can effectively reduce the delay. In fact, many research programs are based on such ideas. In order to improve reliability, increasing the transmission power is the most direct and simple method. In addition, opportunistic routing is a common method to increase the reliability of communication links [64]. In opportunistic routing, the sender node selects multiple nodes as relays, which we refer to as candidate nodes. When data transmission is performed, the sender transmits the data packet to all candidate nodes at one time by broadcasting. Subsequently, one of the node to receive the packet is selected in order of priority, and the data packet is continuously transmitted forward through this node. Obviously, a one-hop transmission fails only if all candidate nodes fail to receive.
5. Reduce the delay of routing packets. Improve the packet transfer rate from a hardware perspective. Increasing the transmission rate, the time it takes to transmit an equal amount of data naturally decreases. The method is simple in thought and the effect is obvious. However, for integrated wireless sensor nodes, increasing the transmission rate means replacing all previous nodes with new sensor nodes, and the workload should not be underestimated. In addition, because of the size and energy of the sensor node itself, the transmission rate can be increased to a lesser extent.

Since the packets collected by the sensor nodes have a certain correlation in time and space, after receiving the packets the nodes can directly remove redundant parts through data aggregation technology [66]. When the transmission rate is constant, it can directly reduce the delay.

Since all data is routed to the sink node, there are far more data packets sent in the near sink area. Thus, by arranging more nodes near to the sink, the average amount of data transmitted by the node is reduced. Through the above scheme, the purpose of reducing the average one-hop delay is achieved.

2.2 Existing research related to energy consumption

Another important indicator in WSNs is energy consumption. If the initial energy is constant, the slower the sensor node consumes energy, the longer it can work, that is, the longer the lifetime [67]. To extend the lifetime as much as possible, the relevant researchers have proposed some common practices.

1. Duty cycle control is the most common method of saving energy consumption [64]. According to research, when the node stays asleep, the energy consumed is only about 1% of the state of waking. It is conceivable to let the node stay asleep in most time slots, which can effectively increase the energy utilization, but the delay becomes larger. Obviously, the first movers have realized this problem and have given strategies to deal with it. For example, the duty cycles of those nodes with residual energy are appropriately increased to keep them awake in more time slots, thereby reducing the latency of the transmitter nodes without changing the network lifetime.
2. The energy consumed by the node to listening can be reduced by reasonable control in a network with duty cycle control. For example, multiple possible recipients are prepared for each sender node [68]. If a node wants to transfer packets, as long as one receiver is awake, the transfer can proceed. By controlling the receiver's wake-up slot, the average duration of the listening by the sender node can also be reduced. For example, a receiver is always awake according to priority, in which case the node keeps listening for zero energy.
3. Data aggregation technology [66]. Through this technology, the amount of transmitted data and packet size are effectively reduced, thereby increasing energy utilization and reducing energy consumption.

2.3 Convergecast data collection

The slots and delays involved in the previous data collection strategy are not fixed. In different rounds, the delay in data collection may be doubled, and the slot of data operation is not fixed. Convergecast data collection is another data collection strategy in which the time slot of data operation for each node is determined, so the delay at which the data reaches the sink is determined [4, 5, 53–56].

Convergecast is ideal for data fusion networks. In such a network, m packets can be combined into one. There are many cases of Convergecast data collection, the most common of which are tree-based data collection [53] and cluster-based data collection [5, 56]. The main difference between these two types of strategies is that different network structures are used separately. (1) The main feature of the tree-based data collection strategy is to divide the network into a tree. The data collection process is the process of arranging slots for each node.

In order to save energy, its data collection also obeys the basic characteristics of the convergecast, that is, the node must wait until the data packets of all the child nodes are collected before sending data to the parent node. In such a strategy, the focus of the research is on how to control the wake-up time slots so that the time required for the entire data collection is the smallest. (2) In the cluster-based data collection strategy, each cluster first collects data within the cluster, and then performs inter-cluster data relay. Relatively speaking, the delay of the cluster-based convergecast strategy is relatively small. The reason is as follow: the tree-based convergecast needs to start with leaf nodes and collect data to the root node. In cluster-based converged broadcasts, the first phase of data collection in different clusters is simultaneous. In cluster-based convergecast, after cluster data collection, all CM nodes can be ignored, and the remaining CH nodes continue to form a tree.

However, as mentioned above, all current convergecast strategies are for networks where data is generated periodically. In many practical applications, the data of its nodes are generated randomly. Therefore, it is urgent to propose a strategy of low-energy consumption and low delay for networks that randomly generate data, which is a huge challenge.

2.4 Existing research related to matrix filling

Matrix filling (also known as matrix completion) technology is a common data processing optimization method [69]. Only a subset of known data is needed to recover other unknown data through matrix filling techniques. This feature makes matrix filling especially suitable for WSNs.

Based on matrix filling, an optimal unmanned aerial vehicle data collection trajectory (OUDCT) scheme was proposed [25]. In this scheme, the drone trajectory is optimized according to the matrix filling theory.

In Tan et al. [46], researchers proposed an adaptive collection scheme based on matrix completion (ACMC) scheme. The receiving scheme of the node is adaptively adjusted, thereby reducing the delay and increasing the energy utilization. Specifically, when the energy is sufficient, the node collects more data, and conversely, reduces the number of collected data.

In addition, matrix filling technology is combined with mobile crowdsourcing technology (an emerging data collection solution) to obtain a novel matrix completion technique-based data collection (MCTDC) scheme [63].

3 The system model and problem statement

3.1 The network model

We refer to a common clustering network model [56]. This is a circular-network with the sink node deployed at the center. The nodes are evenly distributed, and ρ is the density. The node generates a metadata packet with probability p in each round of data transmission. All sensor nodes are first grouped into groups according to their location in the network, each group being called a cluster. In a cluster, one node acts as a cluster head, denoted by CH, and other nodes naturally become members of the cluster, represented by CM. In particular, all nodes in the cluster take turns acting as CHs in different transmission rounds.

Data transmission can be separated into two parts, namely intra-cluster transmission and inter-cluster relay.

During the first part, the cluster member (CM) nodes will transmit the metadata packet to the CH node. In this model, it takes one time slot to transmit a packet, the energy consumed is constant, and the packet loss is not considered. After the CH node receives the packets, it aggregates them together to form a new packet, and then enters the inter-cluster relay process.

During inter-cluster relay, the sink node starts receiving packets of the CH nodes. In this process, the impact of packet size on transmission time is also ignored. The CH node located at the edge only collects the data packets sent by its CM, while the other CH nodes also receive the data packets sent by the outer CH nodes. After completing the above steps, they are sent out together with the data packets generated by themselves.

3.2 The clustering model

According to our previous research results, this paper uses a clustering method with different cluster radii [5]. The clusters of nodes close to sink have larger radii, as shown in Fig. 1.

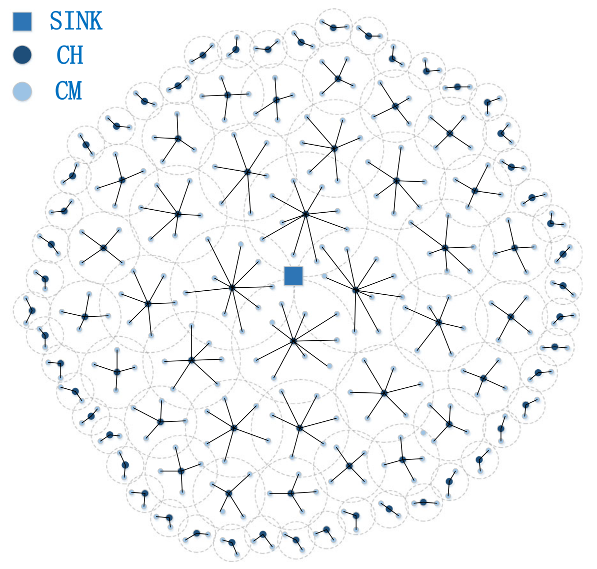

Fig. 1 Clustering method with different cluster radii

Nodes with small radii (away from the sink node) can complete data collection in the cluster in less time. At this point, the nodes with large radii (near the sink) are still collecting data in the cluster. Through reasonable control, when a CH node with a large cluster radius completes the cluster collection work, the data packet sent from the outer layer of the network just arrives at this CH node. In short, the in-cluster data transmission work and the inter-cluster relay work in the original series are changed to be parallel, thereby greatly reducing the delay and energy consumption.

For a network with radius R, it is divided into multiple layers by clustering. For nodes in different layers, the cluster radii are different and are represented as r_{h-1}, $r_{h-2}, \dots,$ r_1, r_0 from the inside to the outside. There is $\partial \sum_{i=0}^{h-1} r_i \geq R$, and ∂ is a constant less than or equal to 1. By clustering, we can convert all the nodes into one tree, which can be seen in Fig. 2.

In the layer i, the degree of the CH node ($i = 0, 1, 2, \dots h-1$) relative to the ordinary node is represented by d_i^{CM}. d_i^{CH} indicates the degree of the CH node relative to other CH nodes. In particular, $d_0^{CH} = 0$, d_{h-1}^{CH}, and d_{h-1}^{CM} represent the degrees of the sink node.

When the CH node on the layer i begins to send packet, the cluster head on the layer $i+1$ just completed the data collection in the cluster. In order to achieve this effect, the degree of the node must meet the following requirements:

$$d_1^{CM} = d_0^{CM} \tag{1}$$

$$d_i^{CM} = d_{i-1}^{CM} + d_{i-1}^{CH} \tag{2}$$

For ease of calculation, in Li et al. [5], $d_i^{CH} (i = 0, 1, 2, \dots h-1)$ is also specified as a constant, denoted by d^{CH}. Since the cluster radii of adjacent layers are different, the number of nodes contained in a cluster on layer i is not d_i^{CM}, but is expressed as:

$$n_i = d_i^{CM} - \rho\left(d_i^{CM} - d_{i-1}^{CM}\right) \tag{3}$$

Where $i = 1, 2, \dots h-1$. In particular, when $i = 0$, $n_0 = d_0^{CM}$.

Therefore, according to the density ρ, the radius of the cluster on the layer i ($i = 1, 2, \dots h-1$) can be obtained by:

$$r_i = \sqrt{\frac{n_i + 1}{\rho\pi}} \tag{4}$$

In particular, when $i = 0$:

$$r_0 = \sqrt{\frac{d_0^{CM} + 1}{\rho\pi}} \tag{5}$$

3.3 The matrix filling model

Matrix filling is a hot topic about data acquisition and data recovery after compression sensing [69]. Simply put, the purpose of matrix filling is to recover the complete data as much as possible through a small number of collected packets. The premise of matrix filling feasibility is the low rank of the matrix. Because of this feature, matrix filling can be well applied to WSNs.

Matrix filling is represented by an optimization problem:

$$\begin{array}{ll} \text{minimize} & \operatorname{rank}(X) \\ s.t. & X_{ij} = M_{ij} \quad (i,j) \in \Omega \end{array} \tag{6}$$

Where $rank(X)$ represents the rank of the matrix X, Ω represents a set of known elements, and M_{ij} represents the sum of known elements. Therefore, the problem of matrix filling is to complete the recovery of the missing matrix in the case of low rank.

As we all know, the solution of rank is an NP problem. Therefore, the relevant scholars proposed to use the

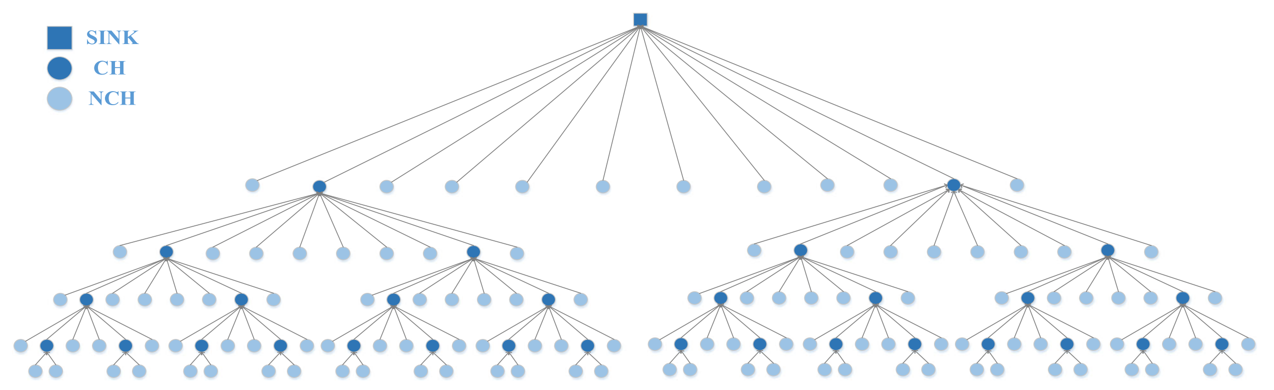

Fig. 2 Transform the network into a tree

nuclear norm to replace the rank of the matrix, and convert the above problem into:

$$\begin{array}{ll} \text{minimize} & \|X\|_* \\ s.t. & X_{ij} = M_{ij} \quad (i,j)\in\Omega \end{array} \tag{7}$$

Here, $\|X\|_*$ represents the kernel norm, which is equal to:

$$\|X\|_* = \sum_{k=1}^{n} \sigma_k(X) \tag{8}$$

Here, $\sigma_k(X)$ represents the kth singular value (arranged in descending order).

According to the research in Candès and Recht [69], for a matrix M with $n_1 \times n_2$ and rank r, m elements are known. With to constants C and c, when:

$$m \geq Cn^{5/4} r \log n \tag{9}$$

Here, $n = \max(n_1, n_2)$, the reduction degree is at least $1 - cn^3$ or more through the known m elements, that is, the above optimization problem is completely solved correctly.

In particular, if $r \leq n^{1/5}$, the probability of accurately restoring the matrix M can reach $1 - cn^3$, as long as:

$$m \geq Cn^{6/5} r \log n \tag{10}$$

With a small number of known elements, matrix filling can perfectly recover a matrix with a low rank. For example, if $r = O(1)$ or $r = O(\log n)$, we just need $n^{6/5}$ known elements to complete the matrix M [69].

3.4 The energy consumption model

The energy consumed is mainly due to [59] packet transmission, keeping the listening and switching states (awake/sleep). The calculation includes: CM sends a metadata packet; CH receives and sends a packet; node keep listening for a slot; and switching state once, as shown in Table 1.

3.5 Problem statement

For networks that randomly generate packets, we use the on-demand allocation method to arrange the transmission time slots of nodes in the DEEDC scheme. Compared with the general method, we hope to achieve the following optimization effects by dynamically allocating time slots:

Table 1 Energy consumption for different operations

Symbol	Meaning	Value (j)
$E_{s,\ CM}$	CM sends a metadata packet	4.5×10^{-5}
$E_{s,\ CH}$	CH sends a packet	6×10^{-5}
E_r	CH receives a metadata packet	4.5×10^{-5}
E_l	Node keep listening for a slot	4.5×10^{-5}
E_{ats}	Node switching state once	2×10^{-5}

1. Reduce the delay. Here, the time at which a node completes data collection is defined as a delay. The delay of node A is mainly determined by two aspects.

The first is to collect the packets inside the cluster. This is followed by the start time and duration of the packet sent by other CH nodes to A. By using T_{CM}, T_S, and T_L to represent them respectively, the delay is:

$$T = \begin{cases} T_{CM} + T_L & if\ T_{CM} \geq T_S \\ T_S + T_L & if\ t_{CM} < T_S \end{cases} \tag{11}$$

To reduce the delay:

$$\min(T) = \min(T_{CM}, T_S) + \min(T_L) \tag{12}$$

That is, the data collection in the cluster is completed as soon as possible, and the inter-cluster relay is completed as soon as possible.

2. Reduce energy consumption. We mainly consider the consumption to transmit data packets, to keep listening, and to switch state, and represent them by E^T, E^{LPL}, and E^{ats} respectively. Thus, the optimization of energy is:

$$\min(E) = \min\left(E^T + E^{LPL} + E^{ats}\right) \tag{13}$$

In order to increase the utilization of the energy, reducing the transmitted data, the listening time and the state switching are effective methods.

Therefore, the optimization goal can be summarized as:

$$\begin{cases} \min(T) \\ \min(E) \end{cases} \tag{14}$$

4 The design of DEEDC scheme

In a common clustering route, the cluster head node always reserves time for each node in the cluster to transfer data. For example, for a cluster with n nodes within a cluster, the CH node always allocates a time slot for each of them. In the corresponding time slot, the CM nodes send packets to the CH node. After n slots, the CH node completes the collection of data within the cluster and sends a data packet generated by itself along with other received data packets.

For networks where nodes randomly generate packets, fixed time slots are obviously not appropriate. For example, only $m(m < n)$ nodes generate data packets, but the CH node allocates n time slots for them. This allows the CH node to wake up in $n - m$ time slots without doing anything meaningful, wasting energy.

For such special cases, in order to be suitable for dynamic traffic wireless sensor networks, the DEEDC scheme arranges adaptive transmission times for CM nodes, that is, through the control of the CH node, only the time slots are arranged for the CM nodes that need to transmit data.

Simply put, the CM node that needs to send the data packet wakes up, and the other CM nodes continue to sleep. In the first time slot, the awake CM node sends a request-to-send message (RTS) to the corresponding CH node. In the RTS packet, the ID information of the CM node is mainly included. If the CH node receives a total of x RTS packets within a specified time, it will schedule slots for the corresponding CM nodes in the order in which the RTS packets are received. In detail, the ID information of the CM node is taken out from the received RTS data packet, and then all the ID information and the information of the allocated time slot are placed in one data packet and broadcasted. After receiving the broadcast, the CM node first finds the time slot information corresponding to its own ID, then sets the time of re-awake according to the information, and finally go to the sleep state. The CM node will wake up in the set time slot, sends a packet, and finally enters the sleep state. In particular, if a CM node is ready to send data at the second slot, it will eliminate the process of sleeping first and then waking up. The CH node arranges time slots for the CM node in the first time slot. In the next x time slots, the CH node sequentially receives the packets from corresponding CM node. After successfully receiving x packets, these packets are organized into one large packet. In particular, if the CH node itself also has a data packet, it is also included in the large packet. Through the above process, the node successfully completes the data collection in the cluster, and then, it is the relay process between the cluster heads.

In addition, we learned from the matrix-filling model in Section 3.3 that even if only a part of the data in the network is collected, the unknown data could be recovered. To this end, we have also added restrictions on the number of collected packets.

Assume that collecting x packets per round, the data not collected in the cluster can be completely recovered. Since the nodes in the network generate packets completely irregularly, it is impossible to generate exactly x packets each time, sometimes it may be more, sometimes it may be less. Therefore, for such a special case, the response strategy given by the DEEDC scheme is to make the average number of data packets collected per round greater than or equal to x. That is, a threshold y is found, which represents the threshold of the number of collected packets. Under this constraint, the workflow of the CH node and the CM node also needs to be adjusted accordingly.

The CH node first determines if it has generated a data packet. If so, $y' = y - 1$; otherwise, $y' = y$. After the CH node completes the reception of the RTS packet, the time slots are arranged for the first y' corresponding CM nodes in the order of arrival. As for other CM nodes that also sent RTS packets, the CN node schedules the slot to − 1, indicating that the CH node does not intend to receive its data packets in this round. After receiving the broadcast data, the CM node finds that the time slot is scheduled to be − 1, and then it directly discards the data packet to be sent, and turn to the sleep stage.

In Algorithm 1 and Algorithm 2, we respectively show the working process of CH node and CM node in a round of data transmission.

Algorithm 1: Workflow of cluster head node

```
1: For each new round Do
2:     Slot_num = 0, Node_num = 0
3: End for
4: Awake
5: When Recive RTS message then
6:     Node_num += 1
7:     ID[Node_num] = Get Node_id     // From RTS
8:     IF Node_num ≤ Max_nodenum then
9:         OP[Node_num] = Node_num
10:    End if
11:    Else
12:        OP[Node_num] = -1
13:    End else
14: End
15: Broadcast an assignment message     //Information including ID[ ] and OP[ ]
16: Slot_num += 1
17: For each time slot Do
18:      Receive a packet       // The sender is ID [Slot_num]
19:     Slot_num += 1
20:   IF Slot_num > Node_num or Slot_num > Max_nodenum then
21:         Integrate all data packets
22:         Break
22:     End if
23: End for
```

Among them, the symbols appearing respectively indicate:

Slot_num: indicates the sequence number of the time slot. Before the start of each round, it will be reset to 0
Node_num: indicates the number of CMs inside the cluster that are ready to send packets
RTS: request-to-send, which is a request message sent by CM that wants to send a data packet
ID[ ]: record the ID of the CM node that is ready to send the packet
Node_id: ID of the CM node, obtained from the corresponding RTS packet
OP[ ]: a transmission slot assigned to the CM node. If equal to − 1, the CH node in the round does not receive its data packet
Max_nodenum: the maximum number of nodes that the CH node can receive its data packets

We assume that there are eight CM nodes in a cluster, which are represented by CM_0, CM_1,..., CM_7. The working process of the CH node and the CM_i node ($i = 0, 1, 2, \ldots, 7$) are as follows.

First, it is first necessary to clarify which CM nodes' packet the CH node will receive. As shown in Figs. 3 and 4 out of eight CM nodes that generate data packets, all four nodes send RTS to the CH. With the matrix

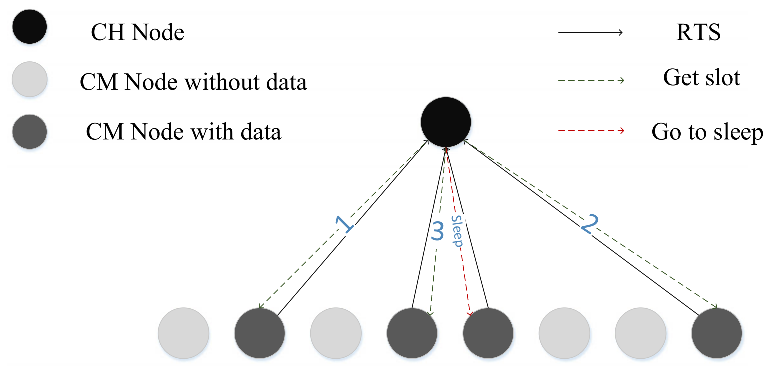

Fig. 3 Schematic diagram of the workflow of the CH node

filling theory, all data can be recovered by only receiving three packets. Then, the CH node arranges the transmission time slot for the CM nodes corresponding to the three RTS data packets received first, and as for the fourth CM node, let it directly enter the sleep state until the next data transmission.

The working process of the CM nodes are as follows:

```
Algorithm 2: Workflow of cluster member node
1:For each new round Do
2:   Slot_num = 0
3: End for
4: IF generated a packet then
5:   Awake
6:   Put Node_id into RTS
7:   Send RTS to CH node
8:   Receive broadcast message
9:   Get Node_num where ID[Node_num] = Node_id
10:   Get OP[Node_num]
11:   IF OP[Node_num] = -1or OP[Node_num] = 1 then
12:      Go to sleep
13:   End if
14:   IF Get  Slot_num = 1 then
15:      Send packet to CH node
16:      Go to sleep
17:   End if
18:   Else Get  Slot_num = OP[Node_num]
19:      Awake
20:      Send packet to CH node
21:      Go to sleep
22:   End else
23:End if
24:Else
25:   Keep sleeping
26:End else
```

Taking the above example as an example, the working process of the CM node is as shown in Fig. 4.

First, once the CM node wakes up (data is collected), it reports to the CH node (sends the RTS packet). In the next few time slots, the work is started according to the indication sent by the CH node. Assuming that the CH node does not intend to receive a packet from a certain CM node, for example, CM_4, it will receive an instruction to go directly to sleep state, and will not wake up after entering the sleep state until the next time data is collected.

Other CM nodes, such as CM_1, CM_3, and CM_7, will wake up in the scheduled time for data packet transmission.

For the convenience of the following description, we will explain the related symbols appearing below as follows:

According to the clustering model in Section 3.2, it is assumed that the network with radius R is divided into h layers by clustering. From the inside to the outside, the cluster head nodes on each layer are called CH_0, CH_1, …, CH_{h-1}, and the nodes in the cluster are called CM_0, $CM_1, \dots,$ CM_{h-1}. Among the clusters on the layer $i(i = 1, 2 \dots h - 1)$, n_{CM_i} CM_i and $n_{CH_{i-1}}$ CH_{i-1} transmit data packets to CH_i.

Specifically, there are n_{CM_0} CM_0 and zero CH nodes transmitting data packets to CH_0 on the 0th layer.

C_i^j indicates combination. For the sake of convenience, we introduce the symbol A_i^j:

$$A_i^j = C_i^j * p^i * (1-p)^{i-j} \tag{15}$$

Here, p represents the probability that a node generates a data packet. A_i^j indicates the probability that exactly j nodes in i nodes generate data packets, while other $i-j$ nodes do not generate data packets.

4.1 The maximum value of the collected packets

With the theoretical support of matrix filling, we can reduce the collected data and thus reduce energy consumption. However, be careful; otherwise, it cannot complete data recovery.

Theorem 1 *There is a two-dimensional matrix A whose size is $N * T$. Where N is the total number of nodes (excluding the sink node), T is the round of data transmission. There are n_i CM nodes in each cluster on layer i. Then, the number of packets that must be collected by CH_i in each round is:*

$$x_i = \frac{(n_i+1)(\max(N,T))^{6/5}}{N * T} \tag{16}$$

Proof According to 3.3, we only need $\max(N,T)^{6/5}$ known elements to recover the complete matrix. $\max(N,T)^{6/5}$ elements are generated from $N * T$ corresponding positions. We have $n_i + 1$ nodes in the cluster; it is necessary to acquire x_i known elements on average. Therefore, there are the following equations:

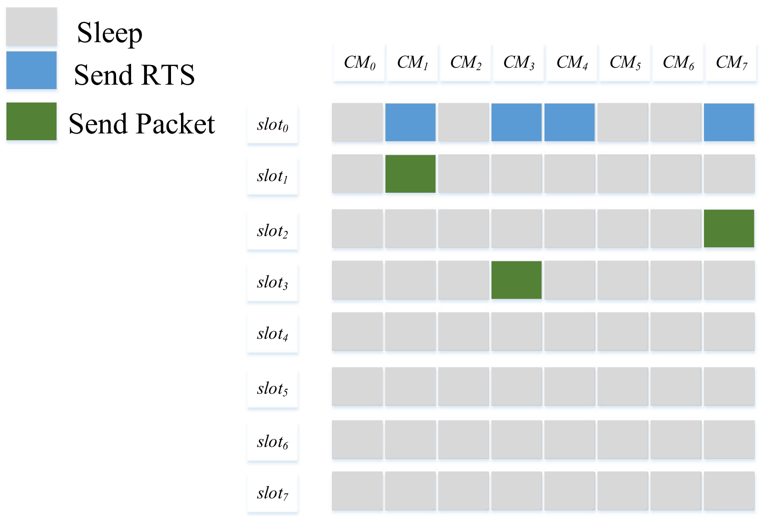

Fig. 4 Schematic diagram of the workflow of the CM node

$$\frac{(\max(N,T))^{6/5}}{N*T}=\frac{x_i}{n_i+1} \tag{17}$$

By shifting the item, you can finally prove the result in theorem 1.

In the actual situation, the number of packets collected by the CH is random. We specify a threshold. Under the control of the threshold, the average number packets collected per round is exactly x_i after rounding down, which satisfies the requirement of restoring the complete matrix in the matrix filling theory.

Theorem 2 *Use y_i to represent the threshold of packets collected by CH_i $(i = 0, 1, \ldots h - 1)$, which is:*

$$\begin{aligned} & y_i = \min(j_i) \\ & s.t. \sum_{j=0}^{j_i} j * P_{j_i} \geq x_i \end{aligned} \tag{18}$$

Where j_i indicates that a total of j data packets are generated in the cluster on the layer i, The clusters on layer i generate a total of j packets represented by j_i, and P_{j_i} represents the corresponding probability.

Proof The probability that a node has generated a packet is p $(0 < p < 1)$. For a cluster on the layer i, a total of $n_i + 1$ nodes may generate a data packet, that is, the value range of j_i is: $0 \leq j_i \leq n_i + 1$. In other words, at most $n_i + 1$ packets in the cluster (all CM nodes have generated packet) and at least zero packets (no CM node generates packet) are generated. The corresponding probability is:

$$P_{j_i} = \mathrm{A}_{n_i+1}^{j} \tag{19}$$

If y_i packets are from other nodes in the cluster (non-CH), then a total of $1 + y_i + 1$ slots are needed. The CH node that wants to transfer data sends a RTS, and then the CH nodes allocate time slots to them by broadcasting. In the next y_i time slots, the intra-cluster nodes sequentially send packets. Finally, the CH organizes the received data packets and its own generated data packets (if any) into one data packet and sends it out. If the CH node generates a data packet, it needs $1 + (y_i - 1) + 1$ time slot.

In particular, the last time slot is calculated in the next layer (because in this time slot, the transmission operation of this layer and the reception operation in the next layer are performed sequentially, no need to repeat the calculation).

The probability of including the CH node among the y_i nodes that generate the data packet is:

$$P_i^{i-CH} = \frac{C_{n_i}^{y_i-1}}{C_{n_i+1}^{y_i}} \tag{20}$$

The probability of not including the CH is:

$$P_i^{ni-CH} = \frac{C_{n_i}^{y_i}}{C_{n_i+1}^{y_i}} \tag{21}$$

Therefore, the average time for the CH node to complete data collection in the cluster is:

$$t_i = P_i^{i-CH} * y_i + P_i^{ni-CH} * (1 + y_i) \tag{22}$$

In particular, if $t_i \geq n_i$, we do not use the above method to allocate time slots for nodes on the layer i, but to fix the time slots. Even if the corresponding intra-cluster node does not transmit data packets in the time slot, other nodes cannot occupy this time slot.

4.2 Number of transmitted packets

The energy consumed to transmit packets is mainly considered: the CM node and the CH node send data packets, and the CH node receives the data packet. In addition, the CM node will also send RTS packets, and the CH node will send a broadcast. Since these data packets are smaller than the foregoing three, both in terms of packet size and number, they are not considered in this paper.

Theorem 3 *Use $N^s_{CM_i}$, $N^s_{CH_i}$, and $N^r_{CH_i}$ to represent the number of packets sent by CM_i, sent and received by CH_i $(i = 0, 1, \dots h-1)$.*

$$N^s_{CM_i} = P_i^{ni-CH} * \left[\sum_{k=0}^{y_i-1} k * A^k_{n_i} + y_i * \left(1-\sum_{k=0}^{y_i-1} A^k_{n_i}\right)\right] + P_i^{i-CH} * \left[\sum_{k=0}^{y_i-1} (k-1) * A^{k-1}_{n_i} + (y_i-1) * \left(1-\sum_{k=0}^{y_i-1} A^{k-1}_{n_i}\right)\right] \tag{23}$$

$$N^s_{CH_i} = (1-p) * N^r_{CH_i} + p * \left(N^r_{CH_i} + 1\right) \tag{24}$$

$$N^r_{CH_i} = N^s_{CM_i} + N^s_{CH_{i-1}} \tag{25}$$

Especially:

$$N^r_{CH_0} = N^s_{CM_0} \tag{26}$$

Proof The number of CM_i nodes that send metadata packets to CH_i is equal to the number of transmitted metadata packets. According to the matrix filling theory, suppose the number of packets generated in the cluster exceeds y_i, the CH nodes do not need to collect all. If the CH node generates a data packet, then at most only $y_i - 1$ data packets need to be collected in the cluster. Otherwise, at most y_i are collected. Thus, $N^s_{CM_i}$ is equal to:

$$N^s_{CM_i} = P_i^{ni-CH} * \left[\sum_{k=0}^{y_i-1} k * A^k_{n_i} + y_i * \left(1-\sum_{k=0}^{y_i-1} A^k_{n_i}\right)\right] + P_i^{i-CH} * \left[\sum_{k=0}^{y_i-1} (k-1) * A^{k-1}_{n_i} + (y_i-1) * \left(1-\sum_{k=0}^{y_i-1} A^{k-1}_{n_i}\right)\right] \tag{27}$$

If the CH node does not generate a packet, then it sends the same number of packets as received. Otherwise, it needs to send one more. Which is:

$$N^s_{CH_i} = (1-p) * N^r_{CH_i} + p * \left(N^r_{CH_i} + 1\right) \tag{28}$$

The data packets received by the CH_i come from two parts: CM_i and CH_{i-1}. Which is:

$$N^r_{CH_i} = N^s_{CM_i} + N^s_{CH_{i-1}} \tag{29}$$

In particular, CH_0 only needs to receive data from CM_0:

$$N^r_{CH_0} = N^s_{CM_0} \tag{30}$$

4.3 Delay

In the network model described above, after the CH node receives the data packets of the nodes in the cluster, it may take a while to forward them. For example, the CH node will go to sleep after completing the data collection in the cluster. Wait until the CH node outside it is ready to send packets and then switch to the awake state. After the inter-cluster packet relay is completed, the packet will be sent to the next hop. In this case, the delay under the general definition is not accurate enough.

Therefore, we redefine the delay, that is, the time at which the (CH) node completes all its collection work. For a sink node, that is, when it receives all the packets sent from the network.

Theorem 4 *Assuming that the time slots is cleared before the new round starts, the time in which the CH_i node $(i = 0, 1, \dots h-1)$ completes all collection operations (including collecting data packets sent by CM_i and CH_{i-1}) can be expressed as:*

$$T_i = max\left(T_{i-1}, T_i^{CM}\right) + d^{CH} * N^s_{CH_{i-1}} \tag{31}$$

Where d^{CH} represents the degree between the cluster heads, $N^s_{CH_{i-1}}$ is the amount of data sent by CH_{i-1}, and T_i^{CM} is the time in which CH_i completes the data collection work in the cluster (y_i is the threshold of the packet collected by CH_i), and:

$$T_i^{CM} = 1 + y_i \tag{32}$$

Proof The packet collection process in the network can be summarized as follows: first, in the first time slot, the node that wants to send data transmits a RTS to the corresponding CH node, and acquires corresponding time slot information from the broadcast data packet that is sent back. If the CH node decides not to receive the data of node A, then A will directly discard the collected data packet and go to sleep. Instead, to reduce energy consumption, first turn to the sleep state.

For CH_i, it may receive k packets from the CM_i node, where $k = 0, 1, 2, \dots, y_i$. $k = y_i$ if and only if the CH_i does not generate a data packet. Therefore, the average slot in which CH_i completes the data collection can be expressed as:

$$T_i^{CM} = 1 + \sum_{k=0}^{y_i-1} k * A_{n_i}^k + y_i * P_i^{ni-CH} * \left(1 - \sum_{k=0}^{y_i-1} A_{n_i}^k\right) \tag{33}$$

The above average time slots do not represent well the time slots in the actual situation. We fixed the number of slots of data collected by CH_i to y_i which ensures that all CH_i can complete data collection as required. If CH_i completes its work before the time slot y_i, it will go to sleep. Therefore, the time for CH_i to complete the data collection in the cluster can be simplified to:

$$T_i^{CM} = 1 + y_i \tag{34}$$

After the CH node collects the data packets inside the cluster, it also needs to continue to accept the packets from the CH nodes located at upper layer. Since the CH_0 does not need to receive packets sent by other CH nodes, therefore:

$$T_0 = T_0^{CM} + 0 \tag{35}$$

For other CH_i nodes ($i = 1, 2, \ldots, h-1$), the time to start receiving the packets sent by CH_{i-1} depends on the time when CH_i completes the data collection work in the cluster (T_i^{CM}) and the time when all the data collection work is completed by CH_{i-1} (T_{i-1}). That is, only after completing the above two parts of work, the CH node can start the relay process between clusters.

The time required for CH_i to receive the packets sent by CH_{i-1} depends on the number of CH_{i-1} nodes (d^{CH}) and the amount of data sent by each CH_{i-1} ($N_{CH_{i-1}}^s$). Thus, the time for CH_i to complete all collection work (including collecting data packets among CM_i nodes and CH_{i-1} nodes) can be expressed as:

$$T_i = \max\left(T_{i-1}, T_i^{CM}\right) + d^{CH} * N_{CH_{i-1}}^s \tag{36}$$

It should be noted that when CH_{i-1} sends a packet to CH_i, we do not have to send the RTS data packet to allocate the time slot first. There are two main reasons:

First, the packet sent by CH_{i-1} is a summary of multiple previous packets and must be accepted.

Secondly, the probability that a CH_{i-1} node has a data packet to send is very large, especially as the number of layers increases, the probability approaches infinitely to one.

In this case, the practice of data collection within clusters is always counterproductive, increasing energy consumption and latency. Therefore, the best method is to arrange a fixed time slot directly for each CH_{i-1}.

4.4 Energy consumption

In our research, the energy consumption mainly comes from three parts (packet transmission, keep listening, as well as switching status). As for other energy consumption, they can be ignored.

Theorem 5 *For a node on layer* ($i = 0, 1, \ldots h-1$), *the energy consumed is mainly: the energy consumed to transfer packets* (E_i^T), *to keep listening* (E_i^{LPL}) *and by switching state* (E_i^{ats}):

$$E_i = E_i^T + E_i^{LPL} + E_i^{ats} \tag{37}$$

Here:

$$E_i^T = \frac{N_{CM_i}^s * E_{s,CM} + N_{CH_i}^s * E_{s,CH} + N_{CH_i}^r * E_r}{n_i + 1} \tag{38}$$

$$E_i^{LPL} = \frac{\left(N_{CM_i}^{lpl} + 1\right) * E_l}{n_i + 1} \tag{39}$$

$$E_i^{ats} = \frac{\left(N_{CM_i}^{sta} * n_i + N_{CH_i}^{sta}\right) * E_{ats}}{n_i + 1} \tag{40}$$

$N_{CM_i}^{lpl}$ indicates the time the node keeps listening:

$$N_{CM_i}^{lpl} = \sum_{k=0}^{n_i} k * A_{n_i}^k \tag{41}$$

$N_{CM_i}^{sta}$ and $N_{CH_i}^{sta}$ indicates how many times the state of the CM node and the CH node switched:

$$N_{CM_i}^{sta} = p * \sum_{k=1}^{n_i} A_{n_i}^k * \left(1 * \frac{1}{k} + 2 * \frac{k-1}{k}\right) \tag{42}$$

$$N_{CH_i}^{sta} = \begin{cases} 1 & \text{when} \quad T_{i-1} \le y_i \\ 2 & \text{when} \quad T_{i-1} > y_i \end{cases} \tag{43}$$

Proof According to the Theorem 3 in Section 4.2, the data packets transmitted by nodes are mainly divided into three categories. In the layer i, the packets sent by the CM node is $N_{CM_i}^s$, sent and received by the CH node is $N_{CH_i}^s$ and $N_{CH_i}^r$. So for a node on layer i, the energy consumed for transmission is:

$$E_i^T = \frac{N_{CM_i}^s * E_{s,CM} + N_{CH_i}^s * E_{s,CH} + N_{CH_i}^r * E_r}{n_i + 1} \tag{44}$$

Because of the two mutually influential processes of listening and state switching, they are discussed together.

In comparison, the process of keeping the CM node listening is much simpler. After sending the RTS packet, it keeps listening until the broadcast data is received. Therefore, the average number of CM nodes that are listening is:

$$N_{CM_i}^{lpl} = \sum_{k=0}^{n_i} k * A_{n_i}^k \tag{45}$$

The initial state of the nodes in these clusters is the sleep state, and the nodes with data to be sent will be switched to the awake state in the first time slot. These nodes will then remain in sleep until the corresponding time slot arrives. These nodes will go to sleep again after the data has been transferred. In particular, if a transmission time slot allocated by a CM node happens to be the second time slot, then the node will remain awake in the second slot.

In summary, the number of state transitions has the following possibilities: (1) zero times, that is, no data transmission; (2) once, that is, the assigned time slot is the second time slot; (3) twice, that is, nodes assigned to other time slots. Therefore, the average number of times the node switches states is:

$$N_{CM_i}^{sta} = p * \sum_{k=1}^{n_i} A_{n_i}^k * \left(1 * \frac{1}{k} + 2 * \frac{k-1}{k}\right) \tag{46}$$

Where p is the probability of generating a packet.

The CH_i node keeps listening for a duration of one slot (used to receive RTS packets in the first slot), which is:

$$N_{CH_i}^{lpl} = 1 \tag{47}$$

The CH_i node wakes up at the beginning of each round of data transmission, and then its state switching has two cases. If the CH_i completes the in-cluster receipt collection, the CH_{i-1} node has not completed the data collection. Then the CH_i will switch to sleep state first, and wait until the CH_{i-1} starts sending data before it wakes up again. In this case, the state is switched twice. Conversely, the CH_i will continue to receive packets from the CH_{i-1}. In this case, the CH_i only needs to switch the state once.

$$N_{CH_i}^{sta} = \begin{cases} 1 & \text{when} \quad T_{i-1} \le y_i \\ 2 & \text{when} \quad T_{i-1} > y_i \end{cases} \tag{48}$$

Where T_{i-1} represents the time when the CH_{i-1} (i = 1, 2, … , $h-1$) node completes the collection of all data, and y_i represents the time when the CH_i node completes the data collection in the cluster.

Due to listening, the average energy consumed is expressed as follows:

$$\begin{aligned} E_i^{LPL} &= \frac{\left(N_{CM_i}^{lpl} + N_{CH_i}^{lpl}\right) * E_l}{n_i + 1} \\ &= \frac{\left(N_{CM_i}^{lpl} + 1\right) * E_l}{n_i + 1} \end{aligned} \tag{49}$$

Similarly, due to the switching state, the average energy consumed is:

$$E_i^{ats} = \frac{\left(N_{CM_i}^{sta} * n_i + N_{CH_i}^{sta}\right) * E_{ats}}{n_i + 1} \tag{50}$$

Thus, by adding the above three energies, the total energy consumed on layer i can be obtained.

$$E_i = E_i^T + E_i^{LPL} + E_i^{ats} \tag{51}$$

5 The experimental results and analysis

At this part, we analyzed and compared the proposed strategy, including delay and energy consumption. Before that, we first complete some basic work.

In the previous chapter, the method of calculating the energy consumed and the delay generated due to the transfer data are given in detail. We will give the relevant performance indicators when the DEEDC scheme is not used, for comparison.

Theorem 6 *In the case of selecting to receive all generated data packets,* $N_{CM_i}^s$, $N_{CH_i}^s$, *and* $N_{CH_i}^r$ $(i = 0, 1, \ldots h-1)$ *should be expressed as*:

$$N_{CM_i}^{s'} = \sum_{k=0}^{n_i} k * A_{n_i}^k \tag{52}$$

$$N_{CH_i}^{s'} = (1-p) * N_{CH_i}^{r'} + p * \left(N_{CH_i}^{r'} + 1\right) \tag{53}$$

$$N_{CH_i}^{r'} = N_{CM_i}^{s'} + N_{CH_{i-1}}^{s'} \tag{54}$$

When $i = 0$:

$$N_{CH_0}^{r'} = N_{CM_0}^{s'} \tag{55}$$

Proof As we all known, there are n_i CM nodes in the cluster on the layer i; they generated k packets ($k \in [0, n_i]$). Multiply the corresponding probability and sum to get a weighted average of packets generated by CM_i:

$$N_{CM_i}^{s'} = \sum_{k=0}^{n_i} k * A_{n_i}^k \tag{56}$$

If the CH itself generates a data packet (probability is p), then the number of packets sent by it is incremented by one based on received packets. The opposite is the packets it receives:

$$N^{s'}_{CH_i} = (1-p) * N^{r'}_{CH_i} + p * \left(N^{r'}_{CH_i} + 1\right) \tag{57}$$

The data packets received by the CH_i node, one part comes CM_i, and the other part comes from CH_{i-1}:

$$N^{r'}_{CH_i} = N^{s'}_{CM_i} + N^{s'}_{CH_{i-1}} \tag{58}$$

In addition, since the CH_0 node does not need to receive packets from other CH nodes, there is:

$$N^{r'}_{CH_0} = N^{s'}_{CM_0} \tag{59}$$

Theorem 7 *In the case of accepting all the generated data packets, the time for the* CH_i *node* ($i = 0, 1, \ldots h-1$) *to complete all data collection work is*:

$$T'_i = max\left(T'_{i-1}, T^{CM'}_i\right) + d^{CH} * N^{s\prime}_{CH_{i-1}} \tag{60}$$

Here, $T^{CM'}_i$ is the time to complete the collection of data packets in the cluster.

Proof The cluster head node schedules a fixed time slot for all CM nodes. Therefore, there is no need to take out a single time slot to arrange the time slots. That is, the time at which the cluster head node completes the data collection work in the cluster is:

$$T^{CM'}_i = n_i \tag{61}$$

The time when the node CH_i starts to collect the data packet of CH_{i-1}, is after the CH_i node completed the in-cluster data collection and the CH_{i-1} nodes finished collection work. That is, the time slot in which the CH_i node collects all data is:

$$T'_i = max\left(T'_{i-1}, T^{CM'}_i\right) + d^{CH} * N^{s\prime}_{CH_{i-1}} \tag{62}$$

In particular, when $i = 0$, CH_0 only needs to receive data in the cluster:

$$T'_0 = T^{CM'}_0 + 0 = n_0 \tag{63}$$

Theorem 8 *In the case of acceptance of all generated data, the average energy consumption of nodes on layer* i ($i = 0, 1, \ldots h-1$) *can be expressed as*:

$$E_i{}' = E^{T'}_i + E^{LPL'}_i + E^{sta'}_i \tag{64}$$

Here, $E^{T'}_i$, $E^{LPL'}_i$ and $E^{sta'}_i$ indicate the energy consumption to transmits data packets, keeps listening and switching states without using the DEEDC scheme, respectively. Moreover, their values are:

$$E^{T'}_i = \frac{N^{s'}_{CM_i} * E_{s,CM} + N^{s'}_{CH_i} * E_{s,CH} + N^{r'}_{CH_i} * E_r}{n_i + 1} \tag{65}$$

$$E^{LPL'}_i = \frac{N^{lpl'}_{CH_i} * E_l}{n_i + 1} \tag{66}$$

$$E^{sta'}_i = \frac{\left(N^{sta'}_{CM_i} * n_i + N^{sta'}_{CH_i}\right) * E_{ats}}{n_i + 1} \tag{67}$$

Proof Similar to the Theorem 5, the average energy consumption of node because of transmit packets can be expressed as:

$$E^{T'}_i = \frac{N^{s'}_{CM_i} * E_{s,CM} + N^{s'}_{CH_i} * E_{s,CH} + N^{r'}_{CH_i} * E_r}{n_i + 1} \tag{68}$$

Since all CM nodes are assigned a slot, they wake up and send data in the corresponding time slot (it will not wake up if no data packets are sent), so the listening time is zero.

In a certain time slot, the corresponding CM node sends a packet, and then the CH node receives it; otherwise, it keeps listening state. That is, if there are k nodes send packets to the CH, the CH needs to keep listening in the remaining $n_i - k$ slots. That is, the average listening time of CH node is:

$$N^{lpl'}_{CH_i} = \sum_{k=0}^{n_i} (n_i - k) * A^k_{n_i} \tag{69}$$

Thus, the average energy consumed due to the listening is ($i = 0, 1, \ldots h-1$):

$$E^{LPL'}_i = \frac{N^{lpl'}_{CH_i} * E_l}{n_i + 1} \tag{70}$$

The state switching of the CM node is relatively simple. If no data is sensed, the node does not need to wake up and the number of state switching is zero. If there is data to be sent, it wakes up in the corresponding time slot and enters the sleep state after the data transmission is completed. The number of state switches is once. Therefore, the times of CM node switches states is:

$$N^{sta'}_{CM_i} = p * \sum_{k=1}^{n_i} A^k_{n_i} \tag{71}$$

If the two processes of data collection in the cluster and relay between clusters are continuous, then the CH node only needs to switch the state once, otherwise twice. The times the CH node switches states is as fallow:

$$N^{sta'}_{CH_i} = \begin{cases} 1 & \text{when} \quad T'_{i-1} \leq n_i \\ 2 & \text{when} \quad T'_{i-1} > n_i \end{cases} \tag{72}$$

Where T'_{i-1} represents the time when the node CH_{i-1} completes all data collection without using the DEEDC scheme, and y_i represents the time that the node CH_i completes the data collection within the cluster.

Therefore, the average energy consumption due to switching states is:

$$E_i^{sta'} = \frac{\left(N_{CM_i}^{sta'} * n_i + N_{CH_i}^{sta'}\right) * E_{ats}}{n_i + 1} \tag{73}$$

In summary, the average energy consumption on the layer i $(i = 0, 1, \ldots h - 1)$ is:

$$E_i' = E_i^{T'} + E_i^{LPL'} + E_i^{sta'} \tag{74}$$

There is a network with $R = 500$m, the sink node is in the middle. The density between nodes is $\rho = 0.002$, and the simulation time is 1000 data transmission rounds.

5.1 Analysis of the number of packets

In Fig. 5, the number of data packets transmitted by the node when d_0^{CM} is different (10 and 12 respectively) in the case where the degree between cluster heads is the same ($d^{CH} = 3$) is given.

As the distance from the sink node is shortened, the number of transmitted packets gradually expands, as shown in Fig. 5. In addition, when d_0^{CM} becomes larger, the number of transmitted packets also increases, and the more layers, the more obvious the increase. This is because a part of the data packet on the layer i comes from the previous i layers.

In the case where d_0^{CM} is the same ($d_0^{CM} = 10$), when the degree d^{CH} between the cluster heads is different (equal to 3 and 4 respectively), the number of transmitted packets can be seen from Fig. 6. The larger d^{CH}, the more packets are sent.

Without the DEEDC scheme, the number of data packets transmitted by the node is as shown in Fig. 7 ($d^{CH} = 3$, $d_0^{CM} = 10$).

Nodes that are farther away from the sink node send fewer packets. Conversely, by using the DEEDC scheme, the reduction is very significant, as shown in Fig. 8.

From Fig. 8, we can draw two conclusions: first, the number of packets sent by the CH node is gradually reduced, as the number of layers increases. This is because the closer to the sink node, the more packets are transmitted by the cluster head node. Secondly, the degree of decline in the number of transmitted packets generally shows a downward trend, but does not want to show a monotonous decreasing trend like the CH node. The reason is that in reality, the number of packages cannot be a decimal, which requires us to round up the theoretical calculation results, resulting in errors. In comparison, the CM node sends smaller packets, so this error is more obvious when studying the rate of decline.

By using the DEEDC scheme, the number of transmitted packets is greatly reduced (more than 45%), thereby reducing the delay and energy consumption generated by transmitting data packets. At the same time, due to the support of the matrix padding technology, the sink node can still recover the complete data in the case of reduced data packets.

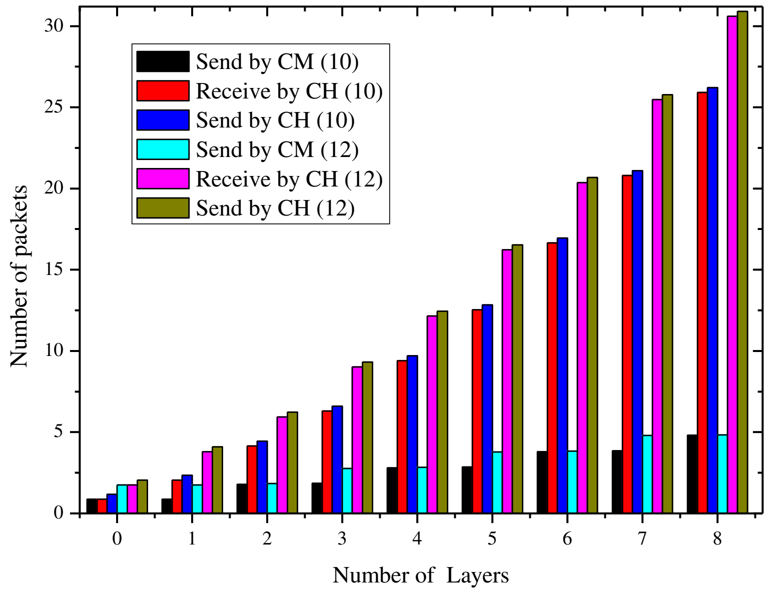

Fig. 5 The number of packets when d_0^{CM} takes different values

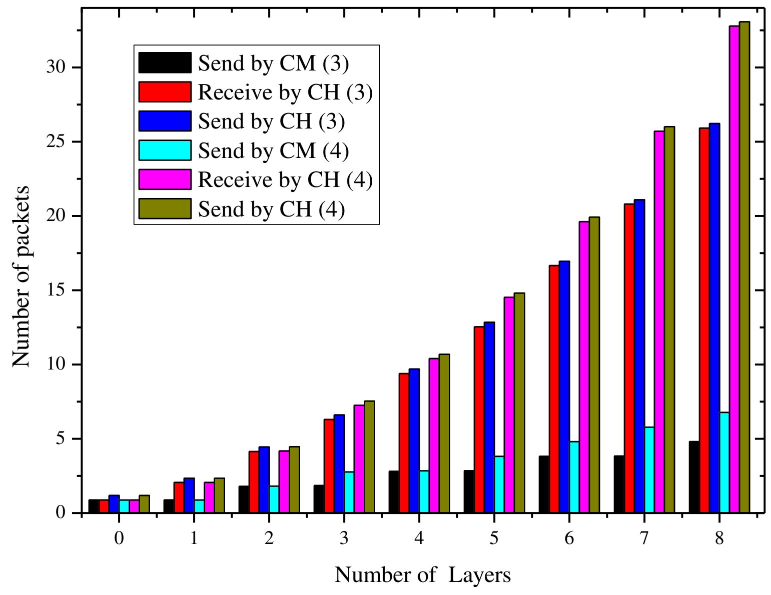

Fig. 6 The number of packets when d^{CH} takes different values

5.2 Analysis of delay

In Section 4.3, we give the calculation of the delay in the DEEDC scheme. When d^{CH} are the same ($d^{CH} = 3$) and d_0^{CM} is not the same, the delay of the nodes can be seen from Fig. 9.

The larger the d_0^{CM}, the greater the delay. When d_0^{CM} in the network is equal (10) and the degrees d^{CH} between the clusters are not the same, the delay can be seen from Fig. 10. Similar to Fig. 9, the larger the d^{CH}, the greater the delay. The maximum delay in the network is also reduced with the number of layers of the cluster decreases, which is because the number of hops of the relay is reduced.

In the case where the DEEDC scheme is not used, the delay can be seen from Fig. 11.

Intuitively, as in the case of the DEEDC scheme, the larger the d_0^{CM}, the greater the delay. In contrast, after using the DEEDC scheme, the percentage of delay reduction of a node is as shown in Fig. 12.

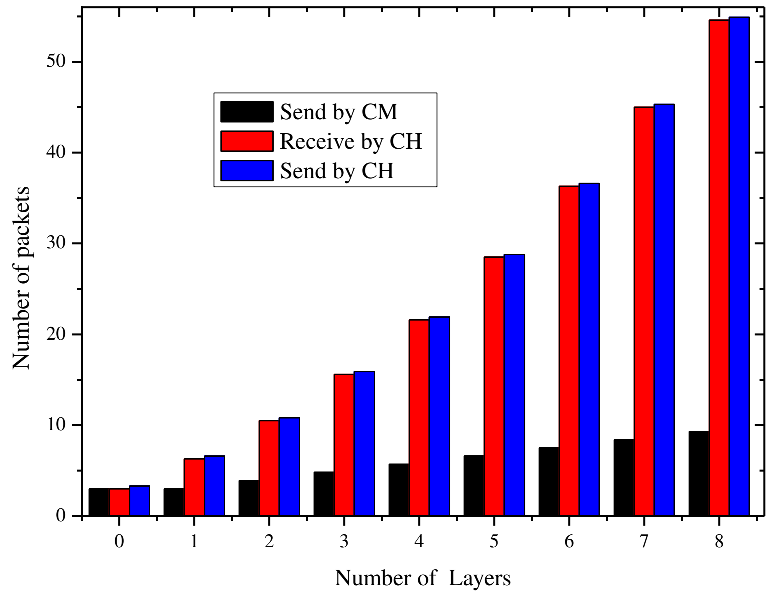

Fig. 7 The number of packets without DEEDC scheme

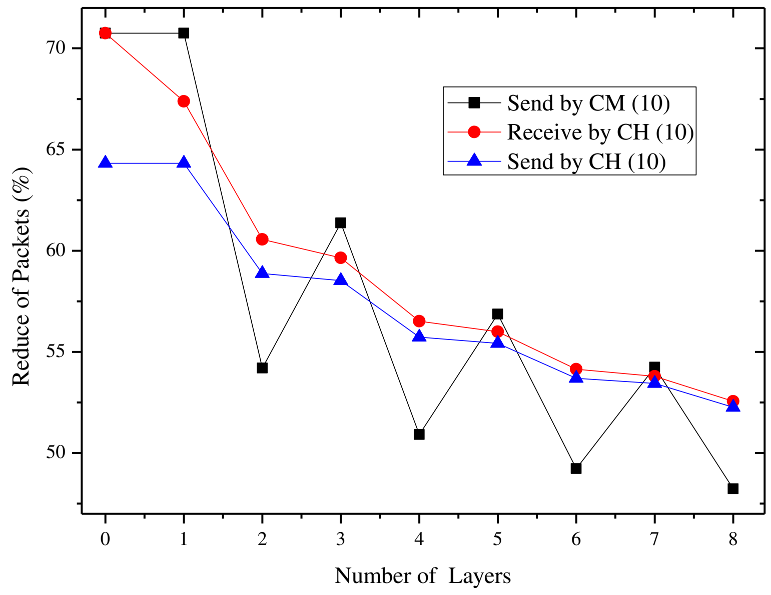

Fig. 8 The reduced percentage of the number of packets

The delay of the node is reduced most at a position away from the sink node. This is mainly because the nodes in this part are in the beginning stage of the packet forwarding process, the data collection work can be completed quickly, and the delay is small, so that the slight reduction can also make the degree of decline larger. However, compared to the decline in this part of the delay, the decline of the node near the sink is more valuable. For example, in Fig. 12, when $d_0^{CM} = 10$, the latency of the sink node is reduced by 56.06%.

5.3 Analysis of energy consumption

There are three main parts of the node's energy consumption, as shown in Fig. 13 ($d^{CH} = 3$, $d_0^{CM} = 10$). From Fig. 11, we can see that the energy consumption because of transfer packets is different at different layers. Obviously,

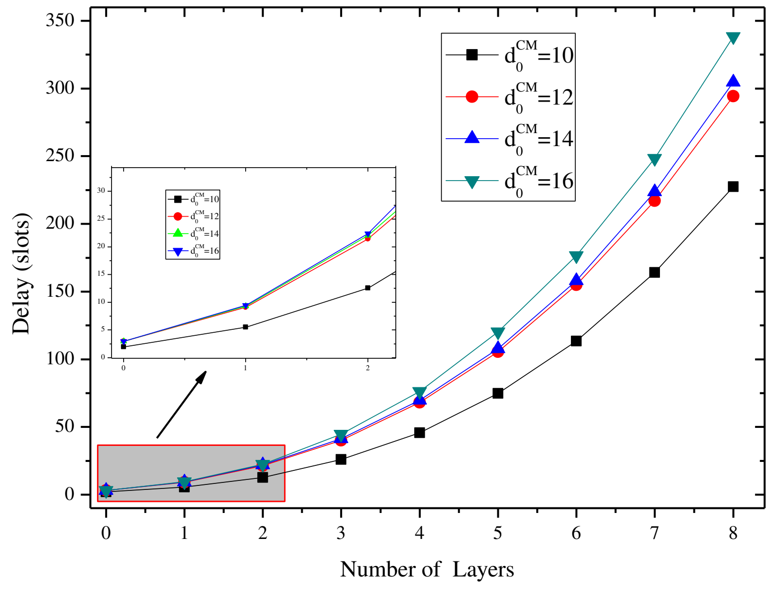

Fig. 9 Delay with different d_0^{CM}

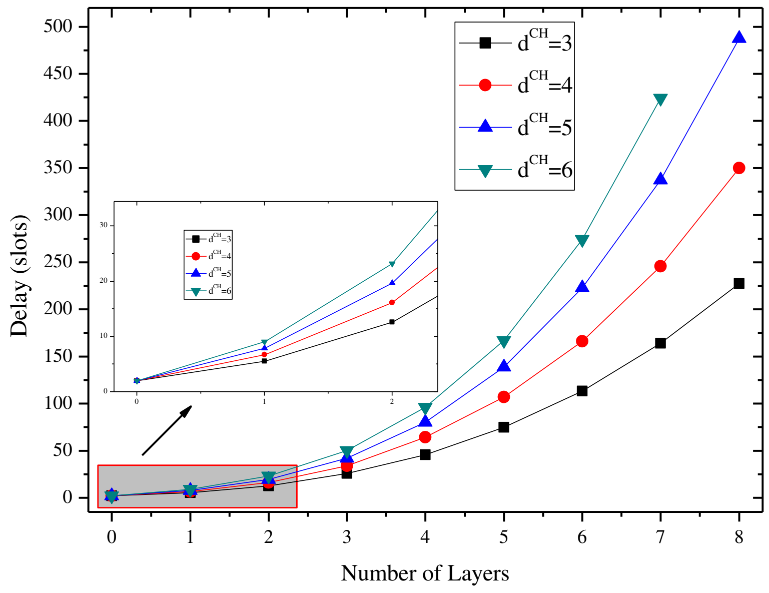

Fig. 10 Delay with different d^{CH}

this is due to the continuous accumulation of packets. Unlike transmitting data, the energy consumption to keep listening and switching states is not much different on different layers.

Figure 14 shows the total energy consumption when the degrees between the cluster heads are the same ($d^{CH}=3$) and d_0^{CM} are different.

As can be seen from Fig. 14, it is difficult to see the direct relationship between the value of d_0^{CM} and the total energy consumption. This phenomenon can be understood as the value of d_0^{CM} mainly determines the number of packets received by CH.

When d_0^{CM} takes the same value ($d_0^{CM}=10$) and d^{CH} is different, the total energy consumption is as shown in Fig. 15. The energy consumed by the node decreases as d^{CH} increases. The larger the d^{CH}, the more CM nodes, the smaller the average energy consumption to transfer packets, and ultimately, the smaller the total energy consumption.

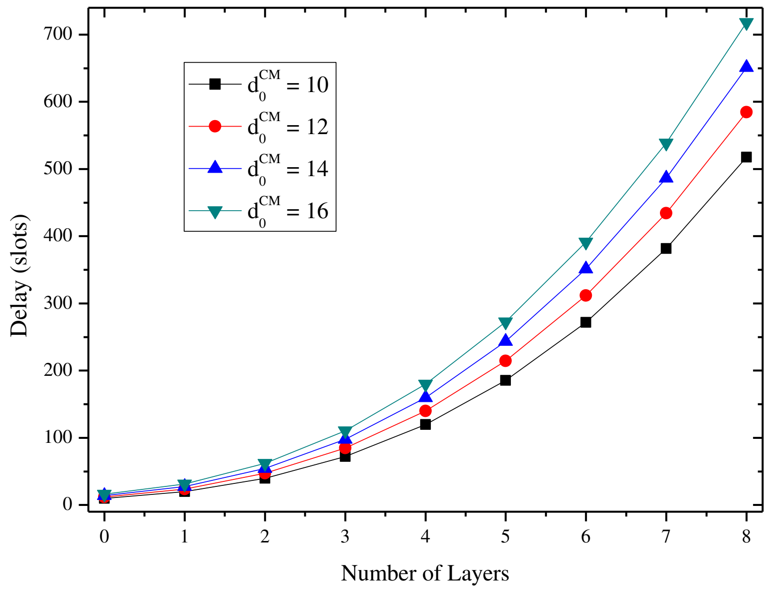

Fig. 11 Delay without DEEDC scheme

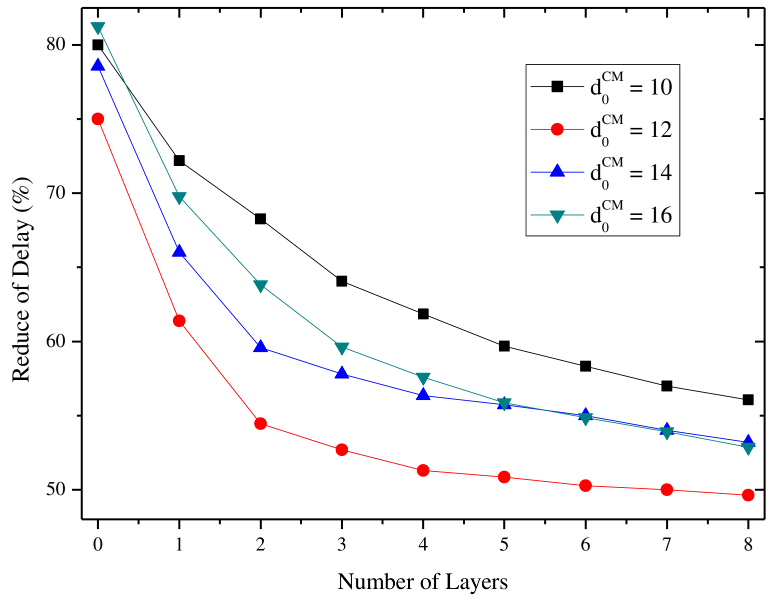

Fig. 12 The reduced percentage of the delay

Under the same conditions ($d^{CH} = 3$, $d_0^{CM} = 10$), when all the packets are collected, the energy consumption shown in Fig. 16.

Obviously, compared to the case of using the DEEDC scheme (see Fig. 13), the energy consumed is roughly the same as that of the DEEDC scheme. The difference is that the energy consumption value becomes smaller after using the DEEDC scheme.

Under the same conditions, the energy consumption changes by using the DEEDC scheme are shown in Figs. 17 and 18.

From them, it cannot be seen that there is a direct linear relationship between the percentage of energy reduction and d_0^{CM} and d^{CH}, but the percentage of energy consumption reduction is at least 40%, which effectively saves energy.

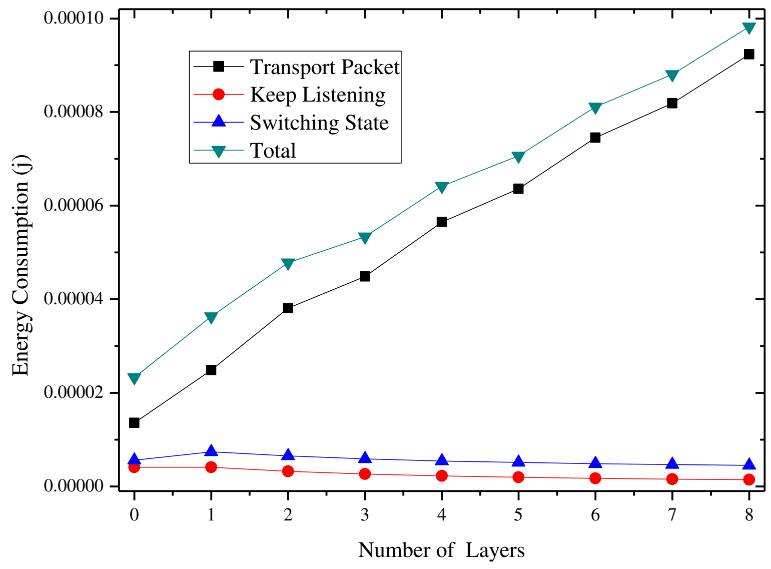

Fig. 13 Different energy consumed by nodes on different layers

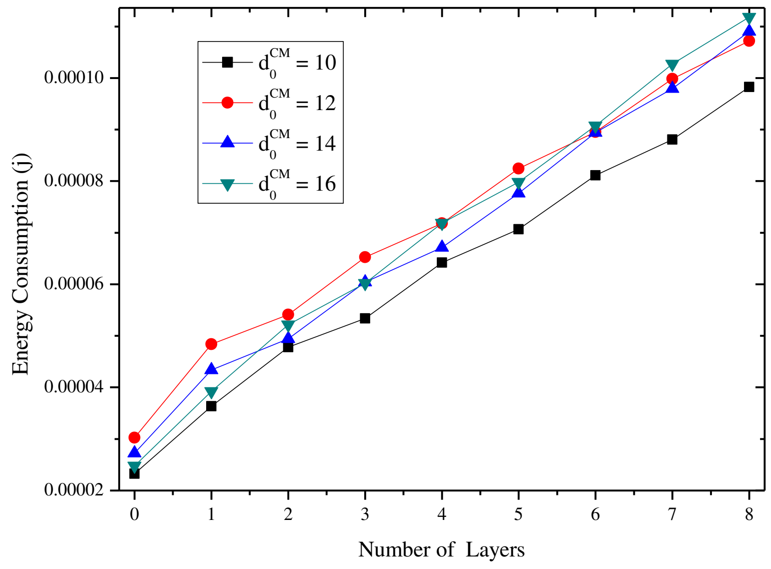

Fig. 14 Energy consumption when d_0^{CM} takes different values

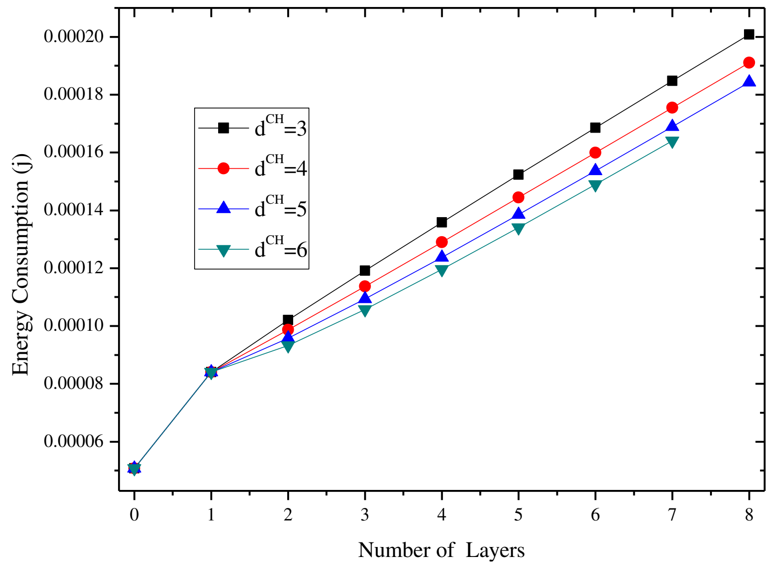

Fig. 15 Energy consumption when d^{CH} takes different values

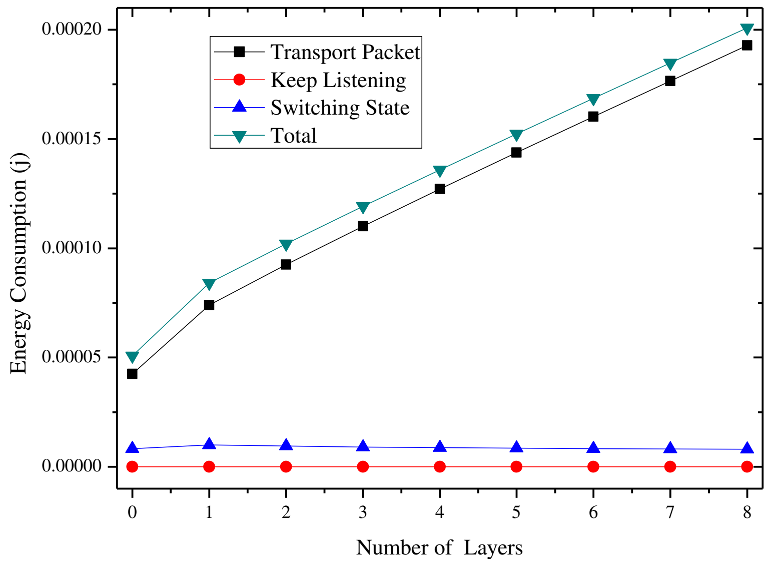

Fig. 16 Energy consumption without DEEDC scheme

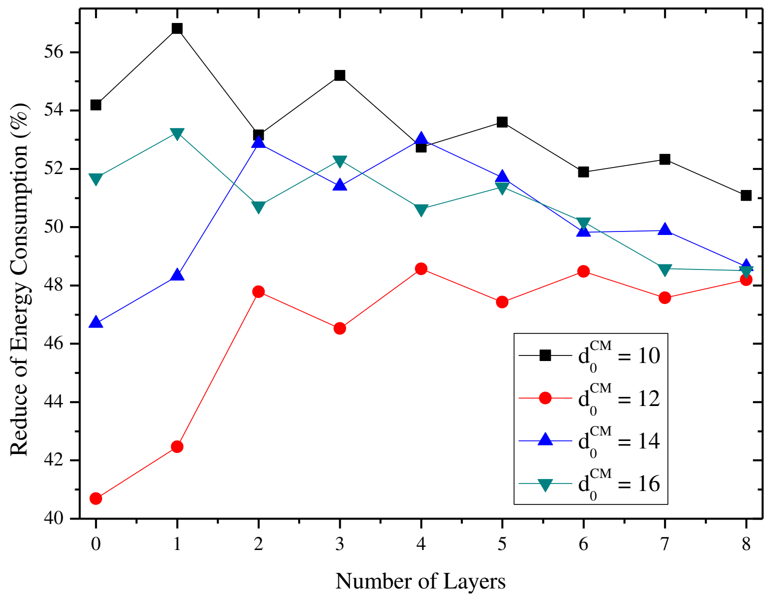

Fig. 17 The reduced percentage of the energy consumption (different d_0^{CM})

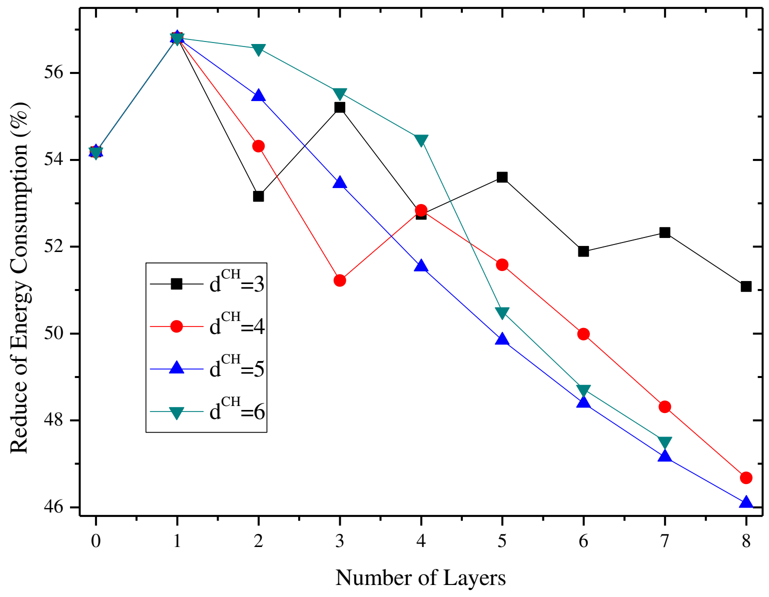

Fig. 18 The reduced percentage of the energy consumption (different d^{CH})

6 Conclusion

A delay and energy-efficient data collection scheme was proposed based on matrix filling theory for dynamic traffic WSNs in this paper. In this algorithm, our main work includes the following two points.

First, the original fixed transmission time slot is changed to on-demand allocation, thereby avoiding the fact that the corresponding node does not have a data packet transmission in some time slots, resulting in wasted time and energy. Under this mechanism, the cluster member node that wants to transfer a packet first transfers an RTS data packet to the CH node. The CH node arranges time slots for the corresponding CM nodes after receiving the RTS data packets, and transmits the time slot information by broadcasting. Thus, all nodes are able to complete the data transmission work in the allocated time slots. Compared to fixed time slots, this strategy reduces the delay that the CH node receives packets and the length of time the CH node remains awake. Secondly, through the support of matrix filling, the number of packets is reduced, and achieve the purpose of minimize both energy consumption and delay.

To sum up, a clustering routing algorithm suitable for networks that generate packets from time to time has successfully proposed, which reducing both energy consumption and delay. By introducing matrix filling technology, the delay is further reduced and the energy utilization rate is improved.

Abbreviations

CH: Cluster head; CM: Cluster member; DEEDC: Delay and energy-efficient data collection; IoT: Internet of Things; RTS: Request to send; TDMA: Time division multiple access; WSNs: Wireless sensor networks

Acknowledgements

The authors thank to the person who provided meticulous and valuable suggestions for improving the paper.

Authors' contributions

XX performed the experiment, analyzed the experiment results, and wrote the manuscript. WL, TW, XL, and HS comment on the manuscript. AL conceived of the work and wrote part of the manuscript. GZ comment on the manuscript. All authors read and approved the final manuscript.

Funding

This research was funded by the National Natural Science Foundation of China (61772554, 61572526, 61572528), and The Natural Science Foundation of Zhejiang Province (No. LY17F020032).

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]School of Computer Science, Central South University, Changsha 410083, China. [2]School of Informatics, Hunan University of Chinese Medicine, Changsha 410208, China. [3]Department of Computer Science and Technology, Huaqiao University, Xiamen 361021, China. [4]School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China. [5]School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China. [6]Department of Electrical, Computer, Software, and Systems Engineering, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114, USA. [7]Faculty of Informatics & Electronics, Zhejiang SCI-Tech University, Hangzhou 310018, China.

Received: 16 February 2019 Accepted: 7 June 2019
Published online: 24 June 2019

References

1. Z. Li, J. Gui, N. Xiong, Z. Zeng, Energy-efficient resource sharing scheme with out-band D2D relay-aided communications in C-RAN-based underlay cellular networks. IEEE Access **7**(1), 19125–19142 (2019)
2. H. Xiong, H. Zhang, J. Sun, Attribute-based privacy-preserving data sharing for dynamic groups in cloud computing. IEEE Syst. J. (2018). https://doi.org/10.1109/JSYST.2018.2865221
3. J. Ren, Y. Zhang, K. Zhang, A. Liu, J. Chen, X. Shen, Lifetime and energy hole evolution analysis in data-gathering wireless sensor networks. IEEE Trans. Ind. Inf. **12**(2), 788–800 (2016)
4. J. Park, S. Lee, S. Yoo, Time slot assignment for convergecast in wireless sensor networks. J. Parallel Distrib. Comput. **83**, 70–82 (2015)
5. Z. Li, Y. Liu, A. Liu, S. Wang, H. Liu, Minimizing convergecast time and energy consumption in green internet of things. IEEE Trans. Emerg. Top. Comput. (2018). https://doi.org/10.1109/TETC.2018.2844282
6. J. Wang, C. Jiang, L. Gao, S. Yu, Z. Han, Y. Ren, in *2016 IEEE International Conference on Communications (ICC)*. Complex network theoretical analysis on information dissemination over vehicular networks (2016), pp. 1–6
7. X. Ju, W. Liu, C. Zhang, A. Liu, T. Wang, N. Xiong, Z. Cai, An energy conserving and transmission radius adaptive scheme to optimize performance of energy harvesting sensor networks. Sensors **18**(9), 2885 (2018). https://doi.org/10.3390/s18092885
8. T. Li, K. Ota, T. Wang, X. Li, Z. Cai, A. Liu, Optimizing the coverage via the UAVs with lower costs for information-centric internet of things. IEEE Access **7**(1), 15292–15309 (2019)
9. X. Luo, C. Jiang, W. Wang, Y. Xu, J.H. Wang, W. Zhao, User behavior prediction in social networks using weighted extreme learning machine with distribution optimization. Futur. Gener. Comput. Syst. **93**, 1023–1035 (2019)
10. Y. Liu, A. Liu, N. Zhang, X. Liu, M. Ma, Y. Hu, DDC: dynamic duty cycle for improving delay and energy efficiency in wireless sensor networks. J. Netw. Comput. Appl. **131**, 16–27 (2019)
11. S. Zhang, X. Li, Z. Tan, T. Peng, G. Wang, A caching and spatial K-anonymity driven privacy enhancement scheme in continuous location-based services. Futur. Gener. Comput. Syst. **94**, 40–50 (2019)
12. J. Ren, Y. Zhang, N. Zhang, D. Zhang, X. Shen, Dynamic channel access to improve energy efficiency in cognitive radio sensor networks. IEEE Trans. Wirel. Commun. **15**(5), 3143–3156 (2016)
13. Y. Liu, M. Ma, X. Liu, N. Xiong, A. Liu, Y. Zhu, Design and analysis of probing route to defense sink-hole attacks for internet of things security. IEEE Trans. Netw. Sci. Eng. (2018) https://doi.org/10.1109/TNSE.2018.2881152
14. L. Hu, A. Liu, M. Xie, T. Wang, UAVs joint vehicles as data mules for fast codes dissemination for edge networking in Smart city. Peer-Peer Netw. Appl. (2019). https://doi.org/10.1007/s12083-019-00752-0
15. Z. Kuang, L. Li, J. Gao, L. Zhao, A. Liu, Partial offloading scheduling and power allocation for mobile edge computing systems. IEEE Internet Things J. (2019). https://doi.org/10.1109/JIOT.2019.2911455
16. W. Zhang, W. Liu, T. Wang, A. Liu, Z. Zeng, H. Song, S. Zhang, Adaption resizing communication buffer to maximize lifetime and reduce delay for WVSNs. IEEE Access **7**(1), 48266–48287 (2019)
17. X. Liu, Q. Yang, J. Luo, B. Ding, S. Zhang, An energy-aware offloading framework for edge-augmented mobile RFID systems. IEEE Internet Things J. (2018). https://doi.org/10.1109/JIOT.2018.2881295
18. L. Yin, J. Gui, Z. Zeng, Improving energy efficiency of multimedia content dissemination by adaptive clustering and D2D multicast. Mob. Inf. Syst. **2019**, 5298508 (2019). https://doi.org/10.1155/2019/5298508
19. Z. Gao, D. Wang, S. Wan, et al., Cognitive-inspired class-statistic matching with triple-constrain for camera free 3D object retrieval. Futur. Gener. Comput. Syst. **94**, 641–653 (2019)
20. C. Zhou, Y. Gu, S. He, Z. Shi, A robust and efficient algorithm for coprime array adaptive beamforming. IEEE Trans. Veh. Technol. **67**(2), 1099–1112 (2018)
21. T. Wang, J. Zhou, A. Liu, M. Bhuiyan, G. Wang, W. Jia, Fog-based computing and storage offloading for data synchronization in IoT. IEEE Internet Things J. (2018). https://doi.org/10.1109/JIOT.2018.2875915
22. Y. Liu, A. Liu, N. Xiong, T. Wang, W. Gui, Content propagation for content-centric networking from location-based social networks. IEEE Trans. Syst. Man Cybern. Syst. (2019). https://doi.org/10.1109/TSMC.2019.2898982
23. H. Teng, Y. Liu, A. Liu, N. Xiong, Z. Cai, T. Wang, X. Liu, A novel code data dissemination scheme for internet of things through mobile vehicle of smart cities. Futur. Gener. Comput. Syst. **94**, 351–367 (2019)
24. H. Zhang, Y. Qi, H. Zhou, J. Zhang, J. Sun, Testing and defending methods against DoS attack in state estimation. Asian J. Control **19**(3), 1–11 (2017)
25. X. Liu, Y. Liu, N. Zhang, W. Wu, A. Liu, Optimizing trajectory of unmanned aerial vehicles for efficient data acquisition: a matrix completion approach. IEEE Internet Things J. (2019). https://doi.org/10.1109/JIOT.2019.2894257
26. L. Zhou, X. Li, K. Yeh, C. Su, W. Chiu, Lightweight IoT-based authentication scheme in cloud computing circumstance. Futur. Gener. Comput. Syst. **91**, 244–251 (2019)
27. X. Liu, S. Wu, X. Xu, J. Jiao, Q. Zhang, Improved polar SCL decoding by exploiting the error correction capability of CRC. IEEE Access **7**, 7032–7040 (2019)
28. T. Wang, G. Zhang, A. Liu, M. Bhuiyan, Q. Jin, A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing. IEEE Internet Things J. (2018). https://doi.org/10.1109/JIOT.2018.2870288
29. Z. Zhu, J. Peng, X. Gu, H. Li, K. Liu, Z. Zhou, W. Liu, Fair resource allocation for system throughput maximization in mobile edge computing. IEEE Access **6**, 5332–5340 (2018)
30. Q. Liu, Y. Guo, J. Wu, G. Wang, Effective query grouping strategy in clouds. J. Comput. Sci. Technol. **32**(6), 1231–1249 (2017)
31. T. Wang, G. Zhang, M. Bhuiyan, A. Liu, W. Jia, M. Xie, A novel trust mechanism based on fog computing in sensor-cloud system. Futur. Gener. Comput. Syst. (2018) https://doi.org/10.1016/j.future.2018.05.049
32. M. Luo, K. Wang, Z. Cai, A. Liu, Y. Li, C. Cheang, Using imbalanced triangle synthetic data for machine learning anomaly detection. Comput. Mater. Continua **58**(1), 15–26 (2019)
33. S. Wan, Y. Zhao, T. Wang, et al., Multi-dimensional data indexing and range query processing via Voronoi diagram for internet of things. Futur. Gener. Comput. Syst. **91**, 382–391 (2019)
34. Q. Li, A. Liu, T. Wang, M. Xie, N. Xiong, Pipeline slot based fast rerouting scheme for delay optimization in duty cycle based M2M communications. Peer-Peer Netw. Appl. (2019). https://doi.org/10.1007/s12083-019-00753-z
35. C. Zhang, Y. Lin, L. Zhu, A. Liu, Z. Zhang, F. Huang, CNN-VWII: An efficient approach for large-scale video retrieval by image queries. Pattern Recogn. Lett. **123**, 82–88 (2019)
36. X. Luo, Y. Lv, M. Zhou, W. Wang, W. Zhao, A laguerre neural network-based ADP learning scheme with its application to tracking control in the internet of things. Pers. Ubiquit. Comput. **20**(3), 361–372 (2016)
37. X. Luo, D. Zhang, T. Yang, J. Liu, X. Chang, H. Ning, A kernel machine-based secure data sensing and fusion scheme in wireless sensor networks for the cyber-physical systems. Futur. Gener. Comput. Syst. **61**, 85–96 (2016)
38. Wang Q, Liu W, Wang T, Zhao M, Li X, Xie M, Ma M, Zhang G, Liu A. Reducing Delay and Maximizing Lifetime for Wireless Sensor Networks with Dynamic Traffic Patterns. IEEE Access. **7**(1), 70212–70236 (2019).
39. J. Chen, K. Hu, Q. Wang, Y. Sun, Z. Shi, S. He, Narrow-band internet of things: implementations and applications. IEEE Internet Things J. **4**(6), 2309–2314 (2017)
40. Q. Deng, H. Zeng, J. Zhang, S. Tian, J. Cao, Z. Li, A. Liu, Compressed sensing for image reconstruction via back-off and rectification of greedy algorithm. Signal Process. **157**, 280–287 (2019)
41. S. Ding, S. Qu, Y. Xi, et al., A long video caption generation algorithm for big video data retrieval. Futur. Gener. Comput. Syst. **93**, 583–595 (2019)
42. Q. Liu, G. Wang, X. Liu, T. Peng, J. Wu, Achieving reliable and secure services in cloud computing environments. Comput. Electr. Eng. **59**, 153–164 (2017)
43. W. Yang, W. Liu, Z. Zeng, A. Liu, G. Huang, N. Xiong, Z. Cai. Adding active slot joint larger broadcast radius for fast code dissemination in WSNs. Sensors, 18(11), 4055 (2018). DOT: https://doi.org/10.3390/s18114055
44. W. Qi, W. Liu, X. Liu, A. Liu, T. Wang, N. Xiong, Z. Cai, Minimizing delay and transmission times with long lifetime in code dissemination scheme for high loss ratio and low duty cycle WSNs. Sensors **18**(10), 3516 (2018) https://doi.org/10.3390/s18103516
45. Y. Ren, W. Liu, T. Wang, X. Li, N. Xiong, A. Liu, A collaboration platform for effective task and data reporter selection in crowdsourcing network. IEEE Access **7**(1), 19238–19257 (2019)
46. J. Tan, W. Liu, T. Wang, N. Xiong, H. Song, A. Liu, Z. Zeng, An adaptive collection scheme based matrix completion for data gathering in energy-harvesting wireless sensor network. IEEE Access **7**(1), 6703–6723 (2019)
47. Y. Liu, A. Liu, X. Liu, X. Huang, A statistical approach to participant selection in location-based social networks for offline event marketing. Inf. Sci. **480**, 90–108 (2019)

48. X. Xiang, W. Liu, N. Xiong, H. Song, A. Liu, T. Wang, Duty cycle adaptive adjustment based device to device (D2D) communication scheme for WSNs. IEEE Access **6**(1), 76339–76373 (2018)
49. B. Huang, W. Liu, T. Wang, X. Li, H. Song, A. Liu, Deployment optimization for data centers in vehicular networks. IEEE Access **7**(1), 20644–20663 (2019)
50. A. Liu, J. Min, K. Ota, M. Zhao, Reliable differentiated services optimization for network coding cooperative communication system. Int. J. Comput. Syst. Sci. Eng. **33**(4), 235–250 (2018)
51. J. Gui, Z. Li, Z. Zeng, Improving energy-efficiency for resource allocation by relay-aided in-band D2D communications in C-RAN-based systems. IEEE Access **7**(1), 8358–8375 (2019)
52. W. Liu, P. Zhuang, h. Liang, J. Peng, Z. Huang, Distributed economic dispatch in microgrids based on cooperative reinforcement learning. IEEE Trans. Neural Netw. Learn. **29**(6), 2192–2203 (2018)
53. L. CARR-MOTYČKOVÁ, D. Dryml, *Convergecast tree construction in wireless sensor networks. Adhoc & Sensor Wireless Networks*, vol 27 (2015), pp. 3–4, 263-293
54. B. Malhotra, I. Nikolaidis, M. Nascimento, Aggregation convergecast scheduling in wireless sensor networks. Wirel. Netw **17**(2), 319–335 (2011)
55. S. Gandham, Y. Zhang, Q. Huang, Distributed time-optimal scheduling for convergecast in wireless sensor networks. Comput. Netw. **52**(3), 610–629 (2008)
56. X. Xu, X. Li, X. Mao, S. Tang, S. Wang, A delay-efficient algorithm for data aggregation in multihop wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. **22**(1), 163–175 (2011)
57. X. Xiang, W. Liu, A. Liu, N. Xiong, Z. Zeng, Z. Cai, Adaptive duty cycle control based opportunistic routing scheme to reduce delay in cyber physical systems. Int. J. Distrib. Sens. Netw.. https://doi.org/10.1177/1550147719841870 (2019)
58. H. Teng, W. Liu, T. Wang, A. Liu, X. Liu, S. Zhang, A cost-efficient greedy code dissemination scheme through vehicle to sensing devices (V2SD) communication in Smart city. IEEE Access **7**(1), 16675–16694 (2019)
59. X. Wu, Y. Li, Y. Li, W. Lou, Energy-efficient wake-up scheduling for data collection and aggregation. IEEE Trans. Parallel Distrib. Syst. **21**(2), 275–287 (2010)
60. J. Li, W. Liu, T. Wang, H. Song, X. Li, F. Liu, A. Liu, Battery-friendly based relay selection scheme to prolong lifetime for sensor nodes in internet of things. IEEE Access **7**(1), 33180–33201 (2019)
61. M. Wu, Y. Wu, C. Liu, Z. Cai, N. Xiong, A. Liu, M. Ma, An effective delay reduction approach through portion of nodes with larger duty cycle for industrial WSNs. Sensors **18**, 1535 (2018)
62. Y. Liu, A. Liu, Z. Chen, Analysis and improvement of send-and-wait automatic repeat-request protocols for wireless sensor networks. Wirel. Pers. Commun. **81**, 923–959 (2015)
63. Y. Ren, Y. Liu, N. Zhang, A. Liu, N. Xiong, Z. Cai, Minimum-cost mobile crowdsourcing with QoS guarantee using matrix completion technique. Pervasive Mob. Comput. **49**, 23–44 (2018)
64. Y. Liu, A. Liu, Y. Hu, Z. Li, Y. Choi, H. Sekiya, J. Li, FFSC: An energy efficiency communications approach for delay minimizing in internet of things. IEEE Access **4**(3775–3793) (2016)
65. R. Dhanalakshmi, A. Vadivel, P. Parthiban, Shortest path routing in solar powered WSNs using soft computing techniques. J. Sci. Ind. Res. **76**, 23–27 (2017)
66. U. Kim, E. Kong, H. Choi, J. Lee, Analysis of aggregation delay for multisource sensor data with on-off traffic pattern in wireless body area networks. Sensors **16**, 1622 (2016)
67. M. Huang, W. Liu, T. Wang, H. Song, X. Li, A. Liu, A queuing delay utilization scheme for on-path service aggregation in services oriented computing networks. IEEE Access **7**(1), 23816–23833 (2019)
68. X. Xu, M. Yuan, X. Liu, A. Liu, N. Xiong, Z. Cai, T. Wang, Cross-layer optimized opportunistic routing scheme for loss-and-delay sensitive WSNs. Sensors **18**, 1422 (2018)
69. E. Candès, B. Recht, Exact matrix completion via convex optimization. Found. Comput. Math. **9**(6), 717–772 (2009)

7 Publisher's Note