Distributed Coding Schemes for Continuous Data Collection in Wireless Sensor Networks

March 2014

Xiucai Ye

Distributed Coding Schemes for Continuous Data Collection in Wireless Sensor Networks

Graduate School of Systems and Information Engineering

University of Tsukuba

March 2014

Xiucai Ye

Abstract

Wireless Sensor Networks (WSNs) have become one of the most interesting research areas in recent years due to a wide range of potential applications such as environment monitoring and battlefield surveillance. An important problem that arises in application of WSNs is how to collect data continuously, especially in extreme environments. Consider that data are continuously sensed and collected by the sensor nodes in the extreme environments. Data collection is only performed from time to time by a mobile Base Station (mBS). Sensor nodes have to store the continuously collected data segments over time by themselves, and provide the desired data when the mBS arrives and performs data collection. Such kind of data collection is known as continuous data collection. This dissertation addresses the continuous data collection in WSNs with a mobile Base Station (mBS). We propose continuous data collection schemes based on distributed coding methods, with the goal to achieve a high success ratio of data collection and reduce the energy consumption.

We consider two scenarios of continuous data collection. The first scenario is latest data segment collection, which is to collect the m latest data segments, where m is the number of latest data segments in a time interval t in which $n(t)$ $(m \leq n(t))$ data segments are generated. m is a fixed number. No matter how many data segments are generated in a time interval, the required data are the m latest data segments. The second scenario is all data segment collection, which is to collect all the $n(t)$ data segments generated in a time interval t. $n(t)$ is a variable, the value depends on when the mBS performs data collection. $n(t)$ increases as t increases. We propose two Distributed Separate Coding schemes for the two scenarios, respectively.

For the first scenario of continuous data collection, we propose the Distributed Separate Coding scheme for m Latest Data segment Collection (DSC-mLDC) in wireless sensor networks with a mobile Base Station (mBS). By separately encoding a certain number of data segments in a combined segment, and doing decoding-free data replacement in the buffers of each sensor node, the proposed DSC-mLDC scheme is shown as an efficient method for continuously collecting data segments with a high success ratio. The proposed DSC-mLDC scheme has the salient feature: with a minimum buffer size 2 in each sensor node, by querying any $m - 1$ sensor nodes, the mBS can reconstruct the m latest data segments with high probability. The necessary storage space in each sensor node can be adjusted by changing the number of sensor nodes queried by the mBS. Furthermore, the transmission cost for data submission to the mBS can be reduced with a few additional storage space in each sensor node. The comprehensive performance evaluation has been conducted through computer simulation. It is shown that the proposed DSC-mLDC scheme outperforms the existing schemes significantly.

For the second scenario of continuous data collection, we propose the Distributed Separate Coding scheme for All Data segment Collection (DSC-ADC) in wireless sensor networks with a mobile Base Station (mBS). By separately encoding a certain number of data segments in a combined segment, and storing the combined segments in the corresponding buffers of each sensor node, the proposed DSC-ADC scheme provides an efficient storage method to collect all data segments. By randomly querying a small subset of sensor nodes, the mBS can reconstruct all the original data segments with high probability in both the right arrival case and the late arrival case. The number of sensor nodes that should be queried by the mBS can be reduced with a few additional storage space in each sensor node. The performance evaluation has been conducted through computer simulations. It further demonstrates the feasibility and superiority of the proposed DSC-ADC scheme.

Acknowledgements

This dissertation would not have been possible without the help, support and patience of my supervisor, Prof. Jie Li. I would like to express my deepest gratitude to Prof. Li, who gave me the chance to be a member of the Operating System and Distributed Processing (OSDP) lab at University of Tsukuba. His outstanding supervision and guidance play a critical role in my research or even in my life. There is no doubt in my mind that my own research and writing style is greatly influenced by how Prof. Li conducts his research. I hope that I can continue to live up to his high standards.

It is my great honor to thank Prof. Jiro Tanaka, Prof. Hiroyuki Kitagawa, Prof. Kazuki Katagishi, and Prof. Shigetomo Kimura for being my dissertation committee members and providing valuable advices and comments on evaluating this dissertation in its final form. I would like to thank the Department of Computer Science, the Graduate School of Systems and Information Engineering at University of Tsukuba, for their continuous supports on my study in Japan.

I express the gratitude to my supervisor during my MS program study, Prof. Li Xu for his supervision, guidance, and valuable help, without which it would have been difficult for me to have an opportunity to pursue the PhD degree in Japan.

I would like to thank all the members in the OSDP lab for their kind help: Dr. Ghada Ahmed Abdel Monaim Khoriba, Dr. Yongsheng Liu, Dr. Biao Han Dr. Xiaoyan Wang, Huang Lu, Shuai Fu, Li Qiang, Kei Ebana, Yujie Hu, Zichen Jin, Zhengxu Li, Cheng Sun, Ke Tang, Serigne Mbacke Ndiaye, Yuehong Liu, Jiahan Chen, Lei Chen, Dan Zhang, Yulin Shi, Qichao Li, Dr. Wei Li, Dr. Xiaofei Xing, Dr. Yu Gu, Liwen Xu, Atsushi Nagashima, Zhongping Dong, Ou Wang, Yajun Tian and Naoto Ishitsuka. It has been my great honor to work with them. Also, it is my pleasure to thank Prof. Hisao Kameda (University of Tsukuba), Prof. Mohsen Guizani (Qatar University, Qatar), Prof. Jiannong Cao (Hong Kong Polytechnic University), Prof. Lasheng Yu (Central South University, China), Prof. Jianping

Pan (University of Victoria, Canada), and Prof. Wenzhong Guo (Fuzhou University, China).

I am deeply grateful to all my family, especially my parents, my husband, and my beloved daughter. They continually support me without any reservation. I am also grateful for all my friends in Japan.

My apologies if I have inadvertently omitted anyone to whom an acknowledgment is due.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) have become one of the most interesting research areas in recent years due to a wide range of potential applications such as environment monitoring and battlefield surveillance. An important problem that arises in application of WSNs is how to collect data continuously, especially in extreme environments. Due to limited energy and hostile environment, sensor nodes are vulnerable. Thus, redundant data storage are required to improve the reliability of the sensor nodes. Coding is a powerful method to achieve efficient management of redundant data storage. We consider to redundantly store the data in the sensor nodes by using coding method, to improve the probability of successfully collecting the required data and reduce the energy consumption in the sensor nodes.

This chapter is organized as follows. An overview of the wireless sensor networks and sensor data collection are presented in Section 1.1 and Section 1.2, respectively. The problem description of continuous data collection and its existing challenges in WSNs are presented in Section 1.3. We introduce coding for data storage in Section 1.4. The related work are presented in Section 1.5. The dissertation objectives and motivations are presented in Section 1.6. Section 1.7 summarizes the contributions of the dissertation. Section 1.8 illustrates the organization of the dissertation.

1.1 Wireless Sensor Networks (WSNs)

Recent advances in micro-electro-mechanical systems (MEMS) technology, wireless communications, and digital electronics have led to the emergence of *Wireless Sensor Networks* (WSNs). WSNs are composed of a large number of sensor nodes to monitor physical or environmental conditions, and are connected to the outside world via more powerful nodes called base stations (also called as sink nodes). The base station collects the data from the sensor nodes and sends the collected data to the users through an Internet connection, as shown in Fig 1.1. Note that a base station can be a fixed or mobile one. Recent advances of embedded hardware and robot have made mobile base station possible in WSNs [1–3].

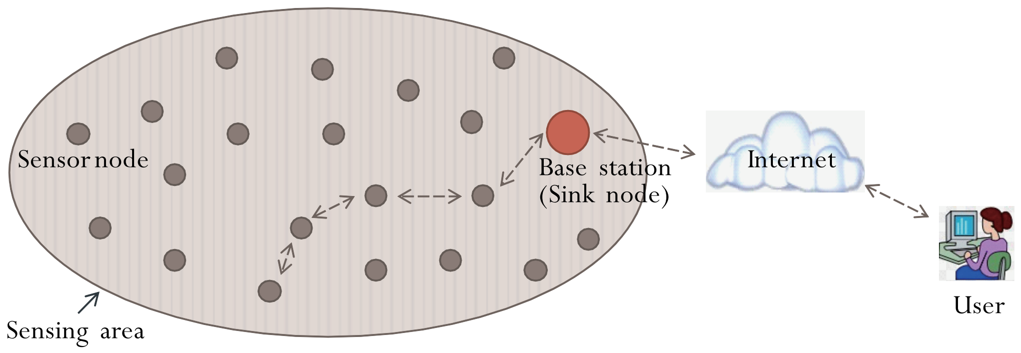

FIGURE 1.1: Architecture of a wireless sensor network.

The sensor nodes consist of sensing, data processing, and communicating capabilities. A sensor node is mainly equipped with four components [4]: a sensing unit, a transceiver unit, a processing unit, and a power unit, as shown in Fig 1.2. It may also has additional components which are application dependent, such as a location unit, a mobilizer unit, and a power generator unit. The sensing unit generates the sensed data based on the observed phenomenon. The sensed data are converted to digital signals, and then fed into the processing unit. The processing unit is generally associated with a small storage unit to manage the procedures. The procedures make the sensor node collaborate with the other sensor nodes to carry out the assigned tasks.

The transceiver unit connects the sensor node to the other sensor nodes, e.g., sending and receiving data. The power unit is one of the most important components of a sensor node, which may be supported by a power scavenging unit such as solar cells.

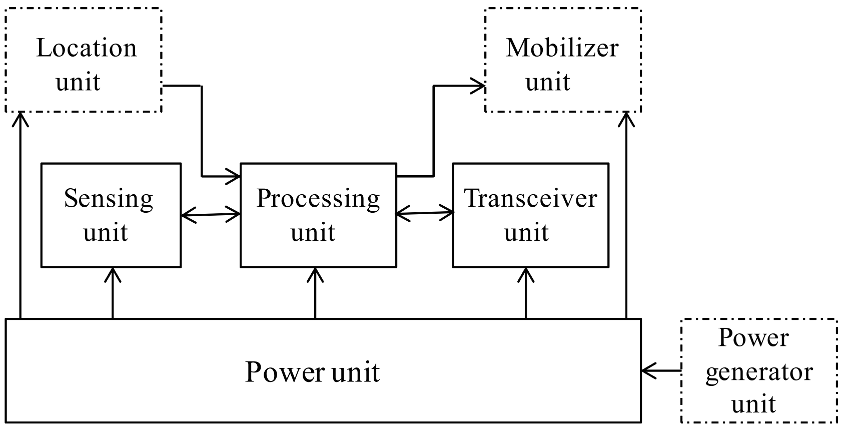

FIGURE 1.2: Components of a sensor node.

WSNs may consist of many different types of sensors to monitor a wide variety of ambient conditions, such as temperature, humidity, vehicular movement, lightning condition, pressure, soil makeup and noise levels [5–7]. Sensor nodes can be used for continuous sensing, event detection, location sensing, and local control of actuators [8]. The concept of micro-sensing and wireless connection of these nodes promise many new application areas. The rapid deployment, self-organization, micro-sensing and wireless connection characteristics of sensor networks make them a wide range of applications, including environment monitoring, medical care, battlefield surveillance, industrial diagnostics, smart spaces and biological detection [9–12].

WSNs represent a significant improvement over traditional networks. The deployment of WSNs require wireless ad hoc networking techniques, which do not rely on any pre-deployed network infrastructure. The position of sensor nodes usually are not engineered or pre-determined. The sensor nodes can be deployed randomly in inaccessible terrains or disaster relief operations [4]. The main difference between the WSNs and the traditional networks is that sensor nodes are resource constrained.

A sensor node has low CPU power, small bandwidth, limited battery and limited memory storage [4, 13]. For example, TelosB sensor nodes have a 16-bit 8 MHz microcontroller with 10 KB Random Access Memory (RAM), 48 KB program flash memory, and 1024 KB measurement serial flash [14]. The total storage energy in a smart dust sensor node is on the order of 1 J [15]. As a result, a sensor node can store only a small amount of data collected from its surroundings and carry out a limited number of computations. In addition, the sensor nodes with limited resources are prone to failures. Thus, sensor nodes are less reliable both in survivability and data transmission. To overcome these shortcomings, a single sensor node is usually expected to cooperate with other sensor nodes. The sensor nodes are redundantly and densely deployed, they may not have global identification (ID) because of the large amount of overhead and large number of sensor nodes.

The design of WSNs are influenced by many factors, which include power conservation; scalability; sensor network topology; hardware constraints; transmission media; production costs; operating environment and so on [4]. Since it is difficult and cost ineffective to recharge the sensor batteries, power conservation is the main focus of the current sensor network design. Data transmission is the main energy consumption in WSNs. The energy consumption of data transmission is affected by the payload and the transmission distance [16, 17]. Some of the measuring methods for energy consumption of data transmission have been introduced in [18, 19].

Based on the unique features and capabilities of WSNs, a large number of research activities have been undertaken in WSNs [20]. In this thesis, we are interested in the problems of continuous data collection in WSNs, with the objective to improve the success ratio of data collection and reduce the energy consumption in the sensor nodes.

1.2 Sensor Data Collection

One of the most important applications in WSNs is data collection, where the sensed data are collected at all or some from the sensor nodes and forwarded to a base station for further processing [21, 22]. A wide range of deployments for data collection in WSNs have been witnessed in real-world in the past few years, including environmental research [23, 24], wildlife habitat monitoring [25], water monitoring [26], volcano monitoring [27, 28], wildland fire forecast/detection [29] and civil engineering [30, 31], to name but a few.

Although many research efforts have been done on WSNs, sensor data collection in WSNs with special features are different from other applications in WSNs. The processing of data collection is much more complex than that in other applications of WSNs, such as target tracking [32]. In target tracking, the sensed data are locally processed and stored at some nodes and may be queried later by some other nodes [33]. The major traffic in sensor data collection is the sensed data from each sensor node to the base station, which may cause high unbalanced and inefficient energy consumption in the whole network. For example, the sensor nodes close to the base station are depleted quickly due to traffic relays, which will cause network disconnected from the base station.

1.2.1 Taxonomy

The data collection in WSNs can be divided into three major stages: the deployment stage, the control message dissemination stage and the data delivery stage [22], as shown in Fig 1.3. Each of the three stages has its own issues and focuses. The deployment stage addresses the issues of how to deploy the WSNs in the sensing field. The deployment of WSNs can be further classified into the area-coverage deployment and the location-coverage deployment based on the application requirement.

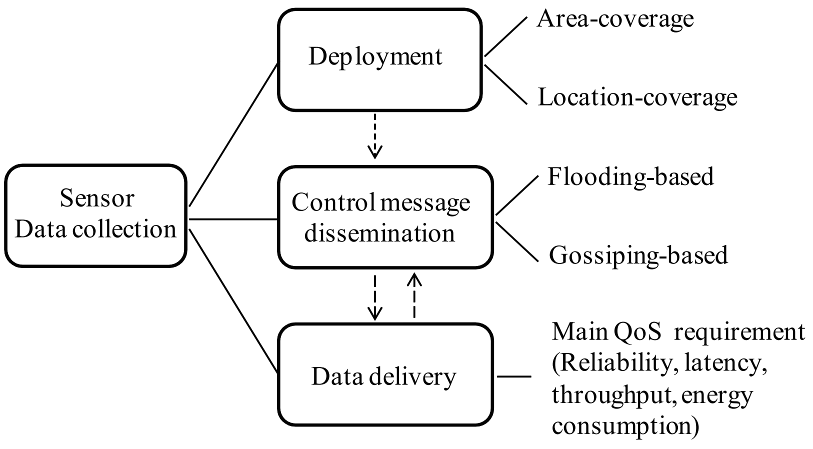
Deployment
Area-coverage
Location-coverage
Sensor Data collection
Control message dissemination
Flooding-based
Gossiping-based
Data delivery
Main QoS requirement (Reliability, latency, throughput, energy consumption)

FIGURE 1.3: Major stages of using wireless sensor networks for data collection.

The area-coverage deployment requires each location within the sensing field must be covered by some sensor nodes and the location-coverage deployment requires the sensor nodes must be attached to some locations specified by the applications. The control message dissemination stage addresses the issues of how to disseminate the control messages (e.g., network management or collection command messages) from the base station to the sensor nodes. The challenges in this stage is how to disseminate messages to the sensor nodes with small transmission costs and low latencies. Flooding and gossiping are the two wildly used dissemination methods in WSNs. Many works have enhanced the two basic dissemination methods to improved the network performance efficiency [34–36]. The data delivery stage is the main task of data collection in WSNs. The sensed data are gathered at different sensor nodes and delivered to the base station based on the control message in stage 2. Different QoS requirements (e.g., reliability, throughput, latency and energy consumption) from the applications will lead to different approach designs.

Note that the issues we address in this thesis belong to the data delivery stage. We will show some data delivery approaches based on different QoS requirements in WSNs.

1.2.2 Data delivery approaches

In the data delivery stage, different approaches based on different QoS requirements in WSNs have been proposed to deliver sensed data from sensor nodes to the base station.

A data delivery approach to improve the network reliability were proposed in [37] by using both hop-by-hop and end-to-end recoveries. Specifically, each sensor node keeps a missing list to record the sequence number of the missing packet. The sensor node that previously relayed the missing packet will then schedule a retransmission. If a sensor node finds some missing packets of other sensor node sharing the same sources with those packets in its own packet cache, it adds these packets into its own missing list. Thus, the missing packet information will trace back hop-by-hop until reaching the source nodes. The source nodes will re-send the packets and finish the circle of end-to-end recoveries.

Lu et al. [38] proposed an energy efficient and low latency scheme to reduce energy consumption and latency in sensor data collection. The energy efficient and low latency scheme is designed to solve the interruption problem and allow continuous packet forwarding by giving the sleep schedule of a node. The duty cycles adaptively are adjusted according to the traffic load in the network. Furthermore, a data prediction mechanism was proposed to alleviate problems pertaining to channel contention and collisions. Another scheme which also targets on minimizing latency and reducing energy consumption was proposed in [39], in which time slot is defined to be the duration for successfully transmitting a maximum transmission unit. Paradis et al. [40] proposed a scheme called TIGRA to reduce the latency by batching small sensed data from different sensor nodes into packets.

There are also a series of prior work focused on energy efficiency in sensor data collection, such as the ultra-low power data delivery scheme [41] and the Time Synchronized Mesh Protocol (TSMP) [42]. The related work of data delivery schemes to improve the throughput of the network can be found in [43, 44].

In this thesis, we consider to improve the success ratio of data collection, so as to improve the reliability and reduce the energy consumption and latency.

1.3 Continuous Data Collection in WSNs

An important problem that arises in sensor data collection is how to collect data continuously, especially in extreme environments [45–47]. In some extreme environments such as Greenland or Alaska, it is difficult to travel and dangerous to work for humans [48, 49]. Instrumenting the environments with WSNs can enable long-term data collection, which could minimize the exposure of humans while allowing dense, targeted data collection to commence [48, 50].

Consider that data are continuously sensed and collected by the sensor nodes in the extreme environments. The communication between the sensor nodes and a mobile Base Station (mBS) is scarce. Data collection is only performed from time to time by a mBS. Sensor nodes have to store the continuously collected data segments over time by themselves, and provide the desired data when the mBS arrives and performs data collection. Such kind of data collection is known as *continuous data collection in WSNs with a mBS* [51]. One of the typical examples is the habitat monitoring system in Great Duck Island [52], in which data collection is performed from time to time since seabird colonies are sensitive to human interaction. An efficient data retrieval is usually desired during the data collection. Applications of monitoring systems in chemical plants also have the similar properties, in which technicians occasionally approach the sensing area to collect data and each data collection should be performed quickly for safety purposed [51].

The main challenge in continuous data collection is how to achieve high reliability of data collection. The reasons that continuous data collection in WSNs may have low reliability can be listed as follows.

- **Vulnerability of sensor nodes** : In the extreme environments, sensor nodes may fail suddenly due to limited energy and hostile environment. As a result, the sensed data in the failed sensor nodes will be lost.
- **Limited storage space in sensor nodes** : In the extreme environments, data collection is only performed from time to time by a mBS. Sensor nodes have to store the continuously collected data segments over time by themselves. The sensed data may exceed the storage space of the sensor nodes. The overflowed data in the sensor nodes will be lost.
- **Low success ratio of data collection** : The mobile Base Station (mBS) can not collect the desired data with high probability due to the data loss based on the above two reasons.

To achieve high reliability of continuous data collection, it is desirable to redundantly store data in sensor nodes. Many data collection schemes as we mentioned in the above section are not suitable for continuous data collection. In these schemes, the interested data are only stored in some source sensor nodes without coding. The base station sends out a query to ask for the data from the sensor nodes of interest. The desired data are then routed from the source sensor nodes to the base station [53]. Since the source sensor nodes may be scattered randomly in a sensor network, these schemes may also introduce a long delay due to data searching from the source sensor nodes [54–56].

To improve data collection reliability, a random data collection scheme is proposed (e.g., the data collection schemes in [51, 57, 58]), in which data are redundantly stored in the sensor nodes. In this scheme, the mBS queries a small subset of sensor nodes uniformly at random from the sensor network to retrieve data. The mBS can retrieve the sensed data from any subset of sensor nodes, even after some sensor nodes have failed. The introduction of data redundancy is to ensure that the whole network is acting as a robust distributed storage database [59]. However, the redundance of data storage with straightforward method may introduce large replication of data in a sensor network.

1.4 Coding for Data Storage

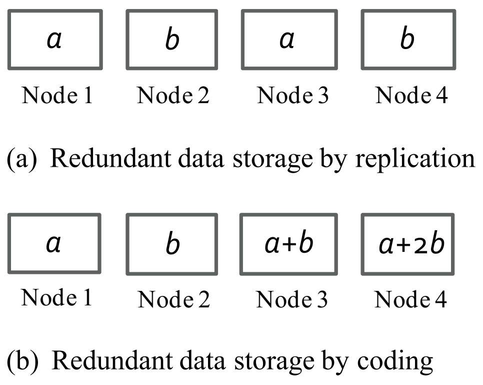
a
b
a
b
Node 1
Node 2
Node 3
Node 4
(a) Redundant data storage by replication
a
b
a+b
a+2b
Node 1
Node 2
Node 3
Node 4
(b) Redundant data storage by coding

FIGURE 1.4: Two methods for redundant data storage in four sensor nodes (Node 1 through Node 4).

Coding is a powerful method for data storage, which can achieve efficient management of redundant data storage [60]. We first take an example to show the advantage of coding method for redundant data storage. Consider that a and b are the data segments stored in node 1 and node 2, respectively. If node 1 fails, the data a is lost. Fig 1.4 (a) shows the replication method for redundant data storage. a and b are replicated in node 3 and node 4, respectively. By replication, if node 1 fails, a is still preserved in node 3. However, if node 3 also fails, a is lost. Then, we show the coding method for redundant data storage. As shown in Fig 1.4 (b), a and b are both encoded in node 3 and node 4. If node 1 and node 3 fail, a can be preserved by decoding the data in node 3 and node 4. Thus, coding is more reliable than replication. Note that by coding, a and b can be decoded from any two nodes out of the four nodes.

When using coding for distributed storage in WSNs, the data should be disseminated from a sensor node to other sensor nodes for encoding. As the example in Fig 1.4 (b), the data a and b should be disseminated from node 1 and node 2 to node 3 and node 4, respectively. After data dissemination, each sensor node performs data encoding on the received data. The base station randomly queries a small subset of sensor nodes to collect data and performs data decoding on the collected data. Many existing work have addressed the problem of coding for distributed storage in WSNs. Among these work, many schemes considered that a small part of sensor nodes function as the source nodes to sense data, and the other sensor nodes function as the storage node to do encoding and storage [57–59, 61]. Vukobratovic et al. [62] proposed a coding based packet centric scheme, in which each sensor node senses data and functions as both the source node and storage node. The source nodes disseminate the sensed data to the storage nodes for encoding. Also, some related work only focus on the data encoding and decoding processes without addressing the data dissemination process. For example, in [51] the authors considered that the data for encoding in each sensor node are the same based on the existing data dissemination methods.

This dissertation focuses on the data encoding process in the sensor nodes and the decoding process in the mBS. We do not address the data dissemination process. Similar to that considered in [51], we consider that the data for encoding in each sensor node are the same based on the existing data dissemination methods.

For data coding, many coding methods have been proposed for data storage. We give a brief survey of erasure coding, network coding and Fountain coding, which are wildly used in WSNs.

1.4.1 Erasure coding

Erasure coding is a linear combinations of original data segments into encoded data segments of the same size. Erasure coding can achieve much higher reliability compared to replication schemes for the same number of storage nodes. The Reed-Solomon coding is a well-known erasure coding scheme, which is widely employed in a computer network with distributed storage systems and redundant disk arrays [63, 64].

Most of the erasure coding schemes proposed in the existing work are linear codes over finite fields, such as random linear coding. In random linear coding, a related coefficient for each data segment is randomly generated from a finite filed. The encoded data segment is the combination of the original data segments multiplied with their related coefficients [65].

The original data segments in erasure codes can be decoded from any subset of the encoded data segments whose size is equal to the number of original data segments. The original data segments can be decoded by using Gaussian elimination in $O(k^3)$, where k is the number of original data segments combined in an encoded data segment. By using the sparsity of the linear equations, faster decoding of erasure codes can be achieved. For example, using the Wiedemann algorithm [66], the erasure codes can be decoded in $O(k^2 \log(k))$ time on average.

1.4.2 Network coding

Network Coding is an emerging technique that has several interesting applications in practical networking systems. By network coding, nodes can combine several input packets into one or several output packets instead of simply forwarding data [67]. Network coding was first introduced for improving the performance of multicast routing, that significantly improve the throughput of network [68]. It was also introduced to wireless applications [69, 70], where the shared nature of the wireless medium in wireless network proposes a natural opportunity for network coding. Network coding can also be employed as a solution to increase robustness and reliability of data transmission [71, 72], protecting against link or node failures in wireless communication networks.

Recently, the idea of network coding is extent for the applications of data storage and distribution [73], where the code is created over the connecting of data and storage nodes. The further theoretically study about network coding for data distribution and the practical system for random file distribution are presented in [65] and [74]. Using network coding for ubiquitous data collection was introduced in [75–77] for wireless sensor networks.

1.4.3 Fountain coding

Fountain coding is to generate linear codes over F_2 with low encoding and decoding complexities [78–80]. Fountain codes are rateless in the sense that the encoding process can generate unlimited number of encoded packets. The encoded packet is generated by the exclusive-or (XOR) of a subset of source packets. The encoding process can be performed online. The original data segments can be decoded from any subset of the encoded data segments whose size is equal to or only slightly greater than the number of original data segments.

Luby transform codes (LT codes) [81] are the first realization of Fountain codes, which make Fountain codes work in practice. In LT codes, each encoded data segment is created by first selecting a degree d from a carefully designed degree distribution (called the robust soliton [81]), and then taking the bitwise XOR of d randomly selected data segments. The decoding process is performed by using the Belief Propagation algorithm [82], which is more computationally efficient than the general matrix inversion process (i.e., Gaussian Elimination).

However, since the number of data segments to do encoding in each sensor node should follow some probability distribution (e.g., the robust soliton distribution in LT codes), the implement of Fountain codes in WSNs is more difficult than that of random linear codes.

1.5 Related Work

Many data storage schemes using coding methods in a centralized way are proposed. A typical coding scheme is the erasure coding [83]. Weatherspoon et al. [84] compared replication with erasure coding in the bandwidth-reliability tradeoff space. The analysis showed that erasure coding reduced bandwidth use by an order of magnitude compared with replication. Bhagwan et al. [85] obtained a similar conclusion by simulations in the Total Recall storage system. Rabin et al. [86] proposed a canonical coding scheme which was based on the optimal block erasure Reed-Solomon codes [87]. Byers et al. [88] optimized large transfers by using codes based on the efficient estimation, summarization, and approximate reconciliation of the sets of data between pairs of collaborating nodes. Considine et al. [89] proposed a heuristic scheme which can be used to construct low complexity erasure codes. These schemes not only emphasize the recovery of data, but are also centrally encoded. However, the centralized coding method cannot be employed directly in a sensor network, in which a sensor node is not able to store all the data segments and perform complicated encoding operations alone.

A promising solution for redundant data storage in sensor networks is the decentralized coding, which distributes the encoding operations from a sensor node to multiple sensor nodes. Such decentralized coding schemes include decentralized implementations of erasure coding [58, 59], growth coding [90], network coding [51, 67, 91], and Fountain coding [61, 92, 93].

Dimakis et al. [58] proposed an interesting coding scheme called Decentralized Erasure Coding (DEC), which may be applied for WSNs. In the DEC scheme, all the data segments recorded by a sensor node are encoded in a combined segment. The base station collects the data by randomly querying some sensor nodes. However, in DEC, the data encoded in a sensor node depend on the data routing form the selected sensor nodes, which increases the complexity of distributed encoding.

The usefulness of random linear coding for data storage was investigated in [65], where the authors showed that a simple distribution scheme using random linear coding and based only on local information can perform almost as well as the case where there was complete coordination among nodes. Similar considerations also has applied to distributed storage in sensor networks.

Lin et al. [94] studied how to differentiate data persistence by using priority random linear codes. They considered to maintain measurement data in different priorities. By using priority random linear codes, critical data has a higher opportunity to survive nodes than data of less importance. A salient feature of the priority random linear codes is that they have the ability to partially recover more important subsets of the original data with higher priorities, when they are not feasible to recover all of them due to node dynamics.

Kamra et al. [90] proposed a class of growth codes to increase the amount of data that can be recovered at the base station. In growth codes, the number of data encoded in an encoded data in each sensor node changes as the sensor node decodes more data from other sensors. Sensor nodes exchanged data with their neighbors and encoded received data with the existing local information, such that, the stored information was coded over more and more information units over time. That is, The number of encoded data in an encoded data starts with one, and the number grows (increases) as the encoded data being transmitted through the network to the base station. The growth codes can increase the amount of data that can be recovered at any storage node at any time period whenever some sensor nodes fail.

Munaretto et al. [91] proposed modifications to Growth Codes, which are able to achieve good performance over a wider range of static and dynamic scenarios. They investigated changes of how many and which data the transmitted information is coded over and how the decoding is performed.

Network coding and its distributed implementations utilized random linear codes, which allowed coding operations besides replication and forwarding on the intermediate nodes and achieve the maximal multicast capacity of a network [94]. Katti et. al. [69] proposed that nodes guess what data other nodes already have and exploit local coding opportunities to reduce the consumed bandwidth. Chunked Codes [95] reduced the complexity of random linear codes by partitioning message to chunks and utilizing pre-coding.

Jiang et al. [96] studied a joint data storage and transmission problem by using network coding, in which the data was transmitted to the sensor nodes whenever the data was updated. The joint storage and transmission problem can be transformed into a pure flow problem and is solvable in polynomial time using linear programming. Although coding is usually necessary for obtaining the optimal solution with the minimum cost, the authors proved that data splitting instead of coding is sufficient for achieving optimality, since adjacent nodes can have asymmetric links in networks of generalized tree structures. Thus, there exists an optimal solution for the joint data transmission and data storage problem, if there was no constraint on the numbers of bits that can be stored in the sensor nodes.

Similar to [96], Hou et al. [77] proposed a reliable data dissemination protocol called AdapCode which used adaptive network coding to reduce broadcast traffic in the process of code updates. The data in each node are coded by linear combination and decoded by Gaussian elimination. The core idea in AdapCode was to adaptively change the coding scheme according to the link quality. Widmer et al. [75] proposed a communication algorithm based on network coding, which significantly reduced the overhead of probabilistic routing algorithms, making it a suitable building block for a delay-tolerant network architecture. Nodes do not simply forward data they overhear and sent out data that are encoded over the amount of data they received. This algorithm achieves the reliability and robustness of flooding at a small fraction of the overhead.

Gkantsidis et al. [97] proposed a new scheme for content distribution of large files based on network coding. They studied the performance of network coding in heterogeneous networks with dynamic node arrival and departure patterns, clustered topologies, and when incentive mechanisms to discourage free-riding are in place. Simulations in varied scenarios are presented to show that network coding improves the robustness of the network and is able to smoothly handle extreme situations when the server and nodes departure the network.

Yang et al. [76] investigated the energy efficiency of distributed data storage in WSNs, and proposed a Compressed Network Coding based Distributed data Storage (CNCDS) scheme by exploiting the correlation of sensor readings, which was based on compressed sensing and network coding theories. The CNCDS scheme achieved high energy efficiency by reducing the total number of transmissions and receptions during the data dissemination process. Theoretical analysis proved that the proposed scheme guaranteed good compressed sensing recovery performance.

Fountain codes are employed for data storage in WSNs due to the efficiency in data encoding and decoding. Dimakis et al. [78] are the first to address the problem of constructing fountain codes for distributed storage in sensor networks. The authors proposed a randomized algorithm to construct fountain codes over grid network by using geographic routing. However, geographic routing requires sensor nodes to know their location, which is energy-inefficient and not suitable for resource constrained sensor nodes.

In [57, 61, 62, 98], the authors proposed distributed coding schemes based on fountain codes, in which the concept of random walks are used for data dissemination for the design of fountain codes. Random walks are routing approach which require only local information and low overhead [99–101]. In [98], several copies of each sensed data are let to randomly walk across the network during data dissemination. A variant of the Metropolis algorithm [102, 103] is employed to specify the transition probabilities in the random walks, so as to provide a stationary distribution design of the desired code degree distribution (i.e., the desire number of data to be encoded in each sensor node) in WSNs. However, the applied Metropolis algorithm requires global information about the WSN available in all sensor nodes to specify the transition probabilities. The scheme in [57] is similar, since multiple copies of sensed data are let to randomly walk around the WSNs. After being disseminated at least the number of hops equal to the cover time of the random walk on graph [104], each sensed data can be recorded by all the sensor nodes in WSNs. Each sensor node makes decisions and performs encoding immediately after each reception of sensed data.

Different from [57, 98] in which collecting sufficient number of sensed data to perform encoding is the task of sensor nodes, in [62] this task is assigned to encoded data. While randomly moving through the network, the encoded data collect and encode into their content required number of sensed data. The encoded data complete their paths in randomly selected sensor nodes. Thus, any degree distribution of encoded data (i.e., any desired number of data to be encoded in each encoded data) can be exactly obtained.

Cao et al. [61] considered the partial data recovery form one or some target source nodes, and proposed a distributed storage coding schemes based on fountain codes to ensure the flexible recovery of data measured by any given subset of source nodes of interest in wireless sensor networks. For this purpose, each sensor node encodes all the data received from the same source node into one encoded data segment so as to create a distributed Fountain code for all source nodes. Also, the proposed scheme in [61] takes advantage of broadcast nature of wireless transmission to improves the efficiency of traditional random walk for data disseminate.

For all of the above referenced schemes, the number of collected data is fixed and usually not large. The above referenced schemes cannot collect data in which the number is a variable. On the other hand, the above referenced schemes are lack of support removing obsolete (old) data, i.e., they cannot support the continuous data collection in which the number of data segments is larger and not predetermined.

Removing obsolete (old) data is another important issue for storing data in a sensor network, since each sensor node has limit storage space. If sensor nodes get unattended from the mBS for a long time (e.g., the bad weather prohibits the mBS from performing data collection for a long time), the total data may exceed the total storage space of the entire sensor network. In many practical applications, new data has higher value than old ones. Thus, a sensor node should be able to remove the old data in order to accommodate newly collected ones [51]. In the above referenced schemes, removing the old data includes decoding and re-encoding operations, which are time and resource consuming.

Wang et al. [51] proposed an interesting decentralized coding scheme called Partial Network Coding (PNC) for continuous data collection in a WSN with a mBS. PNC supports removing the obsolete data. Each combined segment encodes only the part of latest original data segments by removing the older data segments. The number of data segments encoded in a combined segment varies from 1 to m. By randomly querying a small subset of sensor nodes, the mBS can collect the m latest original data segments from the sensor network, where m is the number of latest original data segments in a time interval t in which $n(t)$ $(m \leq n(t))$ data segments are generated. However, not all the m latest original data segments are encoded in each combined segment. That is, the m latest original data segments cannot be always decoded completely when the mBS randomly queries some sensor nodes. Thus, PNC does not have high success ratio of collecting the m latest data segments. The success ratio of data collection in PNC can be improved by extending the storage space in each sensor node and extending the number of sensor nodes queried by the mBS. Note that the storage space of each sensor node depends on the number of latest original data segments. If the number of latest original data segments is large, the overhead for enhancement may be too big for a sensor node.

In this dissertation, we first consider m latest data segment collection and propose a novel distributed coding scheme with data replacement to collect the m latest data segments in the sensor networks. We also consider all data segment collection and proposed an efficient distributed coding scheme to collect all the data segments generated in a time interval. The total number of collect data is a variable. To the best of our knowledge, there is no existing study on distributed coding schemes to collect data with variable number. Since random linear coding is easy and suitable to deploy in wireless sensor networks, the proposed schemes in this dissertation are based on random linear coding. We will consider other coding methods for distributed storage in WSNs in the future work.

1.6 Dissertation Objectives and Motivations

The dissertation objective is to redundantly store the data in the sensor nodes by using coding method. The data can be preserved after some sensor nodes have failed. And, a large number of data can be stored in a small storage space of the sensor nodes after encoding. The mobile Base Station (mBS) can collect the desired data with high probability by decoding the encoded data from any subset of sensor nodes, even after some sensor nodes have failed.

There are two motivations in the dissertation.

- **Motivation 1**: In some harsh environment, removing obsolete (old) data is an important issue. The data is temporarily stored in each sensor node. If sensor nodes get unattended from the mBS for a long time, the total data may exceed the total storage space of the entire sensor network. The sensor nodes have to remove the old data to accommodate newly collected ones, sine new data has higher value than old ones, e.g., the pollution monitoring [105–107]. In this dissertation, we first consider the scenario of continuous data collection to collect the m latest data segments from the sensor network, where m is the number of latest original data segments in a time interval t in which $n(t)$ $(m \leq n(t))$ data segments are generated.

- **Motivation 2**: The sensor nodes may need to collect the data as many as possible and provide them to the mBS. In such kind of application, the overall trend of the data is important, e.g., the temperature monitoring [108]. After data collection by the mBS, the end users can try various physical models and test various hypotheses over a large amount data segments. Thus, we consider the second scenario of continuous data collection to collect all the data segments generated in a data sensing time interval t.

1.7 Our Contributions

In this dissertation, we consider two scenarios of continuous data collection. (1) m Latest data segment collection. (2) All data segment collection. We propose two Distributed Separate Coding schemes for the two scenarios, respectively.

1.7.1 Distributed Separate Coding for m Latest Data segment Collection (DSC-mLDC)

In Chapter 3, we propose Distributed Separate Coding for m Latest Data segment Collection (DSC-mLDC). We consider to collect the m latest data segments, where m is the number of latest data segments in a time interval t in which $n(t)$ $(m \leq n(t))$ data segments are generated. The proposed DSC-mLDC scheme is shown as an efficient method for continuously collecting data segments with a high success ratio. Compare to the related work (i.e., PNC in [51]), DSC-mLDC is flexible and efficient.

In DSC-mLDC, the necessary storage space in each sensor node does not depend on the number of required data. It can be adjusted by changing the number of sensor nodes queried by the mBS. And, the transmission cost for data submission to the mBS can be reduced with a few additional storage space in each sensor node. We compare DSC-mLDC with PNC. The discussion about the number of buffers and the number of data segments submitted from each node makes the scheme proposed much more convincing. The discussion and simulation both show that DSC-mLDC improves the performance in many situations.

1.7.2 Distributed Separate Coding for All Data segment Collection (DSC-ADC)

In Chapter 4, we propose Distributed Separate Coding for All Data segment Collection (DSC-ADC). We consider to collect all the $n(t)$ data segments generated in a time interval t. The proposed DSC-ADC scheme provides an efficient storage method to collect all data segments continuously. By randomly querying a small subset of sensor nodes, the mBS can reconstruct all the original data segments with high probability in both the right arrival case and the late arrival case. DSC-ADC is more energy efficient compared to the related work (i.e., DEC in [58]). We prove that the success ratio of DSC-ADC based data collection is close to 100% by using a large enough finite field size for the coefficients. The number of sensor nodes that should be queried by the mBS can be reduced with a few additional storage space in each sensor node. The performance evaluation has been conducted through computer simulations. It further demonstrates the feasibility and superiority of the proposed DSC-ADC scheme.

1.8 Dissertation Organization

This dissertation is organized as follows. This Chapter covers the background and overview of our research. The system model and problem formulation are presented in Chapter 2. In Chapter 3,we propose Distributed Separate Coding for m Latest Data segment Collection (DSC-mLDC) in wireless sensor network with a mobil base station. In Chapter 4, we propose Distributed Separate Coding for All Data segment Collection (DSC-ADC) in wireless sensor network with a mobil base station. Finally we conclude this dissertation and point out the future work in Chapter 5.

Chapter 2

System Description and Problem Formulation

In this chapter, we give the system description and problem formulation that are used in Chapter 3 and Chapter 4.

This chapter is organized as follows. The system model and notations are presented in Section 2.1. We introduce two methods to generate the coefficients in random linear coding in Section 2.2.

2.1 System Model and Notations

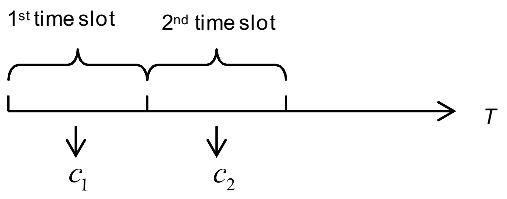

FIGURE 2.1: One data segment is generated in a fixed time slot. c_j is generated in the j^{th} time slot.

Consider that there are N sensor nodes in a wireless sensor network, where a set of sensor nodes sense information. Each sensor node has B buffers, b_1, b_2,..., b_B (i.e., the buffer size of each sensor node is B). Each buffer can store only one data segment. Consider to collect the data of samples (e.g., the temperatures measured in the beginning of some fixed time slots) by using a WSN, where the samples are generated continuously. A sample is represented by one data segment c_j, and generated in a fixed time slot, as shown in Fig 2.1. c_j is generated over the j^{th} time slot. c_q is newer than c_p if $q > p$.

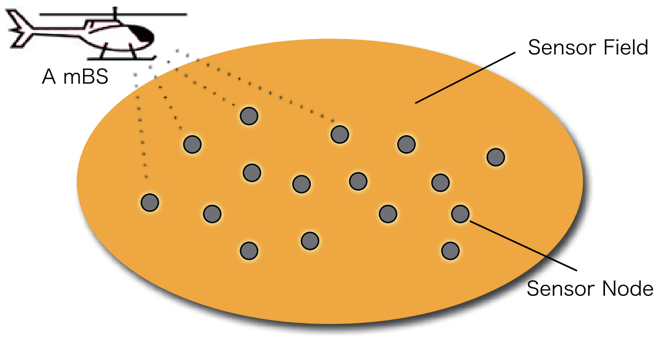

FIGURE 2.2: Data collection by a mBS.

Without loss of generality, we consider that there is one mBS (mobile Base Station) which performs data collection from time to time. For example, a helicopter acts as the mBS, as illustrated in Fig 2.2. During the data collection, the mBS will query a small subset of sensor nodes uniformly at random from the sensor network to collect data.

Consider that the total number of data segments is $n(t)$, where t is the data sensing time interval. Note that t is a variable, which value depends on when the mBS performs data collection. $n(t)$ is a nondecreasing function of t. As an example shown in Fig 2.3, data sensing starts at time t_1, the mBS starts to perform data collection at time t_2. The time interval t between t_1 and t_2 is the data sensing time interval. The data encoding and storage for the sensed data segments are also done in this time interval in each sensor node. The data collection by the mBS is

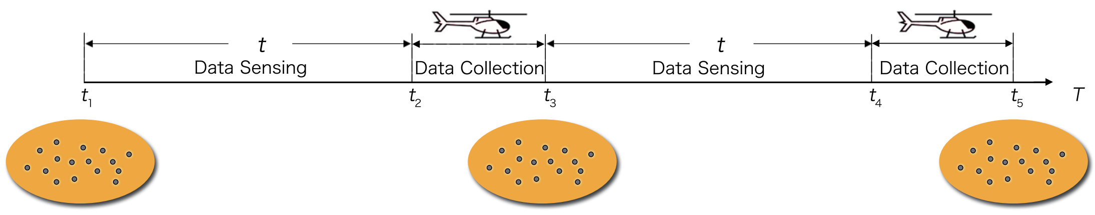

FIGURE 2.3: Continuous data collection. t is the data sensing time interval. Data collection by the mBS is performed between time t_2 and t_3 and between time t_4 and t_5.

performed between t_2 and t_3. If the mBS cannot start to perform data collection until a long time period due to some special reasons (e.g., the bad weather), the data sensing time interval t will be longer. Thus, the total number of data segments $n(t)$ generated during this time interval will be larger.

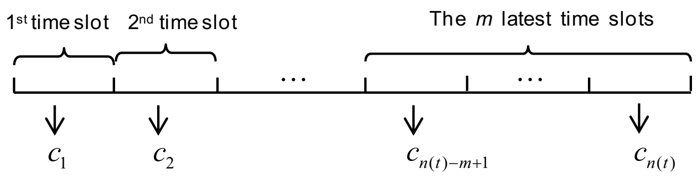

FIGURE 2.4: Data generation in a time interval t.

In this dissertation, we consider two scenarios of continuous data collection. (1) Latest data segment collection. (2) All data segment collection.

In latest data segment collection, we consider to collect the m ($m \leq n(t)$) *latest original data segments*, where m is the number of latest data segments to be collected. Note that, in a time interval t, no matter how many data segments are generated, the required data segments are the m latest original data segments which are generated in the m latest time slots (i.e., the data segments from $c_{n(t)-m+1}$ to $c_{n(t)}$), as shown in Fig 2.4.

In all data segment collection, we consider to collect *all the data segments* generated in a data sensing time interval t. The total number of data segments to be collected is $n(t)$ (i.e., the data segments from c_1 to $c_{n(t)}$), as shown in Fig 2.4.

We show the differences between the two scenarios of continuous data collection. In latest data segment collection, the number of collected data m is a fixed number. The value m is set before data collection. Thus, the mBS and the sensor nodes know the value m before data collection. In all data segment collection, the number of collected data $n(t)$ is a variable. The value $n(t)$ depends on when the mBS performs data collection. Thus, the mBS and the sensor nodes do not know the value $n(t)$ before data collection. m may equal to $n(t)$ on the condition that the mBS performs data collection when the total number $n(t)$ equals to m. However, the value m can not be set to be $n(t)$, since the value $n(t)$ is a variable which can not be known to the mBS and the sensor nodes before data collection.

For data coding, we define a linear function as follows.

$$f_i^u = \sum_{j=1}^{k} \beta_{ij} c_j, \tag{2.1}$$

where f_i^u is referred to as a *combined segment*, which encodes k data segments c_1,..., c_k ($1 \leq k \leq n(t)$) in buffer b_u of sensor node i, as shown in Fig 4.2. $\vec{\beta}_i^u = (\beta_{i1}, \beta_{i2}, \cdots, \beta_{ik})$ is a coefficient vector of f_i^u. Each item β_{ij} is generated from a finite field F_q, where q is the finite field size. β_{ij} is the coefficient of c_j in $\vec{\beta}_i^u$. We will show how to generate the coefficient vector $\vec{\beta}_i^u$ in Section 2.2 in this chapter. Note that the size (in bits) of a combined segment f_i^u equals to the size (in bits) of an original data segment c_j. Sensor node i stores the combined segment f_i^u and the associated coefficient vector $\overrightarrow{\beta_i^u}$ in buffer b_u, instead of storing the k original data segments c_1,..., c_k.

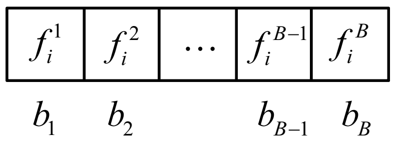

FIGURE 2.5: The combined segment f_i^u is stored in the buffer b_u of sensor node i.

Since sensor nodes sense the similar environment and collect the data, we assume that the data segments for encoding in each sensor node are the same in a time slot. To achieve this condition, the sensor nodes may also communicate with each other to disseminate data segments. Many methods have been proposed for data dissemination for wireless sensor networks (e.g., the data dissemination methods in [57, 98, 109]). This dissertation focuses on the data encoding process in each sensor node and the data decoding process in the mBS. We do not address the data dissemination process. For the sake of convenience, we assume that each data segment is recorded by all the sensor nodes by using some existing data dissemination method. The case that each data segment is not recorded by all the sensor nodes also has been discussed in Chapter 3 and Chapter 4.

To successfully decode the required original data segments, the mBS should access enough number of sensor nodes for the data collection. The mBS first accesses a small subset (i.e., the minimum number) of sensor nodes uniformly at random from the sensor network. Each accessed sensor node uploads the stored combined segments to the mBS. If the mBS cannot decode the required original data segments from the collected combined segments, it will access additional sensor nodes one by one until the required original data segments are decoded. The additional sensor nodes are accessed uniformly at random from the sensor network (excluding the sensor nodes which have accessed in the same time of data collection). The sensor network will consume more energy if the mBS repeats data collection, since more sensor nodes need to upload data to the mBS. Thus, the success ratio of data collection is a major evaluation criterion in the study [51]. We define the success ratio of data collection as follows.

Definition 2.1 (Success ratio of data collection)**.** The success ratio of data collection is the probability that the mBS successfully collects all the m latest original data segments.

For the sake of convenience, a list of the notations is given in Table 2.1.

Table 2.1: List of Notations.

Notation	Definition
N	Number of sensor nodes
B	Buffer size of each sensor node
b_i	Buffer with index i
m	Number of latest original data segments
t	Data sensing time interval
$n(t)$	Number of data segments generated in time interval t
c_j	Original data segment with sequence number j
f_i^u	Combined segment stored in buffer b_u of sensor node i
$\vec{\beta}_i^u$	Coefficient vector of f_i^u
β_{ij}	Coefficient of c_j in $\vec{\beta}_i^u$
$C(f_i^u)$	Number of data segments encoded in f_i^u
q	Size of finite field for coefficients

2.2 Two Methods to Generate the Coefficients in Random Linear Coding

We show two methods to generate the coefficient vector $\vec{\beta}_i^u = (\beta_{i1}, \beta_{i2}, \cdots, \beta_{ik})$ in equation (2.1). One is the general method, in which each β_{ij} is randomly generated from a finite field F_q, where q is the finite field size. Sensor node i should generate k coefficients $\beta_{i1}, \beta_{i2}, \cdots, \beta_{ik}$ for a coefficient vector.

The other method is proposed in [110]. We call it *Suli and Mayers method*. With *Suli and Mayers method*, the coding function in equation (2.1) can be changed as

$$f_i^u = \sum_{j=1}^{k} \beta_i^{j-1} c_j. \tag{2.2}$$

$\vec{\beta}_i^u = (1, \beta_i, \beta_i^2, \cdots, \beta_i^{k-1})$ is the coefficient vector of the combined segment f_i^u, which is stored in buffer b_u of sensor node i. Sensor node i only needs to generate one coefficient β_i for the k encoded original segments, instead of generating k coefficients $\beta_{i1}, \beta_{i2}, \cdots, \beta_{ik}$.

2.2.1 Benefit for Data Storage in Random Linear Coding

In the *general method*, each sensor node generates k coefficients for a coefficient vector. Storing the coefficient vector $\vec{\beta}_i^u = (\beta_{i1}, \beta_{i2}, \cdots, \beta_{ik})$ will take an additional storage space of $k \log_2(q)$ bits. Assume that the size of a combined segment f_i^u is w bits (equals to the size of an original data segment). With coding in equation (2.1), each sensor node stores the combined segment f_i^u and the associated coefficient vector $\overrightarrow{\beta_i^u}$ with $w + k \log_2(q)$ bits. Without coding, each sensor node should store k original data segments c_1,..., c_k with kw bits. Thus, in the *general method*, the percentage of storage overhead reduction P_{sr}

$$P_{sr} = (1 - \frac{w + k \log_2(q)}{kw})\%. \tag{2.3}$$

As shown in equation (2.3), P_{sr} depends on the size of data segment w and the number of original data segments encoded in a combined segment k. The storage overhead can be reduced a lot if the size of data segment is large. That is, the storage overhead of the coefficient vector is very small compared to large size of combined segment f_i^u. Take an example similar to that considered in [65], the size of a combined segment f_i^u is 20 KB, and the size of finite field for coefficients is $q = 2^8$. If f_i^u encodes 10 original data segments, the storage overhead for the coefficient vector $\overrightarrow{\beta_i^u}$ is 80 bits or 10 bytes. Thus, the additional storage space required for the coefficient vector is less than 0.05 %, which is a negligible overhead compared to the combined segment. Note that increase the number of original data segments encoded in a combined segment k also can reduce the storage overhead. However, if the number of original data segments encoded in a combined segment increases, the storage overhead for the coefficient vectors also increases.

In the *Suli and Mayers method*, each sensor node i only needs to generate one coefficient β_i for the k encoded original segments, instead of generating k coefficients. Therefore, each senor node only needs to store a coefficient for a combined segment, no matter the combined segment encodes how many original data segments. Consider that a combined segment with size w bits encodes k original data segments. By using the method in [110], in the *Suli and Mayers method*, the percentage of storage overhead reduction P_{sr} is

$$P_{sr} = (1 - \frac{w + \log_2(q)}{kw})\%. \tag{2.4}$$

In equation (2.4), P_{sr} increases as the size of data segment w and the number of original data segments encoded in a combined segment k increase. The storage overhead for the coefficient vectors remains the same if the value of k increases. Thus, *Suli and Mayers method* is more suitable when the size of combined segment f_i^u is no so large or the encoded number k is large.

2.2.2 Probability of Linear Independency for Coefficient Vectors

The original data segments c_1,..., c_k in equation (2.1) can be decoded by solving a set of linear equations of k combined segments f_1^u,..., f_k^u, as shown in equation (2.5).

$$\begin{pmatrix} f_1^u \\ f_2^u \\ \vdots \\ f_k^u \end{pmatrix} = \begin{pmatrix} \overrightarrow{\beta_1^u} \\ \overrightarrow{\beta_2^u} \\ \vdots \\ \overrightarrow{\beta_k^u} \end{pmatrix} (c_1 \; c_2 \; \ldots \; c_k), \tag{2.5}$$

where the k combined segments f_1^u,..., f_k^u and k coefficient vectors $\overrightarrow{\beta_1^u}$,..., $\overrightarrow{\beta_k^u}$ are collected from buffers b_u of k distinct sensor nodes. The necessary condition for decoding is that the k coefficient vectors $\overrightarrow{\beta_1^u}$,..., $\overrightarrow{\beta_k^u}$ must be linearly independent.

In the *general method*, the probability of linear independency for k coefficient vectors is over 99.6% when the size of finite field for coefficients $q = 2^8$, and this probability is almost independent of k [51]. The probability of linear independency for coefficient vectors increases as the finite field size q increases.

In the *Suli and Mayers method*, the probability of linear independence for k coefficient vectors can be calculated as

$$p = \prod\nolimits_{i=0}^{k-1} \frac{q-i}{q}, \tag{2.6}$$

where q is the size of finite field F_q. Note that k also equals to the number of original data segments encoded in a combined segment. A way to improve the probability of linear independence for the coefficient vectors is using larger size q for the finite field. As shown in Fig 2.6, the probability of linear independence is remarkably improved from $q = 2^8$ to $q = 2^{16}$. The probability of linear independence is very close to 100% when $q = 2^{16}$. Note that the increase of finite field size q costs only logarithmic additional bits. For example, the probability of linear independence can be improved remarkably, while the additional storage for a coefficient is only 8 bits (from $q = 2^8$ to $q = 2^{16}$).

Note that the *Suli and Mayers method* requires $q = 2^{16}$ to guarantee that the probability of linear independence for k coefficient vectors is close to 100%. In the

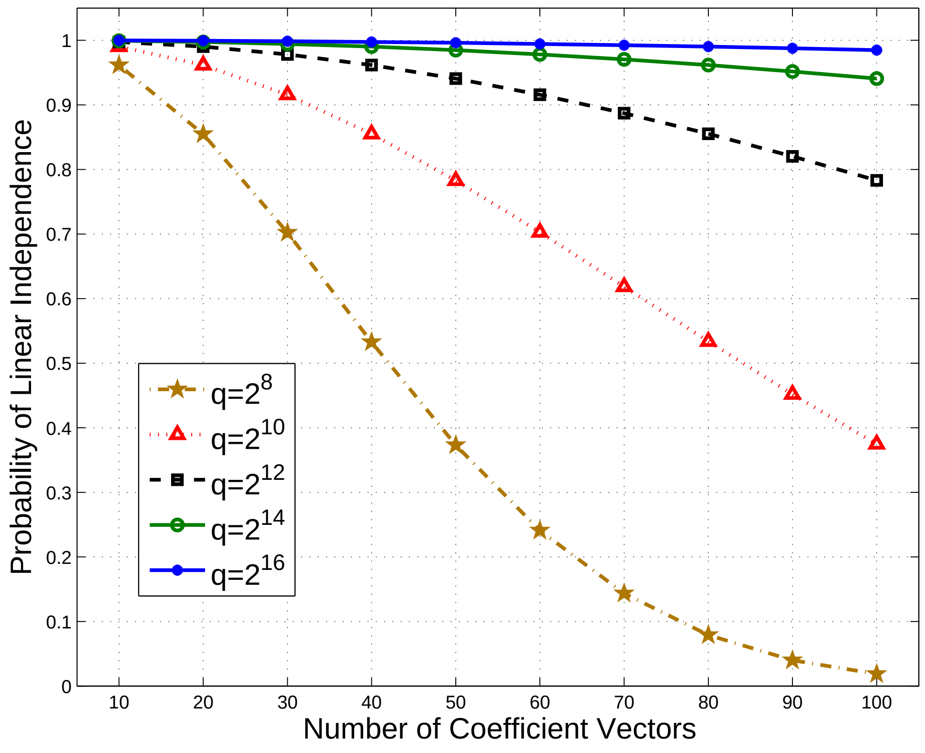

FIGURE 2.6: Probability of linear independence vs. the number of coefficient vectors in the *Suli and Mayers method*.

general method where each coefficient is generated randomly from finite field F_q, the probability of linear independence for k coefficient vectors is close to 100% when $q = 2^8$ [51]. Although the *Suli and Mayers method* requires larger q, it can reduce the storage overhead of the coefficients. And, from $q = 2^8$ to $q = 2^{16}$, the additional storage for a coefficient is only 8 bits. Due to the additional 8 bits for a coefficient, the computation overhead for data encoding in the *Suli and Mayers method* is a little larger than that in the general method.

Chapter 3

Distributed Separate Coding for m Latest Data Segment Collection

In this chapter, we focus on the continuous data collection issue in WSNs with a mBS. We consider the first scenario of continuous data collection (i.e., m Latest data segment collection), and present a novel data collection scheme called Distributed Separate Coding for m Latest Data segment Collection (DSC-mLDC). DSC-mLDC is decentralized and based on the mBS's randomly accessing. By separately encoding a certain number of data segments in a combined segment, and doing decoding-free data replacement in the buffers of each sensor node, DSC-mLDC provides an efficient storage method for continuously collecting data segments with a high success ratio.

This chapter is organized as follows. In Section 3.1, we give an overview of the proposed DSC-mLDC scheme. In Section 3.2, we present DSC-mLDC for the case that each sensor node has two buffers (i.e., buffer size $B = 2$). In Section 3.3, we present DSC-mLDC for the case that each sensor node has more than 2 buffers (i.e., buffer size $B > 2$). Performance analysis and comparison are presented in Section 3.4. In Section 3.5, we evaluate the performance of the proposed scheme through simulations. In Section 3.6, we present the discussion. Section 3.7 concludes this chapter.

3.1 An overview of DSC-mLDC

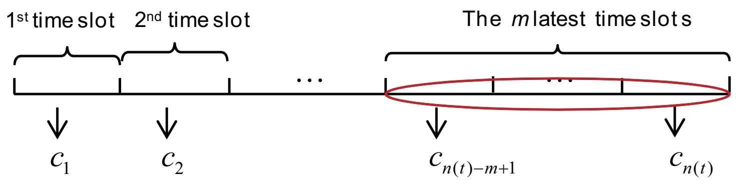

FIGURE 3.1: The m latest original data segments are generated in the m latest time slots (i.e., the time slots in the circle).

In the proposed DSC-mLDC scheme, we consider to collect the m latest data segments, where m is the number of latest data segments in a time interval t in which $n(t)$ ($m \leq n(t)$) data segments are generated, as shown in Fig 3.1. In DSC-mLDC, the data segments are separately encoded in a combined segment in each sensor node. If all the buffers are stored with combined segments, the sensor nodes will do decoding-free data replacement to store the new combined segments. DSC-mLDC includes three processes: the *data encoding process*, the *data replacement process* and the *data decoding process*. The data encoding and replacement processes are performed in each sensor node, while the data decoding process is performed in the mBS.

In this scenario of continuous data collection (i.e., Latest data segment collection), we consider two cases: (1) each sensor node has two buffers (i.e., buffer size $B = 2$); (2) each sensor node has more than 2 buffers (i.e., buffer size $B > 2$).

Note that PNC [51] also addresses the continuous data collection in WSNs. We compare the proposed DSC-mLDC scheme with PNC, since it is the existing scheme that has an efficient solution for continuous data collection in WSNs. To the best of our knowledge, PNC is the only one scheme which can support data replacement for continuously collecting data segments. The comprehensive performance evaluation has been conducted through computer simulation. It is shown that the proposed DSC-mLDC scheme is the most recommendable one.

3.2 DSC-mLDC for the case that each sensor node has two buffers

Consider that each sensor node has two buffers, denoted by b_1 and b_2. Let $F_i = \{f_i^1, f_i^2\}$ be a set of combined segments stored in sensor node i, where f_i^1 is stored in buffer b_1 and f_i^2 is stored in buffer b_2. The associated coefficient vectors $\overrightarrow{\beta_i^1}$ and $\overrightarrow{\beta_i^2}$ are also stored in buffers b_1 and b_2, respectively.

3.2.1 Data encoding and replacement

In DSC-mLDC, each sensor node separately encodes $m-1$ original data segments in a combined segment. We will show later that $m-1$ is the minimum number of original data segments encoded in a combined segment, which can ensure that the two combined segments stored in each sensor node encode all the m latest original data segments. Let $f_i(r)$ be a combined segment which encodes the r^{th} recorded $m-1$ original data segments in sensor node i. As an example shown in Fig 3.2, the first recorded $m-1$ original data segments c_1, c_2,..., c_{m-1} are encoded in $f_i(1)$. The second recorded $m-1$ original data segments c_m, c_{m+1},..., $c_{2(m-1)}$ are encoded in $f_i(2)$. And the third recorded $m-1$ original data segments c_{2m-1}, c_{2m},..., $c_{3(m-1)}$ are encoded in $f_i(3)$. Generally, we have

$$f_i(r) = \sum_{j=(r-1)(m-1)+1}^{r(m-1)} \beta_{ij} c_j, \quad r = 1, 2, \dots \tag{3.1}$$

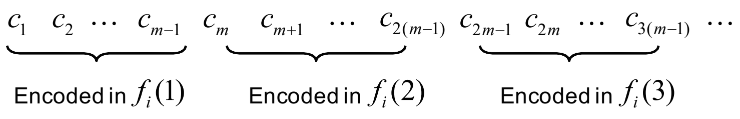
c_1 c_2 $\cdots$ c_{m-1} c_m c_{m+1} $\cdots$ $c_{2(m-1)}$ c_{2m-1} c_{2m} $\cdots$ $c_{3(m-1)}$ $\cdots$
Encoded in $f_i(1)$
Encoded in $f_i(2)$
Encoded in $f_i(3)$

FIGURE 3.2: $m-1$ original data segments are separately encoded in a combined segment.

When a new combined segment $f_i(r)$ is formed, $f_i(r)$ and its associated coefficient vector are stored in a corresponding buffer of sensor node i. If $f_i(r)$ is stored in buffer b_1 of sensor node i, we have $f_i^1 = f_i(r)$. And if $f_i(r)$ is stored in buffer b_2 of sensor node i, we have $f_i^2 = f_i(r)$. A new combined segment encodes the latest original data segments. Note that the mBS wants to collect the m latest original data segments. If there has been a combined segment stored in the corresponding buffer, the new combined segment including the associated coefficient vector will replace the old ones. Let $f_i(l)$ be the latest combined segment formed before the mBS performs data collection. By the data replacement, each sensor node stores the two latest combined segments $f_i(l-1)$ and $f_i(l)$, which encode at least m latest original data segments. We may further drop the subscript i of $f_i(r)$ if the sensor node which stores the combined segment is clear in its context.

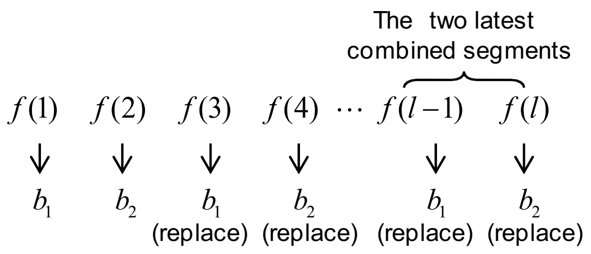

FIGURE 3.3: Data replacement to store the two latest combined segments.

An example is shown in Fig 3.3. At first $f(1)$ and $f(2)$ are formed, and are stored in buffers b_1 and b_2 of sensor node i, respectively. That is, $f_i^1 = f(1)$ and $f_i^2 = f(2)$. Note that each sensor node has two buffers. When $f(3)$ is formed, $f(3)$ replaces $f(1)$ to be stored in buffer b_1 of sensor node i, i.e., $f_i^1 = f(3)$. The data replacement is performed by each sensor node until the mBS performs data collection. With the data replacement, the two latest combined segments $f(l-1)$ and $f(l)$ are stored in the two buffers of sensor node i when the mBS performs data collection. The replacement of the combined segments are performed with the replacement of the associated coefficient vectors (e.g., if $f(3)$ replaces $f(1)$, the associated coefficient vector of $f(3)$ also replaces the associated coefficient vector of $f(1)$). Relation (3.2) shows how to store the combined segment $f(r)$ (including the associated coefficient vector) in the corresponding buffer of sensor node i.

$$\begin{cases} f_i^1 = f(r), & r \text{ is odd}, \\ f_i^2 = f(r), & r \text{ is even}. \end{cases} \tag{3.2}$$

We give a formal description of data encoding and replacement algorithm of SNC-mLDC when each sensor node has two buffers. Every time an original data segment c_j is encoded with a combined data segment. The data encoding algorithm (Algorithm 1) is locally executed at each sensor node. In Algorithm 1, $\lceil j/m \rceil$ is the upper integer bound of j/m (e.g., $\lceil 1/3 \rceil = 1$ and $\lceil 4/3 \rceil = 2$). The encoding process is with the data replacement to encode only the amount of latest original data segments in the combined data segments.

Note that $f(l)$ is the latest combined segment formed before the mBS performs data collection. The number of original data segments encoded in $f(l)$ depends on the total number of original data segments $n(t)$. If $n(t) = l(m-1)$, the number of original data segments encoded in $f(l)$ is $m-1$. And if $n(t) = (l-1)(m-1)+1$, the number of original data segments encoded in $f(l)$ is 1. Note that $n(t)$ cannot be larger than $l(m-1)$ or less than $(l-1)(m-1)+1$, otherwise $f(l)$ is not the latest combined segment. For example, if $n(t) = l(m-1)+1$, the extra one original data segment is encoded in a newer combined segment $f(l+1)$. Then, $f(l+1)$ is the latest combined segment. If $n(t) = (l-1)(m-1)$, the last $m-1$ original data segment are encoded in $f(l-1)$. $f(l-1)$ is the latest combined segment. Thus, $f(l)$ is the latest combined segment on condition that

$$(l-1)(m-1)+1 \leq n(t) \leq l(m-1). \tag{3.3}$$

From equation (3.3), the number of original data segments encoded in the latest combined segment $f(l)$ is with an upper bound $m-1$ and a lower bound 1. The

Algorithm 1: Data encoding when each sensor node has two buffers

```
Input: Original data segment $c_j$, number of latest data segments $m$.
Output: A set of combined segments $F_i = \{f_i^1, f_i^2\}$.
1  for $j = 1$ to $n(t)$ do
2      Let $r = \lceil j/(m-1) \rceil$ ;
3      Randomly generate $\beta_{ij}$ from $F_q$ ;
4      if $r$ is odd then
5          if $C(f_i^1) < m - 1$ then
6              $f_i^1 = f_i^1 + \beta_{ij} c_j$;
7          end
8          else
9              $f_i^1 = \beta_{ij} c_j$
10         end
11     end
12     else
13         if $C(f_i^2) < m - 1$ then
14             $f_i^2 = f_i^2 + \beta_{ij} c_j$;
15         end
16         else
17             $f_i^2 = \beta_{ij} c_j$
18         end
19     end
20 end
```

number of original data segments encoded in $f(l-1)$ is $m-1$. Therefore, in each sensor node, the number of original data segments encoded in the two latest combined segments $f(l-1)$ and $f(l)$ is with an upper bound $2m-2$ and a lower bound m. Then we obtain the following lemma.

Lemma 3.1. *In DSC-*m*LDC, the set of original data segments which are encoded in the two latest combined segments $f(l-1)$ and $f(l)$ in each sensor node, includes all the m latest original data segments.*

Proof. The latest original data segments are encoded in the two latest combined segments $f(l-1)$ and $f(l)$. The number of latest original data segments encoded in

$f(l-1)$ and $f(l)$ is at least m. Thus, the two latest combined segments encode all the m latest original data segments. $\square$

We give an example to show that the two latest combined segments in each sensor node encode all the m latest original data segments. Let x denote the number of original data segments encoded in a combined segment. The original data segments are c_1, c_2, $\cdots$, c_{11}. In this example, let $m = 4$, i.e., the mBS wants to collect the 4 latest original data segments c_8, c_9, c_{10}, and c_{11}. For the sake of convenience, we define

$$[c_1, ..., c_k] = \sum\nolimits_{j=1}^{k} \beta_{ij} c_j \tag{3.4}$$

to denote the combined data segments by omitting the coefficients β_{i1},..., β_{ik}. Each combined segment encodes 3 ($x = m - 1 = 3$) original data segments. Each sensor node has two buffers to store the two latest combined segments $f(l-1)$ and $f(l)$, where $f_i^1 = f(l-1)$ and $f_i^2 = f(l)$. As shown in Fig 3.4 (a), the two latest combined segments encode all the 4 latest original data segments c_8, c_9, c_{10}, and c_{11}.

$$\{f_i^1 = [c_7, c_8, c_9], f_i^2 = [c_{10}, c_{11}]\}$$

(a) $B = 2$, $x = m - 1 = 3$

$$\{f_i^1 = [c_9, c_{10}], f_i^2 = [c_{11}]\}$$

(b) $B = 2$, $x = m - 2 = 2$

$$\{f_i^1 = [c_9, c_{10}, c_{11}]\}$$

(c) $B = 1$, $x = m = 4$

Figure 3.4: Data distribution in sensor node i with $m = 4$. x is the number of original data segments encoded in a combined segment. (a) $B = 2$, $x = m - 1 = 3$, (b) $B = 2$, $x = m - 2 = 2$, (c) $B = 1$, $x = m = 4$.

We prove that $m-1$ is the minimum number of original data segments encoded in a combined segment when each sensor node has two buffers in the following lemma.

Lemma 3.2. *For each sensor node with two buffers in DSC-*m*LDC, $m - 1$ is the minimum number of original data segments encoded in a combined segment.*

Proof. Each sensor node separately encodes $m - 1$ original data segments in a combined segment. From Lemma 3.1, the two latest combined segments $f(l - 1)$ and $f(l)$ encode all the m latest original data segments. Here, we prove that if less than $m-1$ original data segments are encoded in a combined segments, $f(l-1)$ and $f(l)$ cannot always encode all the m latest original data segments. Consider that $m - 2$ original data segments are encoded in a combined segment. The mBS performs data collection when $f(l - 1)$ encodes $m - 2$ original data segment and $f(l)$ encodes one original data segment. In this case, $f(l-1)$ and $f(l)$ encode only $m-1$ original data segments. In the above example, the mBS wants to collect the 4 latest original data segments c_8, c_9, c_{10}, and c_{11}. If each combined segment encodes 2 $(x = m - 2 = 2)$ original data segments, by data replacement, the two latest combined segments encode only 3 latest original data segments when the mBS performs data collection, as shown in Fig 3.4 (b). □

Furthermore, we prove that the minimum buffer size for a sensor node in DSC-mLDC is two in the following lemma.

Lemma 3.3. *In DSC-*m*LDC, the minimum buffer size for a sensor node to store the combined segments which encode all the m latest original data segments is two.*

Proof. In DSC-mLDC, each sensor node with two buffers can store the two latest combined segments $f(l - 1)$ and $f(l)$, which encode the m latest original data segments. Here we prove that each sensor node with one buffer cannot do it. Consider that each sensor node has only one buffer. By the data replacement, each sensor node stores only the latest combined segment $f(l)$. No matter how many original data segments are encoded in a combined segment, when $f(l)$ replaces $f(l - 1)$, the number of original data segments encoded in $f(l)$ has a lower bound 1. If the mBS performs data collection when $f(l)$ encodes less than m original data segments, $f(l)$

cannot encode all the m latest original data segments. In the above example, the mBS wants to collect the 4 latest original data segments c_8, c_9, c_{10}, and c_{11}. If each sensor node has one buffer, it can store only one combined segment. Consider that the combined segment can encode at most 4 ($x = m = 4$) original data segments. However, by data replacement, the combined segment encodes only 3 latest original data segments when the mBS performs data collection, as shown in Fig 3.4 (c). $\square$

3.2.2 Data decoding

The mBS first collects the data by querying any $m - 1$ sensor nodes. Then the mBS performs the data decoding based on the collected data.

Note that there are two combined segments stored in each sensor node. A sensor node queried by the mBS will upload the two combined segments and the associated coefficient vectors to the mBS. In DSC-mLDC, with the data replacement, each sensor node stores the two latest combined segments $f(l-1)$ and $f(l)$. The replaced combined segments are $f(1)$, ..., $f(l-2)$. In each replaced combined segment, the number of encoded original data segments is $m-1$. Thus, the total number of original data segments encoded in the $l-2$ replaced combined segments is $(l-2)(m-1)$. The set of original data segments encoded in the $l-2$ replaced combined segments is $\{c_1, c_2, ..., c_{(l-2)(m-1)}\}$. And the set of original data segments encoded in the two combined segments $f(l-1)$ and $f(l)$ is $\{c_{(l-2)(m-1)+1}, c_{(l-2)(m-1)+2}, ..., c_{n(t)}\}$. We have

$$f(l-1) = \sum_{j=(l-2)(m-1)+1}^{(l-1)(m-1)} \beta_{ij} c_j, \tag{3.5}$$

and

$$f(l) = \sum_{j=(l-1)(m-1)+1}^{n(t)} \beta_{ij} c_j. \tag{3.6}$$

Note that the number of original data segments encoded in $f(l-1)$ is $m-1$. The number of original data segments encoded in $f(l)$ is $n(t)-(l-1)(m-1)$, where $n(t)-(l-1)(m-1) \leq m-1$. In the data encoding process, we have shown that the number of original data segments encoded in $f(l-1)$ and $f(l)$ is with an upper bound $2m-2$ and a lower bound m. The following relation holds.

$$m \leq n(t) - (l-2)(m-1) \leq 2m-2. \tag{3.7}$$

Then,

$$\frac{n(t)}{m-1} \leq l \leq \frac{n(t)+m-2}{m-1}. \tag{3.8}$$

Without loss of generality, we consider that $f_i^1 = f(l-1)$ and $f_i^2 = f(l)$. That is, $f(l-1)$ is stored in buffer 1 of sensor node i and $f(l)$ is stored in buffer 2 of sensor node i.

For convenience, let

$$\mathbf{f_1} = (f_1^1, f_2^1, \ldots, f_N^1)^T, \tag{3.9}$$

where f_i^1 is the combined segment stored in buffer b_1 of sensor node i, $i = 1, ..., N$. $\mathbf{f}_1$ includes the combined segments stored in buffer b_1 of N sensor nodes. Let

$$\boldsymbol{\beta_1} = \begin{pmatrix} \vec{\beta}_1^1 \\ \vec{\beta}_2^1 \\ \vdots \\ \vec{\beta}_N^1 \end{pmatrix} = \begin{pmatrix} \beta_{1.k(m-1)+1} & \beta_{1.k(m-1)+2} & \cdots \beta_{1.(k+1)(m-1)} \\ \beta_{2.k(m-1)+1} & \beta_{2.k(m-1)+2} & \cdots \beta_{2.(k+1)(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N.k(m-1)+1} & \beta_{N.k(m-1)+2} & \cdots \beta_{N.(k+1)(m-1)} \end{pmatrix}, \tag{3.10}$$

and

$$\mathbf{c_1} = \left(c_{k(m-1)+1}, c_{k(m-1)+2}, \ldots, c_{(k+1)(m-1)}\right)^T, \tag{3.11}$$

where $\vec{\beta}_i^1$ is the associated coefficient vector of f_i^1, and $\mathbf{c_1}$ is the set of original data segments encoded in each f_i^1, $i = 1, ..., N$.

From equation (3.5), we have

$$\mathbf{f}_1 = \boldsymbol{\beta}_1 \mathbf{c}_1. \tag{3.12}$$

Similarly, let

$$\mathbf{f}_2 = (f_1^2, f_2^2, \ldots, f_N^2)^T, \tag{3.13}$$

where f_i^2 is the combined segment stored in buffer b_2 of sensor node i, $i = 1, ..., N$. $\mathbf{f}_2$ includes the combined segments stored in buffer b_2 of N sensor nodes. Let

$$\boldsymbol{\beta_2} = \begin{pmatrix} \vec{\beta}_1^2 \\ \vec{\beta}_2^2 \\ \vdots \\ \vec{\beta}_N^2 \end{pmatrix} = \begin{pmatrix} \beta_{1.(k+1)(m-1)+1} & \beta_{1.(k+1)(m-1)+2} & \ldots \beta_{1.n(t)} \\ \beta_{2.(k+1)(m-1)+1} & \beta_{2.(k+1)(m-1)+2} & \ldots \beta_{2.n(t)} \\ \vdots & \vdots & \ddots \quad \vdots \\ \beta_{N.(k+1)(m-1)+1} & \beta_{N.(k+1)(m-1)+2} & \ldots \beta_{N.n(t)} \end{pmatrix}, \tag{3.14}$$

and

$$\mathbf{c}_2 = \left(c_{(k+1)(m-1)+1}, ..., c_{n(t)}\right)^T, \tag{3.15}$$

where $\vec{\beta}_i^2$ is the associated coefficient vector of f_i^2, and $\mathbf{c_2}$ is the set of original data segments encoded in each f_i^2, $i = 1, ..., N$.

From equation (3.6), we have

$$\mathbf{f}_2 = \boldsymbol{\beta}_2 \mathbf{c}_2. \tag{3.16}$$

We show that the mBS can decode the m latest original data segments by querying any $m-1$ sensor nodes with high probability. The sensor nodes queried by the mBS will upload the two combined segments and the two associated coefficient vectors. The mBS querying $m-1$ sensor nodes will gain access to $2(m-1)$ combined segments. To decode the original data segments, it must invert a $(m-1) \times (m-1)$ submatrix $\boldsymbol{\beta}'_1$ of $\boldsymbol{\beta}_1$ and invert a $(n(t)-(l-1)(m-1)) \times (n(t)-(l-1)(m-1))$ submatrix $\boldsymbol{\beta}'_2$ of $\boldsymbol{\beta}_2$. Therefore, the key property required for successfully decoding is that any selection of $\boldsymbol{\beta}'_1$ and $\boldsymbol{\beta}'_2$ form full rank matrixes with high probability. A necessary condition is that the coefficient vectors in $\boldsymbol{\beta}'_1$ and the coefficient vectors in $\boldsymbol{\beta}'_2$ must be linearly independent. This is generally true for a large enough field size q [65]. The probability of linear independency is over 99.6% for $q = 2^8$, and it increases as q increases [51]. Thus, for $q = 2^8$, any selection of $(m-1) \times (m-1)$ submatrix $\boldsymbol{\beta}'_1$ and $(n(t)-(l-1)(m-1)) \times (n(t)-(l-1)(m-1))$ submatrix $\boldsymbol{\beta}'_2$ can form full rank matrixes with high probability.

The original data segments can be decoded using Gaussian Elimination [111], which corresponds to solve a system of linear equations with $m-1$ variables and a system of linear equations with $(n(t)-(l-1)m)$ variables in F_q. Then, we can obtain the following theorem.

Theorem 3.4. *In DSC-*m*LDC, for the case that each sensor node has two buffers and the mBS randomly queries $m-1$ sensor nodes, the success ratio of data collection is very close to 100% by using a large enough finite field size q for coefficients.*

Proof. Whenever the mBS performs data collection, the set of original data segments encoded in the two latest combined segments in each sensor node includes all the m latest original data segments, as shown in Lemma 3.1. In the decoding process, by querying any $m-1$ sensor nodes, the mBS collects $2(m-1)$ latest combined segments and the corresponding coefficient vectors. The key property required for successful decoding is that the coefficient vectors are linearly independent. Therefore, the success ratio of data collection in DSC-mLDC mainly depends on the probability of linear independence for the coefficient vectors. The probability of linear independency for the coefficient vectors is over 99.6% for $q = 2^8$, and it increases as q increases [51]. Thus, the success ratio of data collection is over 99.6% for $q = 2^8$, and it increases as q increases. $\square$

3.3 DSC-mLDC for the case that each sensor node has more than 2 buffers

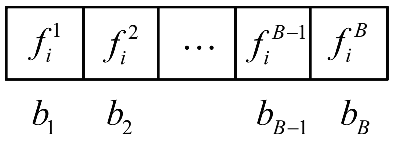

Figure 3.5: B combined segments are stored in the B buffers of sensor node i.

We now present DSC-mLDC for the case that each sensor node has B $(B > 2)$ buffers. The B buffers are denoted by $b_1, b_2, ..., b_B$. Let $F_i = \{f_i^1, f_i^2, ..., f_i^B\}$ be a set of the combined segments stored in sensor node i, where f_i^u is stored in buffer b_u, $u = 1, ..., B$, as shown in Fig 4.2. The associated coefficient vector $\overrightarrow{\beta_i^u}$ is also stored in buffer b_u of sensor node i.

3.3.1 Data encoding and replacement

Similar to DSC-mLDC for the case that each sensor node has two buffers, each sensor node separately encodes a certain number of original data segments in a combined segment. Consider that the certain number of original data segments encoded in a combined segment is x $(x < m)$. We show later that how to set this value x to ensure that the set of original data segments encoded in the B combined segments includes all the m latest original data segments. Similarly, let $f(r)$ be a combined segment which encodes the r^{th} recorded x original data segments in sensor node i. Generally, we have

$$f(r) = \sum_{j=(r-1)x+1}^{rx} \beta_{ij} c_j. \tag{3.17}$$

When a new combined segment $f(r)$ is formed, $f(r)$ and its associated coefficient are stored in the corresponding buffer of sensor node i. If $f(r)$ is stored in buffer b_u of sensor node i, we have $f_i^u = f(r)$, $u = 1, 2, ..., B$. A new combined segment encodes the latest original data segments. If there has been a combined segment stored in the corresponding buffer, the new combined segment including the associated coefficient vector will replace the old ones.

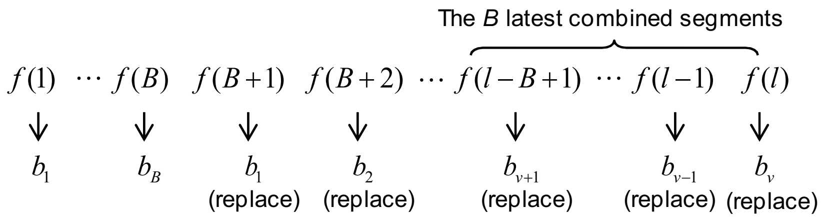

FIGURE 3.6: Data replacement to store the B latest combined segments.

In DSC-mLDC for the case that each sensor node has B buffers, the data replacement is performed after a sensor node has stored B combined segments $f(1)$, $f(2)$, ..., $f(B)$. As an example shown in Fig 3.6, at first $f(1)$, $f(2)$, ..., $f(B)$ are stored in buffer b_1, b_1, ..., b_B of sensor node i, respectively. That is, $f_i^u = f(u)$, $u = 1, 2, ..., B$. When $f(B+1)$ is formed, $f(B+1)$ replaces $f(1)$ to be stored in buffer b_1, i.e., $f_i^1 = f(B+1)$. And when $f(B+2)$ is formed, $f(B+2)$ replaces $f(2)$ to be stored in buffer b_2, i.e., $f_i^2 = f(B+2)$. With the data replacement, each sensor node stores the B latest combined segments $f(l-B+1)$, $f(l-B+2)$,..., $f(l)$. In this example, $f(l)$ is stored in buffer v of sensor node i, $v \in \{1, ..., B\}$. The replacement of the combined segments are performed with the replacement of the associated coefficient vectors. Relation (3.18) shows how to store the combined segment $f(r)$ (including the associated coefficient vector) in the corresponding buffer of sensor node i.

$$\begin{cases} f_i^u = f(r), & r \bmod B = u, \\ f_i^B = f(r), & r \bmod B = 0. \end{cases} \tag{3.18}$$

Note that $f(l)$ is the latest combined segment formed before the mBS performs data collection. The number of original data segments encoded in $f(l)$ depends on the total number of original data segments $n(t)$. If $n(t) = lx$, the number of original data segments encoded in $f(l)$ is x. And if $n(t) = (l-1)x+1$, the number of original data segments encoded in $f(l)$ is 1. Similar to that we showed in DSC-mLDC for the case that each sensor node has two buffers, $n(t)$ cannot be larger than lx or less than $(l-1)x+1$, otherwise $f(l)$ is not the latest combined segment. That is, when each sensor node has B buffers, $f(l)$ is the latest combined segment on condition that

$$(l-1)x + 1 \leq n(t) \leq lx. \tag{3.19}$$

From equation (3.19), the number of original data segments encoded in the latest combined segment $f(l)$ is with an upper bound x and a lower bound 1. The number of original data segments encoded in each of the other $B-1$ combined segments is x. Therefore, in each sensor node, the number of original data segments encoded in the B latest combined segments is with an upper bound Bx and a lower bound $(B-1)x+1$.

We give a formal description of data encoding and replacement algorithm of DSC-mLDC when each sensor node has more than two buffers. Every time an original data segment c_j is encoded with a combined data segment. The data encoding algorithm (Algorithm 2) is locally executed at each sensor node.

Algorithm 2: Data encoding when each sensor node has more than two buffers

```
Input: Original data segment $c_j$, number of latest data segments $m$, buffer size $B$,
       value of $x$.
Output: A set of combined segments $F_i = \{f_i^1, f_i^2, ..., f_i^B\}$.
1  for $j = 1$ to $n(t)$ do
2      for $u = 1$ to $B$ do
3          Let $r = \lceil j/x \rceil$ ;
4          Randomly generate $\beta_{ij}$ from $F_q$ ;
5          if $r \mod B = u$ then
6              if $C(f_i^u) < x$ then
7                  $f_i^u = f_i^u + \beta_{ij} c_j$;
8              end
9              else
10                 $f_i^u = \beta_{ij} c_j$
11             end
12         end
13         else
14             if $C(f_i^B) < x$ then
15                 $f_i^B = f_i^B + \beta_{ij} c_j$;
16             end
17             else
18                 $f_i^B = \beta_{ij} c_j$
19             end
20         end
21     end
22 end
```

We show how to set the value x to ensure that the set of original data segments encoded in the B combined segments includes all the m latest original data segments. $C(f_i^u)$ is the number of original data segments encoded in the combined segment f_i^u. The number of original data segments encoded in the B combined segments $f_i^1, f_i^2, ..., f_i^B$ equals to $\sum_{j=1}^{B} C(f_i^j)$. Note that this number satisfies

$$(B-1)x + 1 \leq \sum_{j=1}^{B} C(f_i^j) \leq Bx. \tag{3.20}$$

To ensure that the set of original data segments encoded in the B combined segments includes all the m latest original data segments, $\sum_{j=1}^{B} C(f_i^j)$ should satisfy

$$\sum_{j=1}^{B} C(f_i^j) \geq m. \tag{3.21}$$

From equation (3.20), we can obtain the sufficient condition of equation (3.21) as

$$(B-1)x + 1 \geq m. \tag{3.22}$$

Then,

$$x \geq \frac{m-1}{B-1}. \tag{3.23}$$

From equation (3.23), we obtain the minimum value of x as

$$x = \begin{cases} \frac{m-1}{B-1}, & m > 1, \\ 1, & m = 1. \end{cases} \tag{3.24}$$

We will show that the minimum value x in equation (3.24) is also the optimal value in the decoding process.

3.3.2 Data Decoding

The mBS first collects the data and then performs the data decoding. Note that there are B combined segments stored in each sensor node. A sensor node queried by the mBS will upload all the B combined segments and the associated coefficient vectors to the mBS. In DSC-mLDC, with the data replacement, each sensor node stores the B latest combined segments $f(l-B+1)$, $f(l-B+2)$, ..., $f(l)$. The replaced combined segments are $f(1)$, ..., $f(l-B)$. In each replaced combined segment, the number of encoded original data segments is x. Thus, the total number of original data segments encoded in the $l-B$ replaced combined segments is $(l-B)x$. The set of original data segments encoded in the $(l-B)x$ replaced combined segments is $\{c_1, c_2, ..., c_{(l-B)x}\}$. And the set of original data segments encoded in the B latest combined segments $f(l-B+1)$, $f(l-B+2)$,..., $f(l)$ is $\{c_{(l-B)x+1}, c_{(l-2)(m-1)+2}, ..., c_{n(t)}\}$. We have

$$f(r) = \begin{cases} \sum\limits_{j=(r-1)x+1}^{rx} \beta_{ij} c_j, & r = l-B+1, l-B+2, ..., l-1, \\ \\ \sum\limits_{j=(l-1)x+1}^{n(t)} \beta_{ij} c_j, & r = l. \end{cases} \tag{3.25}$$

Note that the number of original data segments encoded in each of the $B-1$ combined segments $f(l-B+1)$, ..., $f(l-1)$ is x. The number of original data segments encoded in $f(l)$ is $n(t)-(l-1)x$, where $n(t)-(l-1)x \leq x$. In the encoding process, we have shown that the number of the original data segments encoded in the B latest combined segments is with an upper bound Bx and a lower bound $(B-1)x+1$. The following relation holds.

$$(B-1)x + 1 \leq n(t) - (l-B)x \leq Bx. \tag{3.26}$$

Then,

$$\frac{n(t)}{x} \leq l \leq \frac{n(t) - x - 1}{x}. \tag{3.27}$$

Without loss of generality, we consider that $f_i^v = f(l)$, where $v \in \{1, ..., B\}$. That is, $f(l)$ is stored in buffer v of sensor node i. From the data replacement process in DSC-mLDC, $f_i^{v+1} = f(l-B+1)$ and $f_i^{v-1} = f(l-1)$. That is, $f(l-B+1)$ is stored in buffer b_{v+1} of sensor node i and $f(l-1)$ is stored in buffer b_{v-1} of sensor node i

(as shown in Fig 3.6). The B latest combined segments $f(l-B+1)$, $f(l-B+2)$, ..., $f(l)$ are stored in the corresponding buffers of sensor node i, as shown in equation (3.28).

$$f_i^u = \begin{cases} f(l-v+u), & u = 1, 2, ..., v-1, \\ f(l), & u = v, \\ f(l-B-v+u), & u = v+1, v+2, ..., B. \end{cases} \tag{3.28}$$

For convenience, let

$$\mathbf{f}_u = (f_1^u, f_2^u, \ldots, f_N^u)^T, \quad u = 1, ..., B, \tag{3.29}$$

where f_i^u is the combined segment stored in buffer b_u of sensor node i, $i = 1, ..., N$. $\mathbf{f}_u$ includes the combined segments stored in buffer b_u of N sensor nodes. Let

$$\boldsymbol{\beta}_\mathbf{u} = \begin{pmatrix} \vec{\beta}_1^u \\ \vdots \\ \vec{\beta}_N^u \end{pmatrix} = \begin{cases} \begin{pmatrix} \beta_{1.(l-v+u-1)x+1} & \ldots \beta_{1.(l-v+u)x} \\ \vdots & \ddots \quad \vdots \\ \beta_{N.(l-v+u-1)x+1} & \ldots \beta_{N.(l-v+u)x} \end{pmatrix}, & u = 1, 2, ..., v-1, \\ \\ \begin{pmatrix} \beta_{1.(l-1)x+1} & \ldots \beta_{1.n(t)} \\ \vdots & \ddots \quad \vdots \\ \beta_{N.(l-1)x+1} & \ldots \beta_{N.n(t)} \end{pmatrix}, & u = v, \\ \\ \begin{pmatrix} \beta_{1.(l-B-v+u-1)x+1} & \ldots \beta_{1.(l-B-v+u)x} \\ \vdots & \ddots \quad \vdots \\ \beta_{N.(l-B-v+u-1)x+1} & \ldots \beta_{N.(l-B-v+u)x} \end{pmatrix}, & u = v+1, v+2, ..., B, \end{cases} \tag{3.30}$$

and

$$\mathbf{c}_u = \begin{cases} \left(c_{(l-v+u-1)x+1}, ..., c_{(l-v+u)x}\right)^T, & u = 1, 2, ..., v-1, \\ \left(c_{(l-1)x+1}, ..., c_{n(t)}\right)^T, & u = v, \\ \left(c_{(l-B-v+u-1)x+1}, ..., c_{(l-B-v+u)x}\right)^T, & u = v+1, v+2, ..., B. \end{cases} \tag{3.31}$$

where $\vec{\beta}_i^u$ is the associated coefficient vector of f_i^u, and $\mathbf{c}_u$ is the set of original data segments encoded in f_i^u, $u = 1, ..., B$, $i = 1, ..., N$.

From equations (3.25) and (3.28), we have

$$\mathbf{f}_u = \boldsymbol{\beta}_u \mathbf{c}_u, \quad u = 1, ..., B. \tag{3.32}$$

We show that the mBS can decode the m latest original data segments by querying any x sensor nodes with high probability. Similar to DSC-mLDC for the case that each sensor node has two buffers, to decode the original data segments, the mBS must invert $x \times x$ submatrixs $\boldsymbol{\beta}'_u$ of $\boldsymbol{\beta}_u$ ($u = 1, ..., v-1, v+1, ..., B$) and invert a $(n(t) - (h+B-1)x) \times (n(t) - (h+B-1)x)$ submatrix $\boldsymbol{\beta}'_v$ of $\boldsymbol{\beta}_v$. The key property required for successfully decoding is that the coefficient vectors in any of the inverted submatrix must be linearly independent. This is generally true for a large enough field size q [65].

The original data segments can be decoded using Gaussian Elimination [111], which corresponds to solve $B-1$ systems of linear equations with x variables and a system of linear equations with $(n(t) - (h+B-1)x)$ variables in F_q. Since the set of original data segments encoded in the B latest combined segments includes all the m latest original data segments, we can obtain the following Theorem.

Theorem 3.5. *In DSC-*m*LDC, for the case that each sensor node has more than two buffers and the mBS randomly queries x sensor nodes, the success ratio of data collection is very close to 100% by using a large enough finite field size q for coefficients.*

Note that the value x equals to the number of sensor nodes that should be queried by the mBS during data collection. The minimum value of x is the optimal value of x, since the sensor network will consume less energy if less sensor nodes need to upload data to the mBS. Thus, equation (3.24) is the optimal value of x. From equation (3.24), the number of sensor nodes that should be queried by the mBS can be reduced with a few additional storage space in each sensor node. We will show latter that this can also result in reducing the transmission cost for data submission to the mBS.

3.4 Performance analysis and comparison

3.4.1 Computation, transmission and storage overheads in DSC-mLDC

Since the sensor nodes are power constrained entities, the applied scheme must be light-weighted. The proposed DSC-mLDC scheme is with low computation, storage and communication overheads.

In the encoding process of DSC-mLDC, each sensor node separately encodes a certain number of original data segments in a combined segment. The computation complexity for data encoding in each sensor node is a linear function, which depends on the number of encoded original data segments. The computation complexity for data encoding in DSC-mLDC is similar to that in PNC [51]. The computation overhead for data coding in DSC-mLDC lies mainly in the decoding process. This is performed in the powerful mBS. In the decoding process, the mBS solves B systems of linear equations. The order of coefficient matrix for each system of linear equations is at most x $(< m)$. The original data segments encoded in the B combined segments in a sensor node (i.e., the variables in the B systems of linear equations) can be decoded in $O(x^3)$ by using the Gaussian Elimination [111]. The computation complexity for data decoding in DSC-mLDC is also similar to that in PNC [51].

In DSC-mLDC, a sensor node uploads B combined segments and the associated coefficient vectors to the mBS when it is queried by the mBS. The mBS queries x sensor nodes during data collection. From equation (3.24), the total number of combined segments uploaded from the queried sensor nodes is $\frac{B(m-1)}{B-1}$ $(B \geq 2)$, which decreases as the buffer size B increases. Thus, the transmission overhead for transmitting combined segments to the mBS can be reduced with a few additional storage space in each sensor node. Besides the combined segments, the coefficient vectors are also uploaded from the queried sensor nodes. The overhead of uploading the coefficient vectors is much lower than the overhead of uploading large size of combined combined segments. We will further show the benefits of the proposed DSC-mLDC scheme which overcomes the overhead in the simulations.

In DSC-mLDC, each sensor node stores B $(B \geq 2)$ combined segments and the B associated coefficient vectors. The storage overhead depends on the size of combined segments, the size of finite field for coefficients q, and how many original data segments are encoded in a combined segment. The storage overhead can be reduced a lot by coding, since each sensor node stores the combined segments and the associated coefficient vectors instead of storing the original data segments. We have shown the storage overhead for a combined segment with the associated coefficient vector, and the percentage of storage overhead reduction be coding in Section 2.2.1.

3.4.2 Performance comparison

Note that DEC [58] and PNC [51] also can be applied for data collection in WSNs. To show the superiority of the proposed DSC-mLDC scheme, we give an example to show the differences among DEC, PNC and the proposed DSC-mLDC scheme. Consider that there are 4 sensor nodes S_1, S_2, S_3 and S_4. Each sensor node has two buffers. The original data segments are c_1, c_2, $\cdots$, c_{10}. In this example, let $m = 3$,

$$
\begin{aligned}
S_1 &: f_1^1 = [c_1, c_2, \ldots, c_{10}], f_1^2 = [c_1, c_2, \ldots, c_{10}] \\
S_2 &: f_2^1 = [c_1, c_2, \ldots, c_{10}], f_2^2 = [c_1, c_2, \ldots, c_{10}] \\
S_3 &: f_3^1 = [c_1, c_2, \ldots, c_{10}], f_3^2 = [c_1, c_2, \ldots, c_{10}] \\
S_4 &: f_4^1 = [c_1, c_2, \ldots, c_{10}], f_4^2 = [c_1, c_2, \ldots, c_{10}]
\end{aligned}
$$

(a) DEC

$$
\begin{aligned}
S_1 &: f_1^1 = [c_{10}], f_1^2 = [c_9, c_{10}] \\
S_2 &: f_2^1 = [c_9, c_{10}], f_2^2 = [c_{10}] \\
S_3 &: f_3^1 = [c_8, c_9, c_{10}], f_3^2 = [c_9, c_{10}] \\
S_4 &: f_4^1 = [c_9, c_{10}], f_4^2 = [c_{10}]
\end{aligned}
$$

(b) PNC

$$
\begin{aligned}
S_1 &: f_1^1 = [c_7, c_8], f_1^2 = [c_9, c_{10}] \\
S_2 &: f_2^1 = [c_7, c_8], f_2^2 = [c_9, c_{10}] \\
S_3 &: f_3^1 = [c_7, c_8], f_3^2 = [c_9, c_{10}] \\
S_4 &: f_4^1 = [c_7, c_8], f_4^2 = [c_9, c_{10}]
\end{aligned}
$$

(c) DSC-mLDC

FIGURE 3.7: Data distribution in 4 sensor nodes (S_1 through S_4) with $B = 2$, $m = 3$. (a) DEC, (b) PNC, (c) DSC-mLDC.

i.e., the mBS wants to collect the 3 latest original data segments c_8, c_9, and c_{10}. The associated coefficients are omitted for ease of exposition, as equation (3.4).

In DEC, each sensor node encodes all the recorded data segments in a combined segment without replacement, as shown in Fig 3.7 (a). It is clear that DEC is not suitable for continuous data collection, since too much data segments encoded in each combined segment will be undecodable in the decoding process. In PNC, each combined segment encodes only the part of latest original data segments by removing the older data segments. The number of data segments encoded in a combined segment varies from 1 to m. The sensor nodes queried by the mBS will upload a combined segment which encodes the maximum number of original data segments. Consider the example in Fig 3.7 (b). If the mBS queries S_1, S_2 and S_4, the combined segments f_1^2, f_2^1 and f_4^1 are uploaded to the mBS. The mBS cannot decode c_8, since none of the three uploaded combined segments encode c_8. Thus, PNC does not always decode the m latest original data segments completely if the mBS randomly queries some sensor nodes. The proposed DSC-mLDC scheme encodes all the m latest original data segments in the two combined data segments in each sensor node, as shown in Fig 3.7 (c). And, the mBS can decode all the m latest data segments if the mBS queries any $m-1$ sensor nodes in DSC-mLDC.

We further analyze the performance of DSC-mLDC by comparing it with PNC [51], since PNC also can support data replacement for continuous data collection. We compare the two schemes in the following scenarios. (1) The minimum buffer size of each sensor node. (2) The minimum number of sensor nodes queried by the mBS. (3) The number of upload combined data segments. (4) Success ratio of data collection.

We fist compare the two schemes when each sensor node has minimum buffer size.

In PNC, the minimum buffer size of each sensor node is 1. In the data collection process of PNC, the mBS randomly queries m sensor nodes. Each of the queried sensor nodes uploads one combined segment. In PNC, the total number of combined segments which are uploaded from the m queried sensor nodes is m. However, PNC suffers from low success ratio of data collection. In PNC, when the mBS randomly queries m sensor nodes, the success ratio of collecting the m latest original data segments is less than 20% [51].

In DSC-mLDC, the minimum buffer size of each sensor node is 2. In the data collection process of DSC-mLDC, the mBS randomly queries $m-1$ sensor nodes. Each of the queried sensor nodes uploads two combined segment. In DSC-mLDC, the total number of combined segments which are uploaded from the $m-1$ queried sensor nodes is $2(m-1)$. Although this number seems to be larger than that in PNC, to successfully collect the m latest original data segments, this number could be much less than that in PNC. Because the success ratio of data collection in DSC-mLDC is much higher than that in PNC. For example, to successfully collect the

m latest original data segments, the mBS queries $m-1$ sensor nodes once in DSC-mLDC, while the mBS queries m sensor nodes five times in PNC. The total number of uploaded combined segments is $2(m-1)$ in DSC-mLDC and $5m$ in PNC.

For the sake of convenience, a summary of the comparisons when each sensor node has minimum buffer size are given in Table 3.1.

TABLE 3.1: Comparisons of DSC-mLDC with PNC When Each Sensor Node Has Minimum Buffer Size

Comparison	**PNC**	**DSC-mLDC**
Minimum buffer size of each sensor node	1	2
Minimum number of queried sensor nodes	m	$m-1$
Number of uploaded combined segments from a sensor node	1	2
Total number of upload combined segments	m	$2(m-1)$
Success ratio of data collection	Low	High

We then compare DSC-mLDC with PNC when both of the two schemes achieve high success ratio of data collection.

In PNC, the success ratio of data collection can be improved by extending the buffer size of each sensor node to $\sqrt{m}+1$. The number of queried sensor nodes also should be extended to $m+\sqrt{m}$. In the data collection process of PNC, the mBS randomly queries $m+\sqrt{m}$ sensor nodes. Each of the queried sensor nodes uploads one combined segment which encodes the maximum number of original data segments. In PNC, the total number of uploaded combined segments is $m+\sqrt{m}$.

In DSC-mLDC, the number of queried sensor nodes can be reduced to $\frac{m-1}{B-1}$ by extending the buffer size of each sensor node to B, while maintaining a high success ratio of data collection which is very close to 100%. In the data collection process of DSC-mLDC, the mBS randomly queries $\frac{m-1}{B-1}$ sensor nodes. Each of the queried sensor nodes uploads B combined segments. In DSC-mLDC, the total number of uploaded combined segments is $\frac{B(m-1)}{B-1}$. Notice that this value decreases as the value of B increases. When $B=\sqrt{m}+1$, $\frac{B(m-1)}{B-1} < m+\sqrt{m}$. That is, with the same buffer size in each sensor node, the total number of uploaded combined segments in DSC-mLDC is less than that in PNC. On the other hand, in DSC-mLDC the total combined segments are submitted from fewer sensor nodes, which can reduce the cost for establishing connections between the mBS and sensor nodes. For example, in DSC-mLDC twenty combined segments can be uploaded from four sensor nodes (each sensor node has five buffers and uploads five combined segments). While, in PNC twenty combined segments should be uploaded from twenty sensor nodes (each sensor node has $m + \sqrt{m}$ buffers but uploads only one combined segment).

For the sake of convenience, a summary of the comparisons when both of the two schemes achieve high success ratio are given in Table 3.2.

TABLE 3.2: Comparisons of DSC-mLDC with PNC When Achieving High Success Ratio of Data Collection

Comparison	**PNC**	**DSC-mLDC**
Minimum buffer size of each sensor node	$\sqrt{m} + 1$	2
Minimum number of queried sensor nodes	$m + \sqrt{m}$	$\frac{m-1}{B-1}$, $(B \geq 2)$
Number of uploaded combined segments from a sensor node	1	B
Total number of upload combined segments	$m + \sqrt{m}$	$\frac{B(m-1)}{B-1}$, $(B \geq 2)$

Note that to achieve high success ratio of data collection, in PNC the buffer size of each sensor node is $\sqrt{m}+1$, which depends on the number of latest original data segments m. This is inflexible, since the number of latest original data segments may be varied in different times of data collection (e.g., the mBS collects the 10 latest original data segments in the first time but collects the 15 latest original data segments in the second time). However, the buffer size of each sensor node cannot be changed after being deployed. In DSC-mLDC, the buffer size of each sensor node is independent of the number of latest original data segments m. The buffer size of each sensor node can be adjusted by changing the number of sensor nodes queried by the mBS. DSC-mLDC achieves high success ratio of data collection with a minimum buffer size two in each sensor node.

3.5 Performance evaluation

We evaluate the performance of the proposed DSC-mLDC scheme by comparing it with PNC [51] in simulations. We compare DSC-mLDC with PNC on success ratio of data collection and energy consumption for data transmission.

3.5.1 Performance Metrics

We deploy 1000 sensor nodes uniformly at random into a field of 300m $\times$ 300m. The distance between the sensor nodes and the mBS is much longer than the distance between the sensor nodes, as suggested in Lindsey and Raghavendra [112]. Without necessarily entering deep into the sensor field, the mBS can perform data collection. The time slot to generate an original data segment is one hour. The size of an original data segment is 1 KB.

To reduce the storage overhead and the transmission overhead of the coefficients, we use *Suli and Mayers method* [110] to generate coefficients. Each sensor node only needs to store a coefficient for a combined segment. When the mBS performs data collection, besides the combined segments, each of the queried sensor nodes only needs to upload a coefficient for a combined segment. The mBS wants to collect the m latest original data segments. It will perform data collection after a sensing time interval, in which the number of total generated data is randomly chosen from $[m, 4m]$. Since the value m can affect the performance of the applied schemes, in the simulation, we change the value m from 10 to 100. Unless otherwise specified, the size of finite field for coefficients is $q = 2^{16}$. The solution of linear equations is using the Gaussian Elimination [111].

Table 3.3 demonstrates part of the important parameters and settings in the simulations.

TABLE 3.3: System Parameters and Settings for m Latest Data Segment Collection

System Parameters	**Settings**
Simulator	Matlab
Length $\times$ Width	300 m $\times$300 m
Number of sensor nodes	1000
Transmit range between sensor nodes	20 m
Transmit range between sensor nodes and mBS	150 m $\sim$ 250 m
Size of a data segment	1KB
Energy consumption for sending a data segment	20 nAH
Size of finite field for coefficients q	2^{16}
Simulation times	1000
Confidence interval	95%

3.5.2 Simulation results

3.5.2.1 Comparison on success ratio of data collection

During data collection, the mBS first queries a small subset (i.e., the minimum number) of sensor nodes uniformly at random from the sensor network to collect data. Unless otherwise specified, in DSC-mLDC, the minimum number of queried sensor nodes is x, where $x = \lceil \frac{m-1}{B-1} \rceil$, as we have shown in equation (3.4). If the m latest original data segments cannot be decoded successfully, the mBS will query additional sensor nodes one by one until the m latest original data segments are decoded. The additional sensor nodes are queried uniformly at random from the sensor network (excluding the sensor nodes which have queried in the same time of data collection). The sensor network will consume more energy for sending data to the mBS if the mBS repeats data collection. Thus, a high success ratio of data collection is important.

We first compare DSC-mLDC with PNC on success ratio of data collection by varying the number of latest original data segments m. In PNC, the sensor nodes queried by the mBS will upload a combined segment which encodes the maximum number of original data segments. Fig 3.8 shows the success ratio of data collection as a function of the number of latest original data segments m. In this simulation,

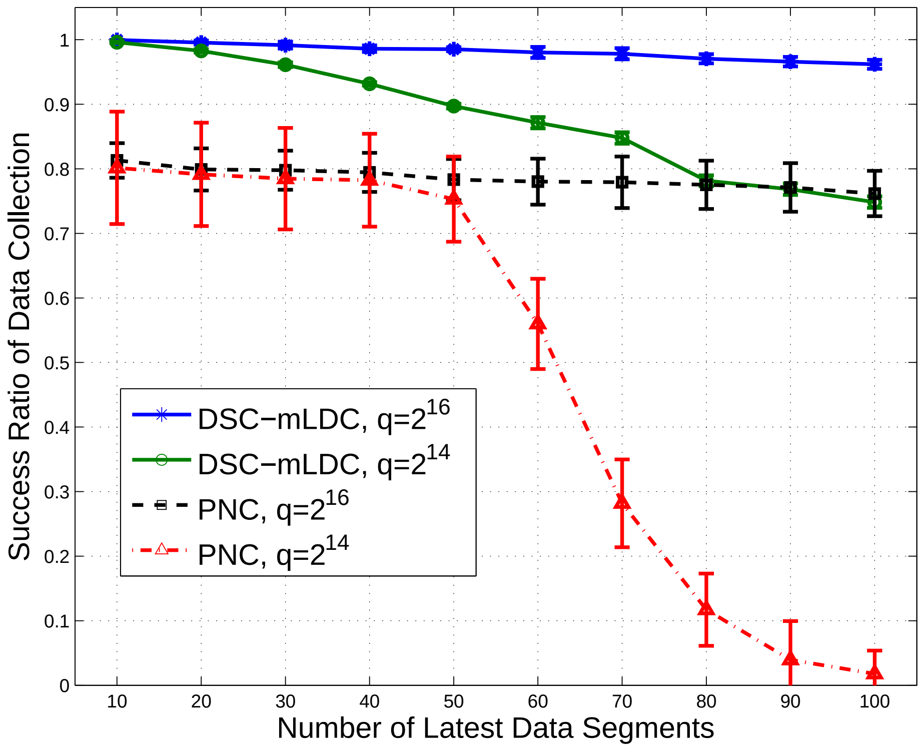

FIGURE 3.8: Success ratio of data collection vs. the number of latest original data segments.

the buffer size is set to $B = 2$. The size of finite field for coefficients is set to $q = 2^8$ and $q = 2^{16}$, respectively. It is clear that DSC-mLDC performs better than PNC no matter the finite field size $q = 2^8$ or $q = 2^{16}$. The success ratio of data collection in DSC-mLDC is close to 100% for a large size of finite field when $q = 2^{16}$. This fact is also stated in the proof of Theorem 3.4. The success ratios of data collection in PNC is lower than that in DSC-mLDC, since the set of original data segments encoded in the collected combined segments not always includes all the m latest original data segments. The success ratio of data collection in DSC-mLDC mainly depends on the probability of linear independence for the coefficient vectors. The success ratio of data collection in PNC not only depends on the probability of linear independence for the coefficient vectors, but also depends on the probability that the set of original data segments encoded in the collected combined segments includes all the m latest original data segments.

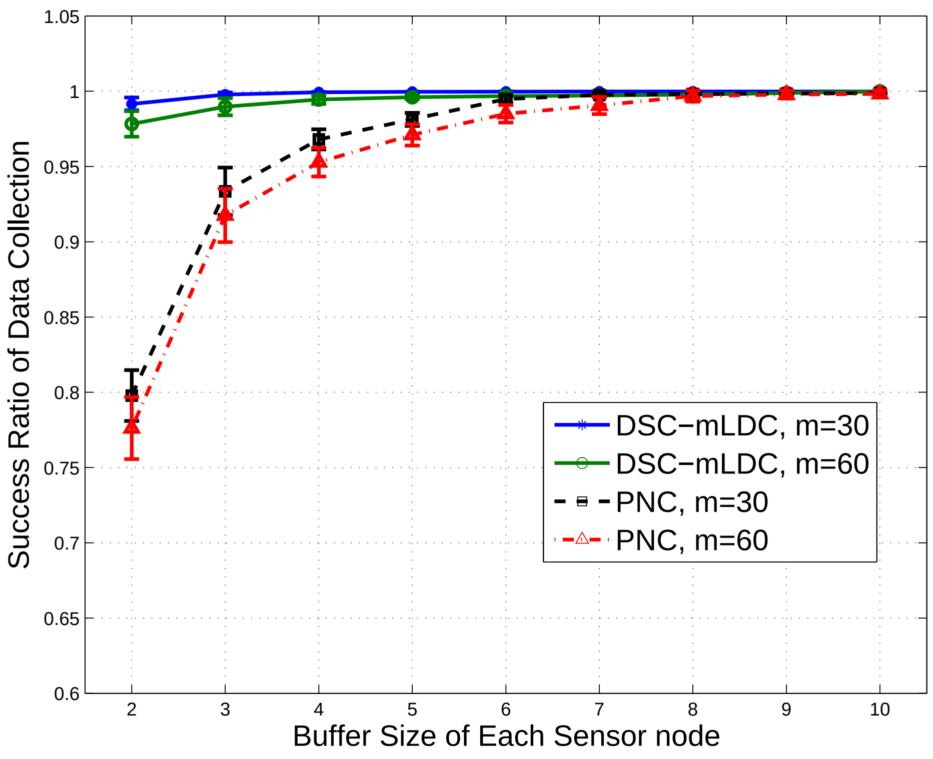

FIGURE 3.9: Success ratio of data collection vs. buffer size B in each sensor node.

Since the buffer size B of each sensor node can affect the performance of the applied scheme, we then compare DSC-mLDC with PNC on success ratio of data collection by varying the buffer size B of each sensor node. As shown in Fig 3.9, the success ratio of data collection in DSC-mLDC maintains high no matter the buffer size of each sensor node is small or big, which is close to 100%. The success ratio of data collection in PNC is not so high when each sensor node has smaller buffer size. The success ratio of data collection in both DSC-mLDC and PNC increase as the buffer size B increases. That is because in DSC-mLDC the number of data segments encoded in a combined segment decreases as the buffer size B increases. The probability of linear independence for the coefficient vectors increases as the number of data segments encoded in a combined segment decreases, as shown in Fig 2.6. In PNC, each sensor node can store more combined segments as the buffer size B increases. Thus, the probability that a combined segment encodes all the m latest original data segments in each senor node are improved.

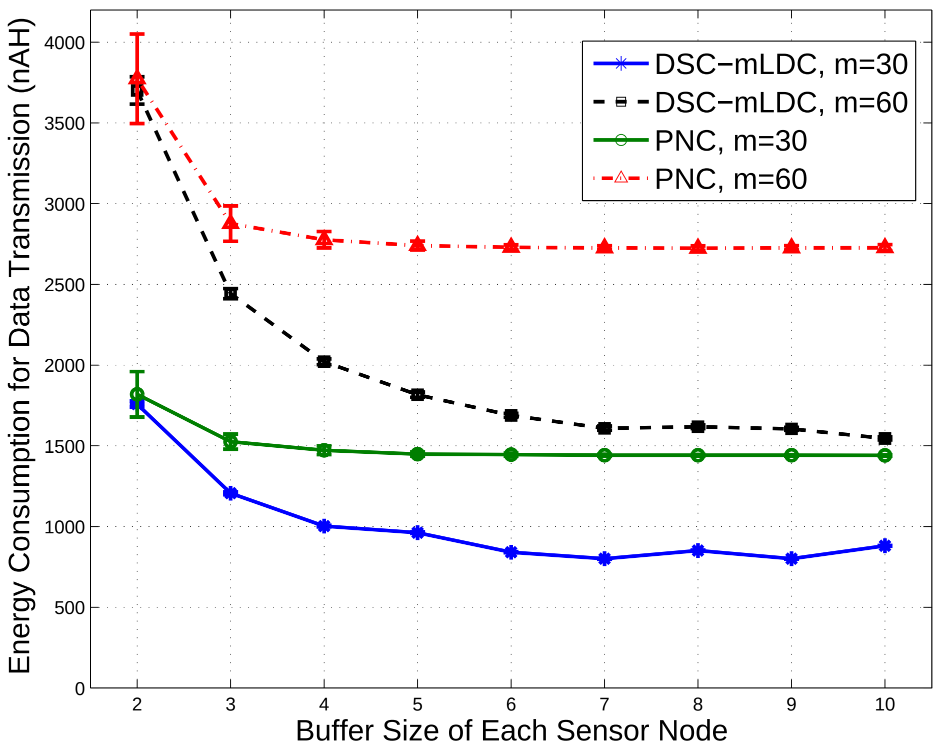

FIGURE 3.10: Energy consumption for data transmission vs. buffer size B in each sensor node.

3.5.2.2 Comparison on energy consumption for data transmission to the mBS

The success ratio of data collection in both DSC-mLDC and PNC can be improved with larger buffer size in each sensor node. We the compare DSC-mLDC with PNC on energy consumption for data transmission by varying the buffer size B of each sensor node. Since the energy consumption for data encoding and decoding in these two schemes are almost the same, for the sake of convenience we compare the two schemes on energy consumption for data transmission during data collection by the mBS. We use the energy model by Mainwaring et al [52]. Note that the same energy model is used in PNC too. The energy consumption for transmitting one combined segment (including the associated coefficient) is 20 nAH (10.9 Ampere hours). As suggested in [113], establishing a connection between a sensor node and the mBS consumes energy too. Consider that the energy consumption for establishing a connection between a sensor node and the mBS is 20 nAH.

In DSC-mLDC, a queried sensor node will upload B combined segments to the mBS. Thus, the energy consumption for a queried sensor node in DSC-mLDC is $(20 + 20B)$ nAH. In PNC, a queried sensor node will upload only one combined segment to the mBS. The energy consumption for a queried sensor node in PNC is 40 nAH. The mBS first queries the minimum number of sensor nodes to collect data. In DSC-mLDC, the minimum number of queried sensor nodes is $\lceil \frac{m-1}{B-1} \rceil$. While in PNC, the minimum number of queried sensor nodes is m. If the m latest original data segments cannot be decoded successfully, the mBS will query additional sensor nodes one by one until the m latest original data segments are decoded.

As shown in Fig 3.10, the energy consumption in DSC-mLDC is less than that in PNC no matter the number of latest original data segments $m = 30$ or $m = 60$. That is because the success ratio of data collection in DSC-mLDC is higher than that in PNC when each sensor node has smaller buffer size. When each sensor node has larger buffer size, the total number of uploaded combined segments in DSC-mLDC is less than that in PNC. We also notice that the energy consumption in DSC-mLDC decreases significantly as the buffer size B increases, since in DSC-mLDC the total number of uploaded combined segments and the number of queried sensor nodes decrease as the buffer size B increases. In PNC, the energy consumption only decreases significantly when buffer size increases from $B = 2$ to $B = 3$, since the success ratio of data collection improved a lot from $B = 2$ to $B = 3$. In PNC the total number of combined segments uploaded to the mBS remains the same when the buffer size B increases. Thus, the range for the reduction of energy consumption in PNC is very small when the buffer size B continuous to increase.

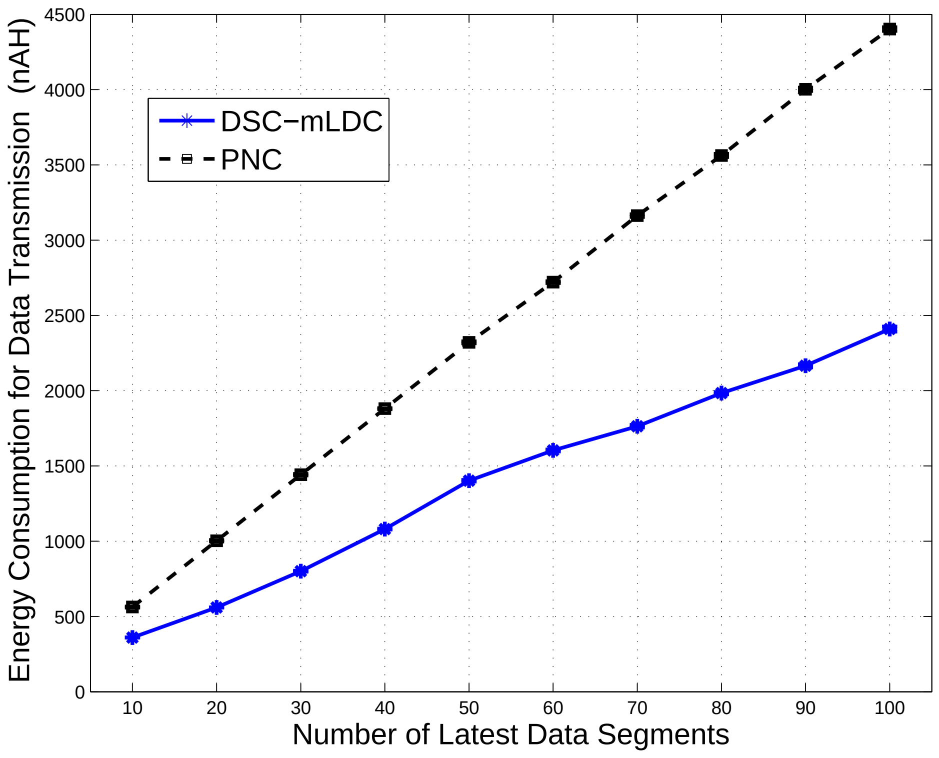

FIGURE 3.11: Energy consumption for data transmission vs. the number of latest original data segments.

The success ratio of data collection in PNC can be improved to be close to 100% by extending the buffer size of each sensor node to $\sqrt{m} + 1$. We then compare DSC-mLDC with PNC on energy consumption for data transmission when both the two schemes have large enough buffer size to achieve high success ratio of data collection (i.e., the success ratio of data collection is close to 100%). Fig 3.11 shows the energy consumption for data transmission as a function of the number of latest original data segments m. In the simulation, the buffer size of each sensor node is set to $B = \sqrt{m} + 1$, which is large enough for PNC to achieve high success ratio of data collection. In PNC, during data collection the mBS queries $m + \sqrt{m}$ sensor nodes to collect data. The sensor nodes queried by the mBS will upload a combined segment which encodes the maximum number of original data segments. As shown in Fig 3.11, the energy consumption for data transmission in DSC-mLDC is much less than that in PNC. That is because, the total number of uploaded combined segments and the number of queried sensor nodes in PNC are $\sqrt{m} + m$, which are more than that in DSC-mLDC with the same buffer size in each sensor node.

3.6 Discussion

In the proposed DSC-mLDC scheme, to focus on the data encoding and replacement processes in each sensor node and the data decoding process in the mBS, we assume that each original data segment is recorded by all the sensor nodes by using some existing data dissemination method. Actually, if each original data segment is not recorded by all the sensor nodes, the mBS still can decode the original data segments with high probability.

As mentioned in [58], if each original data segment is recorded by a certain number of sensor nodes, the original data segments can be successfully decoded with high probability. We have showed that in DSC-mLDC, for the x original data segments encoded in a combined segment, the key property for successfully decoding is that the coefficient vectors in any of the inverted $x \times x$ submatrix $\boldsymbol{\beta}'_u$ of $\boldsymbol{\beta}_u$ (as shown in equation (3.30)) must be linearly independent (i.e., $det(\boldsymbol{\beta}'_u) \neq 0$). According to the result in [58] (see Theorem 1 in [58]), in DSC-mLDC, if each original data segment is recorded by at least $d = \frac{5N}{x} \ln x$ sensor nodes, the original data segments can be successfully decoded with high probability. Specifically, we have $Pr[det(\boldsymbol{\beta}'_u) = 0] \leq \frac{x}{q} + o(1)$. Note that if each original data segment is recorded by less sensor nodes, the data dissemination overhead is lower.

Another way to reduce the data dissemination overhead in sensor network is using cluster structure with distributed clustering heads [114, 115]. The sensed data in each cluster are disseminated to its cluster head. The cluster heads may communicate with each other to exchange data. The cluster heads encode the received original data segments in some combined segments with different coefficient vectors, and send them to their cluster members. In such a way, the data dissemination overhead can be reduced significantly, science the sensed data are not directly disseminated to the whole network. Similar to that considered in [116], to balance the load on cluster heads, the cluster heads can be selected periodically according to a hybrid of the node residual energy and a secondary parameter, such as node proximity to its neighbors or node degree.

Note that in the real application, due to the latency caused by the data dissemination, some data segments may have not been recorded by some sensor nodes when the mBS performs data collection. For this reason, the combined segments stored in the same buffers of different sensor nodes may encode different original data segments. In this dissertation, we focus on the data encoding and replacement methods by neglecting the latency for data dissemination in the proposed DSC-mLDC scheme. In the future, we will consider how to deal with the problem of latency.

3.7 Summary

In this chapter, we consider the first scenario of continuous data collection (i.e., Latest data segment collection), and present a novel data collection scheme called Distributed Separate Coding for Latest Data segment Collection (DSC-mLDC). DSC-mLDC is not only shown as an efficient storage method to collect the m latest data segments continuously, but also achieves a high success ratio of data collection. We present the data encoding process, data replacement precess and data decoding process in DSC-mLDC. We show that in DSC-mLDC, the number of sensor nodes that should be queried by the mBS can be reduced with a few additional storage space in each sensor node, which result in reducing the energy consumption for data transmission to the mBS. We also show that the success ratio of data collection in DSC-mLDC is very close to 100% by using larger enough finite field for coefficients in both theoretical analysis and simulations. Furthermore, we evaluate the performance of the proposed DSC-mLDC scheme by comparing it with the existing PNC scheme. It is shown that DSC-mLDC outperforms the existing scheme significantly.

Chapter 4

Distributed Separate Coding for All Data Segment Collection

In this chapter, we focus on the issue of collecting all data continuously in wireless sensor networks with a mobile Base Station (mBS). To the best of our knowledge, in most of the related work the number of collected data segments is a fixed value. We consider to collect all the $n(t)$ data segments generated in a time interval t, where $n(t)$ is a nondecreasing function of time interval t. We present a novel data collection scheme called Distributed Separate Coding for All Data segment Collection (DSC-ADC). DSC-ADC is decentralized and based on the mBS's randomly accessing. By separately encoding a certain number of data segments in a combined segment, and storing the combined segments in the corresponding buffers of each sensor node, the DSC-ADC scheme provides an efficient storage method to collect all data segments with a high success ratio.

Before introducing the proposed DSC-ADC scheme for all data segment collection, we show the differences between m latest data segment collection in Chapter 3 and all data segment collection in this Chapter. In m latest data segment collection, the collected data is the part of latest data and the number equals to m. m is a fixed number. The value of m is known to each sensor node before the mBS performs data collection. The number of data segments encoded in a combined segment x is set according to the value of m. In all data segment collection, the number of collected data $n(t)$ is a variable. The value of $n(t)$ increases as the time interval t increases. The value of $n(t)$ is unknown until the mBS performs data collection. Thus, the data encoding method in all data segment collection is different from that in m lathe data segment collection, since the data cannot be encoded according to the value of $n(t)$. The value of $n(t)$ depends on the time when the mBS performs data collection. $n(t)$ may equal to m. But, that is only when the mBS performs data collection at the time that the total number of data segments generated in a time interval equals to m. However, the time when the mBS performs data collection cannot be controlled. In the extreme environment, many external factors will affect the arrival of the mBS (e.g., the bad weather).

This chapter is organized as follows. In Section 4.1, we give an overview of the proposed DSC-ADC scheme. In Section 4.2, we present DSC-ADC for the right arrival case. In Section 4.3, we present DSC-ADC for the late arrival case. Performance analysis and discussion are presented in Section 4.4. In Section 4.5, we evaluate the performance of the proposed scheme through simulations. Section 4.6 concludes this chapter.

4.1 An overview of DSC-ADC

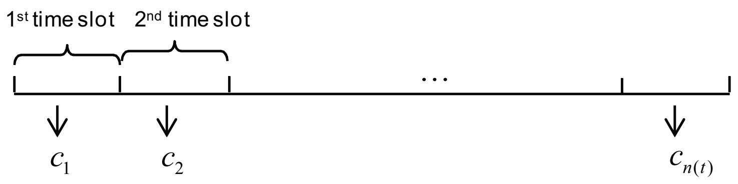

FIGURE 4.1: All the $n(t)$ data segments generated in a time interval t.

In the proposed DSC-ADC scheme, we consider to collect all the $n(t)$ data segments generated in a time interval t, as shown in Fig 4.1. In DSC-ADC, each sensor node separately encodes a certain number of original data segments in a combined segment and stores it in the corresponding buffer. Let $F_i = \{f_i^1, f_i^2, ..., f_i^B\}$ be a set of combined segments stored in sensor node i, where f_i^k is stored in buffer b_k, as shown in Fig 4.2. The associated coefficient vector for f_i^k is also stored in buffer b_k. $C(f_i^k)$ is the number of data segments encoded in f_i^k.

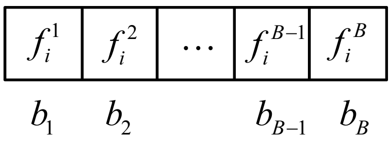

FIGURE 4.2: The combined segment f_i^k is stored in buffer b_k.

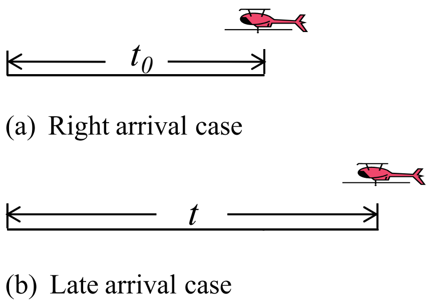

FIGURE 4.3: Two cases in all data segment collection. t_0 = minimum$\{t\}$.

We consider the following two cases: (1) *right arrival case* that the mBS arrives on time, (2) *late arrival case* that the mBS arrives late. In general condition, the mBS performs data collection in a regular time interval t_0, where t_0 = minimum$\{t\}$, as shown in Fig 4.3 (a). This case is called right arrival case. The total number of original data segments generated in time interval t_0 is $n(t_0)$. If the mBS arrives when the time interval $t \geq t_0$, we call this case late arrival case, as shown in Fig 4.3 (b). Note that $n(t_0) \leq n(t)$.

The encoding and decoding processes in the two cases are with some differences. The encoding and decoding processes in the right arrival case is easier, as the total number of original data segments generated in time interval t_0 is a fixed value $n(t_0)$. While in late arrival case, the total number of original data segments may increase as the time interval t ($\geq t_0$) increases. A challenge issue is how to let the fixed buffers in each sensor node to store all the data segments whether the mBS arrives on time or late. We first present the encoding and decoding processes for right arrival case. The enhancement to late arrival case will be presented later.

4.2 DSC-ADC for the right arrival case

4.2.1 Data encoding

Each sensor node will separately encode a certain number of original data segments in each combined segment, and store the B combined segments and the associated coefficient vectors in its B buffers. Assume that the certain number of data segments encoded in a combined segment is x ($x < n(t_0)$). For example, when the first x original data segments c_1, c_2,..., c_x are recorded, they are encoded in f_i^1 and stored in buffer b_1 by sensor node i. When the second x original data segments c_{x+1}, c_{x+2},..., c_{2x} are recorded, they are encoded in f_i^2 and stored in buffer b_2 by sensor node i. Fig 4.4 shows the data encoding process in sensor node i.

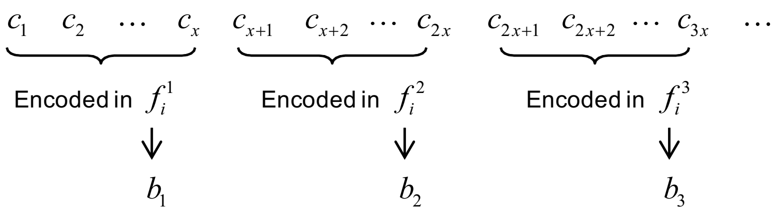
c_1 c_2 $\cdots$ c_x c_{x+1} c_{x+2} $\cdots$ c_{2x} c_{2x+1} c_{2x+2} $\cdots$ c_{3x} $\cdots$
Encoded in f_i^1
Encoded in f_i^2
Encoded in f_i^3
b_1
b_2
b_3

FIGURE 4.4: x original data segments are separately encoded in a combined segment and stored in the corresponding buffer.

We present the encoding equation of each combined segment f_i^k, $k = 1, ..., B$. Let r be a positive integer. If the sequence number of c_j satisfies $(k-1)x + 1 \leq j \leq kx$, c_j will be encoded in f_i^k as the following equation.

$$f_i^k = \sum_{j=(k-1)x+1}^{kx} \beta_{ij} c_j, \tag{4.1}$$

where $k = 1, ..., B$. The coefficients β_{ij} are selected uniformly and independently from a finite field F_q.

Since the total number of data segments $n(t_0)$ may not be an integer multiple of x, the number of data segments encoded in the last combined segment f_i^B may be less than x. Thus, equation (4.1) can be divided into two parts as follows.

$$f_i^k = \begin{cases} \sum\limits_{j=(k-1)x+1}^{kx} \beta_{ij} c_j, & k = 1, ..., B-1, \\ \\ \sum\limits_{j=(B-1)x+1}^{n(t_0)} \beta_{ij} c_j, & k = B. \end{cases} \tag{4.2}$$

For example, when $n(t_0) = 16$, $B = 3$ and $x = 6$, the original data segments c_1, c_2,..., c_6 are first encoded in f_i^1. The original data segments c_7, c_8,..., c_{12} are then encoded in f_i^2. The last 4 original data segments c_{13}, c_{14}, c_{15}, c_{16} are encoded in f_i^3.

The encoding process is performed by each sensor node. An original data segment c_j is encoded with one of the B combined segments in sensor node i. The data encoding algorithm (Algorithm 3) is locally executed at each sensor node.

We show how to set the value x, which is the maximum number of data segments encoded in a combined segment. Since the maximum number of data segments encoded in a combined segment is x, the total number of data segments encoded in the B combined segments is at most Bx. To guarantee that each sensor can encode all the $n(t_0)$ data segments, Bx should satisfy

$$Bx \geq n(t_0). \tag{4.3}$$

Algorithm 3: Data encoding in the right arrival case

```
Input: Original data segment $c_j$, total number $n(t_0)$, buffer size $B$.
Output: A set of combined segments $F_i = \{f_i^1, f_i^2, ..., f_i^B\}$.
for $j = 1$ to $n(t_0)$ do
    for $k = 1$ to $B$ do
        if $(k-1)x + 1 \leq j \leq kx$ then
            Randomly generate $\beta_{ij}$ from $F_q$ ;
            $f_i^k = f_i^k + \beta_{ij} c_j$;
        end
    end
end
```

Then,

$$B \geq n(t_0)/x. \tag{4.4}$$

Since B is an integer, from equation (4.4), we can obtain the minimum buffer size of each sensor node as

$$B = \lceil n(t_0)/x \rceil. \tag{4.5}$$

Thus, we have the following Lemma.

Lemma 4.1. *In the right arrival case, each sensor node with buffer size $B = \lceil n(t_0)/x \rceil$ is enough to store the combined segments which encode all the original data segments.*

4.2.2 Data decoding

When the mBS performs data collection, there are B combined segments stored in each sensor node. Each of the $B-1$ combined segments f_i^1, f_i^2,..., f_i^{B-1} encodes

x data segments, while f_i^B encodes $n(t_0) - (B-1)x$ ($\leq x$) data segments, as the encoding functions shown in equation (4.2).

We show that the mBS can reconstruct all the original data segments by querying any x sensor nodes with high probability. For the sake of convenience, assume that the x sensor nodes queried by the mBS are sensor node 1, sensor node 2,..., sensor node x. The mBS querying these x sensor nodes will collect x sets of combined segments $F_1, F_2, ..., F_x$, where $F_i = \{f_i^1, f_i^2, ..., f_i^B\}$ is the set of combined segments collected from sensor node i, $i = 1, ..., x$. The mBS also collects the related coefficient vectors of the combined segments. Let $\mathbf{f}_k = \{f_1^k, f_2^k, ..., f_x^k\}$ denote the set of combined segments collected from buffer k of the x sensor nodes, $k = 1, ..., B$. Decoding of $\mathbf{f}_k$ is to decode the original data segments from the combined segment which is stored in buffer k in a sensor node. The decoding process includes B stages, decoding $\mathbf{f}_1$, $\mathbf{f}_2$,..., $\mathbf{f}_B$.

In $\mathbf{f}_1$, $\mathbf{f}_2$,...,$\mathbf{f}_{B-1}$, each of the combined segments encodes x original data segments. In the system of linear equations for $\mathbf{f}_k$ where $k = 1, ..., B-1$, the set of original data segments $\{c_{(k-1)x+1}, c_{(k-1)x+2}, ..., c_{(k-1)x+x}\}$ is the solution. $\{c_{(k-1)x+1}, c_{(k-1)x+2}, ..., c_{(k-1)x+x}\}$ is the set of original data segments encoded in the combined segment which is stored in buffer k of each sensor node.

We first show the process of decoding the set of x original data segments $\{c_1, c_2, ..., c_x\}$ from $\mathbf{f}_1$.

From equation (4.2), we know that

$$\mathbf{f}_1 = \mathbf{A}_1 \mathbf{c}_1, \tag{4.6}$$

where $\mathbf{c}_1 = (c_1, c_2, ..., c_x)^T$, and the expansion of equation (4.6) is as follows.

$$\begin{pmatrix} f_1^1 \\ f_2^1 \\ \vdots \\ f_x^1 \end{pmatrix} = \begin{pmatrix} \beta_{11} & \beta_{12} & \dots \beta_{1x} \\ \beta_{21} & \beta_{22} & \dots \beta_{2x} \\ \vdots & \vdots & \ddots \vdots \\ \beta_{x1} & \beta_{x2} & \dots \beta_{xx} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_x \end{pmatrix}. \tag{4.7}$$

The key property required for successfully decoding $\mathbf{f}_1$ is that the coefficient matrix $\mathbf{A}_1$ forms a full rank matrix with high probability. A necessary condition is that the coefficient vectors in $\mathbf{A}_1$ must be linearly independent. This is generally true for a large enough field size q [65]. The probability of linear independency is over 99.6% for $q = 2^8$ [51]. The original data segments can be decoded using Gaussian Elimination [111].

The decoding processes of $\mathbf{f}_2$,...,$\mathbf{f}_{B-1}$ are similar to the decoding process of $\mathbf{f}_1$. The key property required for successfully decoding the set of x original data segments from each $\mathbf{f}_k$ (k=1,...,B-1) is that the coefficient vectors in each coefficient matrix must be linearly independent. The coefficient vectors for the combined segments in $\mathbf{f}_k$ $(k = 1, ..., B-1)$ which form a $x \times x$ coefficient matrix $\mathbf{A}_k$ are as follows.

$$\mathbf{A_k} = \begin{pmatrix} \beta_{1.(k-1)x+1} & \beta_{2.(k-1)x+1} & \dots \beta_{x.(k-1)x+1} \\ \beta_{1.(k-1)x+2} & \beta_{2.(k-1)x+2} & \dots \beta_{x.(k-1)x+2} \\ \vdots & \vdots & \ddots \quad \vdots \\ \beta_{1.kx} & \beta_{2.kx} & \dots \beta_{x.kx} \end{pmatrix}, k = 1, ..., B-1. \tag{4.8}$$

In $\mathbf{f}_B$, each of the combined segments encodes $n(t_0) - (B-1)x$ original data segments. In the system of linear equations for $\mathbf{f}_B$, the set of original data segments $\{c_{(B-1)x+1}, c_{(B-1)x+2},$
$..., c_{n(t_0)}\}$ is the solution. We then show the process of decoding the set of $n(t_0) - (B-1)x$ original data segments $\mathbf{c}_B$ from $\mathbf{f}_B$.

From equation (4.2), we know that

$$\mathbf{f}_B = \mathbf{A}_B \mathbf{c}_B, \tag{4.9}$$

where $\mathbf{c}_B = (c_{(B-1)x+1}, c_{(B-1)x+2}, ..., c_{n(t_0)})^T$, and the expansion of equation (4.9) is as follows.

$$\begin{pmatrix} f_1^B \\ f_2^B \\ \vdots \\ f_x^B \end{pmatrix} = \begin{pmatrix} \beta_{1.(B-1)x+1} & \beta_{1.(B-1)x+2} & \dots \beta_{1.n(t_0)} \\ \beta_{2.(B-1)x+1} & \beta_{2.(B-1)x+2} & \dots \beta_{2.n(t_0)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{x.(B-1)x+1} & \beta_{x.(B-1)x+2} & \dots \beta_{x.n(t_0)} \end{pmatrix} \begin{pmatrix} c_{(B-1)x+1} \\ c_{(B-1)x+2} \\ \vdots \\ c_{n(t_0)} \end{pmatrix}. \tag{4.10}$$

The key property required for successful decoding of $\mathbf{f}_B$ is that the coefficient vectors in $\mathbf{A}_B$ must be linearly independent. This is generally true for a large enough field size q [65]. The original data segments can be decoded using Gaussian Elimination [111]. After decoding all the linear equations, the mBS can obtain all the original data segments $\mathbf{c} = \bigcup_{k=1}^{B} \mathbf{c}_k$.

We have shown that the mBS can decode all the original data segments by querying any x sensor nodes. From equation (4.5), we know that B is inversely proportional to x. The number of sensor nodes that should be queried by the mBS can be reduced with a few additional storage space in each sensor node. On the other hand, the mBS can query more sensor nodes to make the buffer size be available in a sensor node.

4.3 DSC-ADC for the late arrival case

We have shown that the combined segments stored in each sensor node with buffer size $B = \lceil n(t_0)/x \rceil$ can encode all original data segments. But there is still one problem. If the mBS arrives late, the sensor nodes do not know in which time and how long the mBS will delay, they just separately encode x data segments in a combined segment and store it in the corresponding buffer. If the total number of original data segments $n(t) \leq Bx$, by continuing to encode the original data segments in the last combined segment f_i^B, the buffers are till enough. But if $n(t) > Bx$, the data segments will exceed the total storage space of the sensor nodes if they still combine x original data segments in a combined fashion. Then we will give a formal description of the encoding and storage process in the late arrival case.

4.3.1 Data encoding

The encoding and storage process when $n(t) \leq Bx$ are the same as right arrival case. When $n(t) > Bx$, the original data segments will be encoded one by one in the existing combined segments. Let r be a positive integer. If the sequence number of c_j satisfies $j = r \cdot Bx + k$, c_j will be encoded in f_i^k, as the following equation.

$$f_i^k = f_i^{k\prime} + \beta_{ij} c_j, \tag{4.11}$$

where $f_i^{k\prime}$ is the former combined segment stored in buffer k before c_j is encoded.

We then give an example to show how to encode the original data segments. Assume that the buffer size in each sensor node is $B = 3$, the maximum number of data segments encoded in a combined segment is $x = 3$, the total number of data segments generated in time interval t_0 is $n(t_0) = 8$. For the sake of convenience, here we define

$$[c_1, ..., c_l] = \sum\nolimits_{j=1}^{l} \beta_{ij} c_j \tag{4.12}$$

to denote the combined segments, omiting the coefficients β_{i1}, ..., β_{il}.

Equations (4.13), (4.14), (4.15) and (4.16) show the data distribution in sensor node i. The data encoding and storage in the right arrival case is shown in equation (4.13). In the late arrival case, when the total number of data segments is not larger than Bx, the data segments are encoded in the last combined segment f_i^B. As shown in equation (4.14), c_9 is encoded in f_i^3. When the total number of data segments is larger than Bx, the data segments will be encoded one by one in the existing combined segments. As shown in equation (4.15), c_{10} is encoded in f_i^1, c_{11} is encoded in f_i^2. As shown in equation (4.16), c_{12} is encoded in f_i^3, c_{13} is encoded in f_i^1.

$$\{f_i^1 = [c_1, c_2, c_3],\ f_i^2 = [c_4, c_5, c_6],\ f_i^3 = [c_7, c_8]\}. \tag{4.13}$$

$$\{f_i^1 = [c_1, c_2, c_3],\ f_i^2 = [c_4, c_5, c_6],\ f_i^3 = [c_7, c_8, c_9]\}. \tag{4.14}$$

$$\{f_i^1 = [c_1, c_2, c_3, c_{10}],\ f_i^2 = [c_4, c_5, c_6, c_{11}],\ f_i^3 = [c_7, c_8, c_9]\}. \tag{4.15}$$

$$\{f_i^1 = [c_1, c_2, c_3, c_{10}, c_{13}],\ f_i^2 = [c_4, c_5, c_6, c_{11}],\ f_i^3 = [c_7, c_8, c_9, c_{12}]\}. \tag{4.16}$$

The encoding process is performed by each sensor node. An original data segment c_j is encoded with one of the B combined segments in sensor node i. The data encoding algorithm (Algorithm 4) is locally executed at each sensor node.

Note that if $n(t) \leq Bx$, each of the $B-1$ combined segments f_i^1, f_i^2,..., f_i^{B-1} encodes x data segments. Thus, $C(f_i^k) = x$, $k = 1, ..., B-1$. The last combined data segment f_i^B encodes at most x data segments. Thus, $C(f_i^B) \leq x$. If $n(t) > Bx$, each of the B combined segments may encode more than x data segments. From the encoding process, we know that $C(f_i^p) \geq C(f_i^q)$ if $p < q$. Thus, we have

$$C(f_i^1) = maximum\{C(f_i^k)\}. \tag{4.17}$$

Algorithm 4: Data encoding in the late arrival case

```
Input: Original data segment $c_j$, total number $n(t)$, buffer size $B$, integer $r$.
Output: A set of combined segments $F_i = \{f_i^1, f_i^2, ..., f_i^B\}$.
for $j = 1$ to $n(t)$ do
    for $k = 1$ to $B$ do
        if $j \leq B$ then
            if $(k-1)x + 1 \leq j \leq kx$ then
                Randomly generate $\beta_{ij}$ from $F_q$ ;
                $f_i^k = f_i^k + \beta_{ij} c_j$;
            end
        end
        else
            if $j = r \cdot Bx + k$ then
                Randomly generate $\beta_{ij}$ from $F_q$ ;
                $f_i^k = f_i^k + \beta_{ij} c_j$;
            end
        end
    end
end
```

The data segments encoded in f_i^1 include two parts. One part of the data segments are encoded before the mBS delays. The number of these data segments equals to x. Another part of the data segments are encoded after the mBS delays. We denote the number of data segments in this part as v, where $v = \lceil (n(t) - Bx)/B \rceil$. Thus, the number of data segments encoded in f_i^1 equals to

$$C(f_i^1) = x + v. \tag{4.18}$$

From equations (4.17) and (4.18), we can obtain

$$maximum\{C(f_i^k)\} = x + v. \tag{4.19}$$

When the mBS decodes the linear equations for the combined segments, the number of sensor nodes it queries (equals to the number of equations) should be not less than $maximum\{C(f_i^k)\}$ (equals to the number of variables). Thus, we have the following Lemma.

Lemma 4.2. *In the late arrival case, with $n(t) > Bx$, the number of sensor nodes that the mBS needs to query is at least $x + v$, where $v = \lceil (n(t) - Bx)/B \rceil$.*

Note that the value of $x + v$ should not be bigger than the total number of sensor nodes N. That is

$$x + v \leq N. \tag{4.20}$$

Since $v = \lceil (n(t) - BW_0)/B \rceil$ and $W_0 = x$, from equation (4.20), we can obtain

$$n(t) \leq BN. \tag{4.21}$$

That is, the maximum number of data segments that the sensor nodes can collect equals to BN.

From equation (4.21), the maximum number of collected data segments can be adjusted by adjusting the buffer size B of each sensor node. In the practical applications, it needs to make a trade off between the cost of the sensor nodes and the data sensing time interval.

4.3.2 Data decoding

Assume that the last original data segment $c_{n(t)}$ is encoded in f_i^u, where $u = (n(t) - Bx) \bmod B$. From the encoding process, we have

$$C(f_i^1) = C(f_i^2) = ... = C(f_i^u), \tag{4.22}$$

and

$$C(f_i^{u+1}) = ... = C(f_i^B) = C(f_i^1) - 1. \tag{4.23}$$

From equations (4.22), (4.23) and (4.18), we can obtain the number of original data segments encoded in each of the B combined data segments as

$$C(f_i^k) = \begin{cases} x+v, & 1 \leq k \leq u, \\ x+v-1, & u < k \leq B, \end{cases} \tag{4.24}$$

where $v = \lceil (n(t) - Bx)/B \rceil$, $u = (n(t) - Bx) \bmod B$.

When $n(t) \leq Bx$, the mBS can reconstruct all the original data segments by querying any x sensor nodes with high probability. The decoding process is the same as the right arrival case.

We show the decoding process when $n(t) > BW_0$, in which the mBS can reconstruct all the original data segments by querying any $x+v$ sensor nodes with high probability. For the sake of convenience, assume that the $x+v$ sensor nodes queried by the mBS are sensor node 1, sensor node 2,..., sensor node $x+v$. The mBS querying these $x+v$ sensor nodes will collect $x+v$ sets of combined segments $F_1, F_2, ..., F_{x+v}$, where $F_i = \{f_i^1, f_i^2, ..., f_i^B\}$ is the set of combined segments collected from sensor node i, $i = 1, ..., x+v$. The mBS also collects the related coefficient vectors of the combined segments. Let $\mathbf{f}_k = \{f_1^k, f_2^k, ..., f_{x+v}^k\}$ denote the set of combined segments collected from buffer k of the $x+v$ sensor nodes, $k = 1, ..., B$. Decoding of $\mathbf{f}_k$ is to decode the original data segments from the combined segment which is stored in buffer k in a sensor node. The decoding process includes B stages, decoding $\mathbf{f}_1$, $\mathbf{f}_2$,..., $\mathbf{f}_B$.

From equation (4.24), we know that in $\mathbf{f}_1$, $\mathbf{f}_2$,...,$\mathbf{f}_u$, each of the combined segments encodes $x + v$ original data segments, and in $\mathbf{f}_{u+1}$, $\mathbf{f}_{u+2}$,...,$\mathbf{f}_B$, each of the combined segments encodes $x+v-1$ original data segments. In the system of linear equations for $\mathbf{f}_k$ where $k = 1, ..., u$, the solution is the set of original data segments $\{c_{(k-1)x+1}, c_{(k-1)x+2}, ..., c_{(k-1)x+x}$, $c_{Bx+k}, c_{Bx+B+k}, ..., c_{Bx+(v-1)B+k}\}$, which is the set of original data segments encoded in the combined segment in buffer k of each sensor node. In the system of linear equations for $\mathbf{f}_k$ where $k = u+1, ..., B$, the solution is the set of original data segments $\{c_{(k-1)x+1}, c_{(k-1)x+2}, ..., c_{(k-1)x+x}, c_{Bx+k}, c_{Bx+B+k}, ..., c_{Bx+(v-2)B+k}\}$, which is the set of original data segments encoded in the combined segment in buffer k of each sensor node.

We show the process of decoding the solution from $\mathbf{f}_k$, $k = 1, ...B$. From equations (4.2), (4.11) and (4.19), we know that

$$\mathbf{f}_k = \mathbf{A}_k \mathbf{c}_k, \tag{4.25}$$

where

$$\mathbf{c}_k = \begin{cases} (c_{(k-1)x+1}, c_{(k-1)x+2}, ..., c_{(k-1)x+x}, c_{Bx+k}, c_{Bx+B+k}, ..., c_{Bx+(v-1)B+k})^T, & 1 \leq k \leq u, \\ (c_{(k-1)x+1}, c_{(k-1)x+2} ..., c_{(k-1)x+x}, c_{Bx+k}, c_{Bx+B+k}, ..., c_{Bx+(v-2)B+k})^T, & u < k \leq B, \end{cases} \tag{4.26}$$

and

$$
\mathbf{A}_k = \begin{cases}
\begin{pmatrix}
\beta_{1.(k-1)x+1} & \beta_{2.(k-1)x+1} & \cdots & \beta_{(x+v).(k-1)x+1} \\
\beta_{1.(k-1)x+2} & \beta_{2.(k-1)x+2} & \cdots & \beta_{(x+v).(k-1)x+2} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{1.(k-1)x+x} & \beta_{2.(k-1)x+x} & \cdots & \beta_{(x+v).(k-1)x+x} \\
\beta_{1.Bx+k} & \beta_{2_k.Bx+k} & \cdots & \beta_{(x+v).Bx+k} \\
\beta_{1.Bx+B+k} & \beta_{2.Bx+B+k} & \cdots & \beta_{(x+v).Bx+B+k} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{1.Bx+(v-1)B+k} & \beta_{2.Bx+(v-1)B+k} & \cdots & \beta_{(x+v).Bx+(v-1)B+k}
\end{pmatrix}, & 1 \leq k \leq u, \\
\begin{pmatrix}
\beta_{1.(k-1)x+1} & \beta_{2.(k-1)x+1} & \cdots & \beta_{(x+v).(k-1)x+1} \\
\beta_{1.(k-1)x+2} & \beta_{2.(k-1)x+2} & \cdots & \beta_{(x+v).(k-1)x+2} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{1.(k-1)x+x} & \beta_{2.(k-1)x+x} & \cdots & \beta_{(x+v).(k-1)x+x} \\
\beta_{1.Bx+k} & \beta_{2_k.Bx+k} & \cdots & \beta_{(x+v).Bx+k} \\
\beta_{1.Bx+B+k} & \beta_{2.Bx+B+k} & \cdots & \beta_{(x+v).Bx+B+k} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{1.Bx+(v-2)B+k} & \beta_{2.Bx+(v-2)B+k} & \cdots & \beta_{(x+v).Bx+(v-2)B+k}
\end{pmatrix}, & u < k \leq B.
\end{cases} \tag{4.27}
$$

The key property required for successful decoding of $\mathbf{f}_k$ is that the coefficient vectors in $\mathbf{A}_k$ must be linearly independent,$k = 1, ..., B$. This is generally true for a large enough field size q [65]. The original data segments can be decoded using Gaussian Elimination [111]. After decoding all the linear equations, the mBS can obtain all the original data segments $\mathbf{c} = \bigcup_{k=1}^{B} \mathbf{c}_k$.

Theorem 4.3. *The success ratio of data collection by DSC-ADC with buffer size $B = \lceil n(t_0)/x \rceil$ in each sensor node is close to 100% by using a large enough finite field size q for coefficients.*

Proof. When the mBS arrives on time, from Lemma 4.1, each sensor node with buffer size $B = \lceil n(t_0)/x \rceil$ is enough to encode all original data segments. When the BS delays, each sensor node continues to encode the original data segments in the existing combined segments. Thus, each sensor node can encode all the original data segments in the combined segments. To guarantee the decodable of the linear equations for the collected combined segments, the mBS queries x sensor nodes when $n(t) \leq Bx$, and queries $x + v$ sensor nodes when $n(t) > Bx$. The probability of decoding the original data segments from the combined segments are close to 100% for a large enough field size q. □ □

Note that the value of x affects the energy consumption of the sensor nodes, since the number of sensor nodes queried by the mBS depends on the value of x. We will give a further discussion about it in the simulations.

4.4 Performance analysis and discussion

4.4.1 Computation, transmission and storage overheads in DSC-ADC

Similar to the DSC-mLDC scheme that are presented in Chapter 3, DSC-ADC is with low computation, transmission and storage overheads.

In DSC-ADC, each sensor node separately encodes a certain number of original data segments in a combined segment. The computation complexity for data encoding in each sensor node is linear, which depends on the number of encoded original data segments. The computation overhead for data coding in DSC-ADC lies mainly in the decoding process. This is performed in the powerful mBS. In the decoding process, the mBS solves B systems of linear equations. The original data segments encoded in the B combined segments in a sensor node (i.e., the variables in the B systems of linear equations) can be decoded in $O(k^3)$ by using the Gaussian Elimination [111], where k is the order of coefficient matrix for a system of linear equations,and the order of coefficient matrix for each system of linear equations is not larger than k.

In DSC-ADC, a sensor node uploads B combined segments and the associated coefficient vectors to the mBS when it is queried by the mBS. In right arrival case, the mBS queries x sensor nodes during data collection. The total number of combined segments uploaded from the queried sensor nodes is Bx. In late arrival case, the mBS queries $x + v$ sensor nodes during data collection, where $v = \lceil (n(t) - Bx)/B \rceil$. The total number of combined segments uploaded from the queried sensor nodes is $B(x + v)$. From equation (4.5), the number of sensor nodes queried by the mBS can be reduced with a few additional storage space in each sensor node. Since establishing a connection between a sensor node and the mBS also consumes energy [113], the transmission overhead for transmitting combined segments to the mBS can be reduced with a few additional storage space in each sensor node. Besides the combined segments, the coefficient vectors are also uploaded from the queried sensor nodes. The overhead of uploading the coefficient vectors is much lower than the overhead of uploading large size of combined combined segments.

In DSC-ADC, each sensor node stores B combined segments and the B associated coefficient vectors. The storage overhead depends on the size of combined segments, the size of finite field for coefficients q, and how many original data segments are encoded in a combined segment. The storage overhead can be reduced a lot by coding, since each sensor node stores the combined segments and the associated coefficient vectors instead of storing the original data segments. The storage overhead for a combined segment with the associated coefficient vector, and the percentage of storage overhead reduction be coding have been shown in Section 2.2.1.

4.4.2 Discussion

Similar to the DSC-mLDC scheme in Chapter 3, in DSC-ADC, to focus on the data encoding in each sensor node and the data decoding process in the mBS, we assume that each original data segment is recorded by all the sensor nodes by using some existing data dissemination method. Actually, if each original data segment is not recorded by all the sensor nodes, the mBS still can decode the original data segments with high probability.

In DSC-ADC, if each original data segment is recorded by at least $d = \frac{5N}{x} \ln x$ sensor nodes (x is the number of original data segments encoded in a combined segment), the original data segments can be successfully decoded with high probability. Specifically, we have $Pr[det(\boldsymbol{\beta}'_u) = 0] \leq \frac{x}{q} + o(1)$. Note that if each original data segment is recorded by less sensor nodes, the data dissemination overhead is lower.

Similar to the DSC-mLDC scheme in Chapter 3, we neglect the latency for data dissemination in the proposed DSC-ADC scheme and consider it as the future work.

4.5 Performance Evaluation

In this section, we evaluate the performance of the proposed DSC-ADC scheme by simulations. Since in DEC [58] the number of collected data segments is fixed and limited, and in PNC [51] the sensor nodes can collect only part of the data segments, we do not compare with the two schemes in the simulations. In the simulations, we evaluate the performance of DSC-ADC on: (1) Success ratio of data collection; (2) energy consumption for data transmission to the mBS; and (3) time for data transmission to the mBS.

4.5.1 Performance Metrics

We deploy 1000 sensor nodes randomly into a field of 200m $\times$ 200m. The distance between the sensor nodes and the mBS is much longer than the distance between the sensor nodes, as suggested in Lindsey and Raghavendra [112]. Without necessarily entering deep into the sensor field, the mBS can perform data collection. The size of a data segment is 9 KB. The storage capacity of a buffer is 10 KB. For a sensor node with B buffers, the storage capacity is $10B$ KB. We use the general method to generate the coefficient vectors. For a coefficient vector, the number of coefficients should be generated equals to the number of data segments encoded in the associated combined segment. Each coefficients is randomly generated from a finite field F_q. Unless otherwise specified, the size of finite field for coefficients is $q = 2^8$, which can be efficiently implemented in a 8-bit or more advanced microprocessor [75]. Note that the size of coefficient is much smaller than the size of data segment. A buffer can store a combined segment and the associated coefficient vector. In DSC-ADC, one packet includes a combined data segment and the associated coefficient vector. The solution of linear equations in network coding are using the Gaussian Elimination [111].

Table 4.1 demonstrates part of the important parameters and settings in the simulations.

Table 4.1: System Parameters and Settings for All Data Segment Collection

System Parameters	**Settings**
Simulator	Matlab
Length $\times$ Width	200 m $\times$200 m
Number of sensor nodes	600
Transmit range between sensor nodes	20 m
Transmit range between sensor nodes and mBS	150 m $\sim$ 250 m
Size of a data segment	9KB
Energy consumption for sending a data segment	40 nAH
Size of finite field for coefficients q	2^8
Simulation times	1000
Confidence interval	95%

4.5.2 Simulation results

4.5.2.1 Success ratio of data collection

We first evaluate the success ratio of data collection in DSC-ADC by varying the buffer size B in each sensor node. In this simulation, the total number of original data segments in the right arrival case is $n(t_0) = 400$. The total number of original data segments in late arrival case is $n(t)$, where $n(t) \in (400, 600]$. In each data collection, the mBS arriving on time or late is set randomly. During data collection, the mBS first queries a small subset (i.e., the minimum number) of sensor nodes uniformly at random from the sensor network to collect data. If the data segments cannot be decoded successfully, the mBS will query additional sensor nodes one by one until all the data segments are decoded. The additional sensor nodes are queried uniformly at random from the sensor network (excluding the sensor nodes which have queried in the same time of data collection).

Fig 4.5 shows the success ratio of data collection as a function of the buffer size B in each sensor node. The success ratio of data collection is higher for large finite field size q for coefficients. The success ratio of data collection is close to 100% when $q = 2^8$. This fact is also stated in the proof of Theorem 4.3.

4.5.2.2 Energy consumption for data transmission to the mBS

We evaluate the energy consumption for data transmission to the mBS by varying the total number of data segments. We use the energy model by Mainwaring et al [52]. The energy consumption for transmitting one combined segment (including the associated coefficient) is 40 nAH (21.8 Ampere hour). As suggested in [113], establishing a connection between a sensor node and the mBS consumes energy too. Consider that the energy consumption for establishing a connection between a sensor node and the mBS is 20 nAH. A sensor node will upload B packets when it is queried by the mBS. Thus, the energy consumption for a queried sensor node is $(20 + 40B)$ nAH. The mBS first queries the minimum number of sensor nodes to collect data. If the data segments cannot be decoded successfully, the mBS will query additional sensor nodes one by one until all the data segments are decoded completely.

As shown in Fig 4.7, the energy consumption decreases as the buffer size B increases. The energy consumption is large when $B = 1$. Note that in DEC [58] the

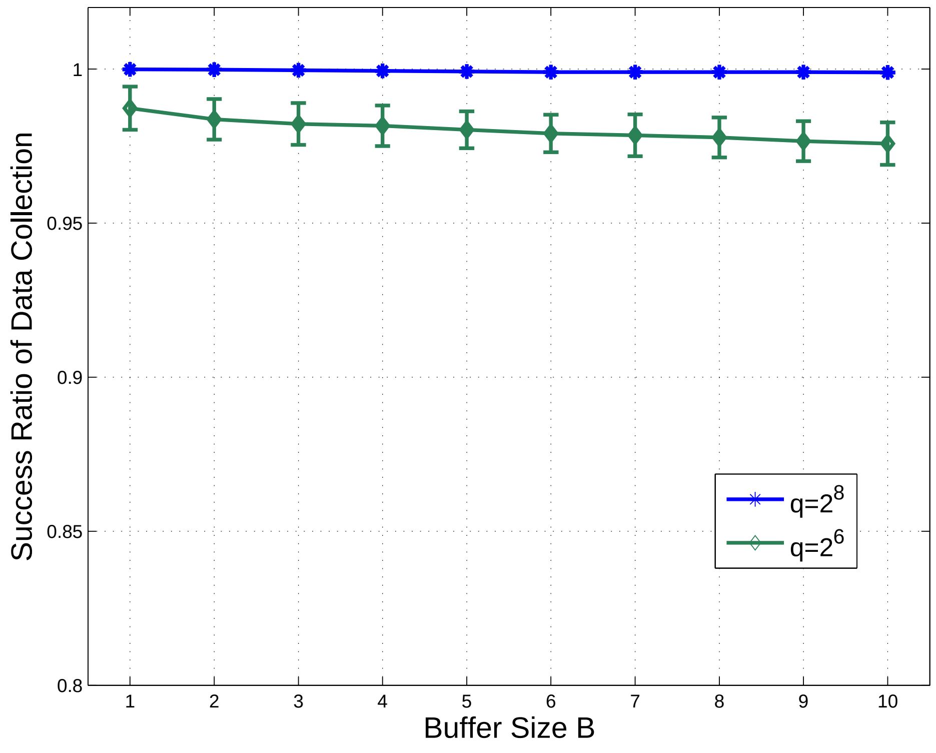

FIGURE 4.5: Success ratio of data collection vs. buffer size B in each sensor node.

buffer size B equals to 1. The DEC scheme is similar to the proposed DSC-ADC scheme when each sensor has one buffer. Thus, the energy consumption in DEC is large. The proposed DSC-ADC scheme can reduce the energy consumption with a few additional storage space in each sensor node. The energy consumption can be reduced a lot with 1 more buffer in each sensor node ($B = 2$). The reduction range is smaller between bigger buffer sizes.

4.5.2.3 Data transmission time to the mBS during data collection

Since a fast data retrieval is usually desired in continuous data collection, we evaluate the time for data transmission to the mBS during data collection by varying the total number of data segments. The time cost for transmitting one packet is 2 second.

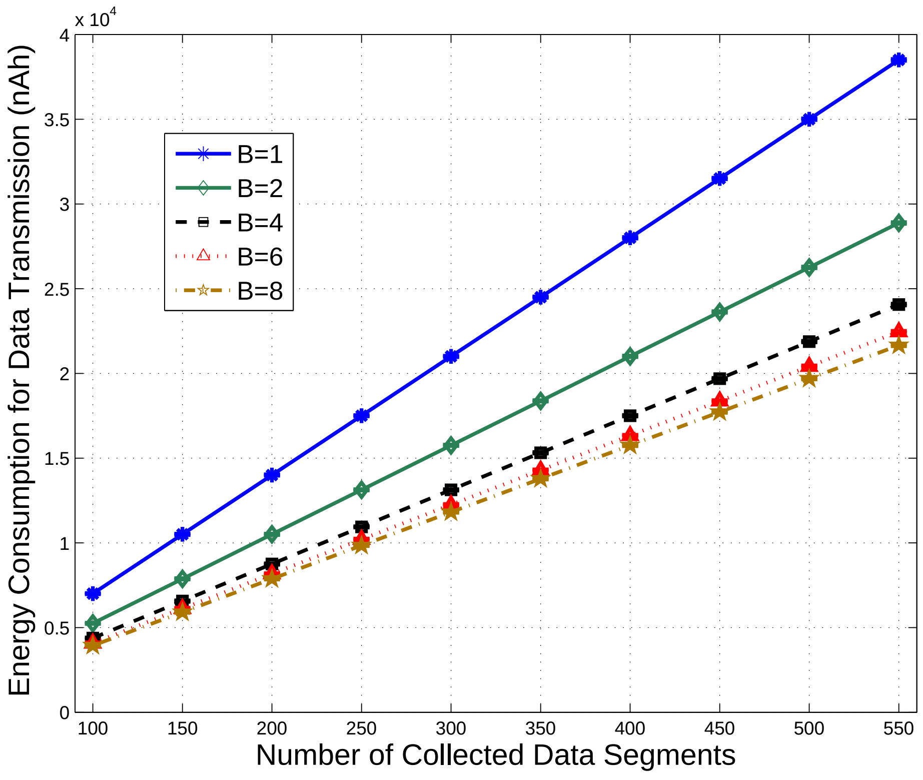

FIGURE 4.6: Energy consumption for data transmission vs. total number of data segments.

The time cost for establishing a connection between the mBS and a sensor node is 4 second. For simplicity, we neglect other affecting factors which is much less than the transmission time. The time cost for a queried sensor node to finish data transmission is $(4 + 2B)$ s.

As shown in Fig 4.7, the data transmission time decreases as the buffer size B increases. The mBS spends more time for data collection when $B = 1$. This is similar to the DEC scheme, in which each sensor node has one buffer. The proposed DSC-ADC scheme can reduce the data transmission time with a few additional storage space in each sensor node. The data transmission time can be reduced a lot with 1 more buffer in each sensor node ($B = 2$). The reduction range is smaller between

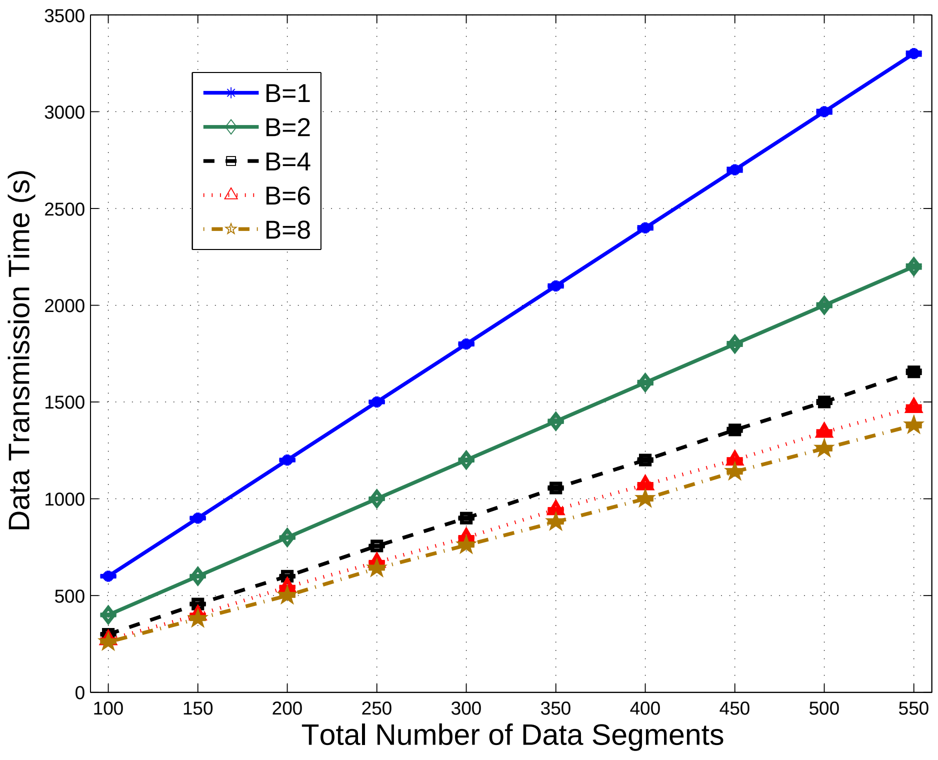

FIGURE 4.7: Energy consumption for data transmission vs. total number of data segments.

bigger buffer sizes. Not surprisingly, DSC-ADC performs better with bigger B. However, too big buffer size may not be available at a sensor node, and the prices of sensor node with bigger buffer size are higher. Thus, in the practical applications, it needs to make a trade off in the affecting factors.

4.6 Summary

In this chapter, we consider the second scenario of continuous data collection (i.e., All data segment collection), and present a novel data collection scheme called Distributed Separate Coding for All Data segment Collection (DSC-ADC). DSC-ADC not only provides an efficient storage method for continuously collecting data segments, but also achieves a high success ratio of data collection. We present the data encoding process and data decoding process in DSC-ADC. We show that in DSC-ADC, the number of sensor nodes that should be queried by the mBS can be reduced with a few additional storage space in each sensor node, which result in reducing the energy consumption for data transmission to the mBS. We also show that the success ratio of data collection in DSC-ADC is very close to 100% in either the right arrival case or late arrival case by using larger enough finite field for coefficients in both theoretical analysis and simulations. Furthermore, The performance evaluation has been conducted through computer simulations. It further demonstrates the feasibility and superiority of the proposed DSC-ADC scheme.

Chapter 5

Conclusions and Future Work

Wireless Sensor Networks (WSNs) open up a new opportunity for us to observe and interact with the extreme environments [117], in which the data are difficult, expensive, or even impossible to collect by humans. Consider the continuous data collection in the extreme environments, in which the that data are continuously sensed and collected by the sensor nodes. Data collection is only performed from time to time by a mobile Base Station (mBS). Sensor nodes have to store the continuously collected data segments over time by themselves, and provide the desired data when the mBS arrives and performs data collection. This dissertation addresses the continuous data collection in WSNs with a mobile Base Station (mBS). We consider two scenarios of continuous data collection. (1) Latest data segment collection. (2) All data segment collection. We propose two Distributed Separate Coding based schemes for the two scenarios, respectively. The two proposed schemes are Distributed Separate Coding for m Latest Data segment Collection (DSC-mLDC) and Distributed Separate Coding for All Data segment Collection (DSC-ADC). Overall system performance (e.g., success ratio of data collection, energy consumption, storage overhead) are improved by applying our proposed schemes for continuous data collection in WSNs.

This chapter is organized as follows. The proposed Distributed Separate Coding for m Latest Data segment Collection (DSC-mLDC) is concluded in Section 5.1. The proposed Distributed Separate Coding for All Data segment Collection (DSC-ADC) is concluded in Section 5.2. We present the future work in Section 5.3.

5.1 Distributed Separate Coding for m Latest Data segment Collection (DSC-mLDC)

In Chapter 3, we focus on the first scenario of continuous data collection, i.e., latest data segment collection. We consider to collect the m latest data segments, where where m is the number of latest data segments in a time interval t in which $n(t)$ ($m \leq n(t)$) data segments are generated. We propose a novel data collection scheme called Distributed Separate Coding for m Latest Data segment Collection (DSC-mLDC). DSC-mLDC is decentralized and based on the mBS's randomly accessing. By separately encoding a certain number of data segments in a combined segment, and doing decoding-free data replacement in the buffers of each sensor node, DSC-mLDC provides an efficient storage method for continuously collecting data segments with a high success ratio. We present the data encoding process, data replacement precess and data decoding process in DSC-mLDC.

The main advantages of the proposed DSC-mLDC scheme are summarized as follows.

- In DSC-mLDC, with a minimum buffer size 2 in each sensor node, by querying any $m-1$ sensor nodes, the mBS can reconstruct the m latest data segments with high probability.
- The success ratio of data collection in DSC-mLDC is very close to 100% by using a large enough finite field size for the coefficients.
- In DSC-mLDC, the necessary storage space in each sensor node does not depend on the number of required data. It can be adjusted by changing the number of sensor nodes queried by the mBS.

- In DSC-mLDC, the transmission cost for data submission to the mBS can be reduced with a few additional storage space in each sensor node.
- Compare to the related work (i.e., PNC in [51]), DSC-mLDC is flexible and efficient in improving the network performance in terms of increasing the success ratio of data collection, reducing the energy consumption and storage overhead.

The discussion about the number of buffers and the number of data segments submitted from each node makes the proposed scheme much more convincing. The comprehensive performance evaluation has been conducted through computer simulation. It is shown that the proposed DSC-mLDC scheme outperforms the existing scheme significantly.

5.2 Distributed Separate Coding for All Data segment Collection (DSC-ADC)

In Chapter 4, we focus on the second scenario of continuous data collection, i.e., all data segment collection. We consider to collect all the $n(t)$ data segments generated in a time interval t. We propose a novel data collection scheme called Distributed Separate Coding for All Data segment Collection (DSC-ADC). DSC-ADC is decentralized and based on the mBS's randomly accessing. We consider the following two cases: (1) *right arrival case* that the mBS arrives on time, (2) *late arrival case* that the mBS arrives late.

By separately encoding a certain number of data segments in a combined segment, and storing the combined segments in the corresponding buffers of each sensor node, the DSC-ADC scheme provides an efficient storage method to collect all data segments with a high success ratio. We present the data encoding process and data decoding process in DSC-ADC. By randomly querying a small subset of sensor nodes, the mBS can reconstruct all the original data segments with high probability in both the right arrival case and the late arrival case. DSC-ADC is more energy efficient compared to the related work (i.e., DEC in [58]). We prove that the success ratio of DSC-ADC based data collection is close to 100% by using a large enough finite field size for the coefficients. The number of sensor nodes that should be queried by the mBS can be reduced with a few additional storage space in each sensor node. The performance evaluation has been conducted through computer simulations. It further demonstrates the feasibility and superiority of the proposed DSC-ADC scheme.

5.3 Future Work

Distributed storage systems introduce redundancy to increase reliability in WSNs. Numerous challenges and opportunities are emerged in distributed storage systems for WSNs. We consider the following issues for our future efforts, w.

- **Diverse coding methods for distribute storage in WSNs** : The propose schemes in this dissertation are based on random linear coding, since random linear coding is easy and suitable to deploy in wireless sensor networks. In the future work, we will consider other coding methods, such as fountain coding. Fountain coding is a promising solution to reduce the decoding complexity. However, the implement of Fountain codes in WSNs is more difficult than that of random linear coding.
- **Efficient data dissemination method for distributed storage in WSNs**: In this dissertation, to focus on the data encoding process in each sensor node and the data decoding process in the mBS, in our proposed schemes, we assume that each original data segment is recorded by the sensor nodes by using some existing data dissemination method. In the future work, we will consider efficient data dissemination methods for distributed storage in WSNs to reduce the energy consumption in the data dissemination process.

- **Repair problem for distributed storage in WSNs** : When using coding for distributed storage in WSNs, the repair problem arises when the network wants to maintain the same level of reliability. If a sensor node storing encoded data segments fails, in order to maintain the same level of reliability, the encoded data segments should be created at a new sensor node [118]. The consideration of the repair network traffic gives rise to new design challenges. In the future work, we will consider efficient methods to minimize the communication required to generated encoded data segments from the alive sensor nodes, so as to maintain the same level of reliability after some sensor node fails.

- **Security in Distributed Storage for WSNs** : The issues of security and privacy are important for distributed storage. When using coding for distributed storage in WSNs, errors can be propagated in several mixed packets and thus error-control mechanism is required [119]. A related issue is that of privacy of the data by information leakage to eavesdroppers during data collection. As the future work, we will consider the security problem in data collection for WSNs, which is to ensure that the original data segments can only be decoded by the specified mBS. Kher-01

Bibliography

[1] M. D. Francesco, S. K. Das, and G. Anastasi, "Data collection in wireless sensor networks with mobile elements: A survey," *ACM Transactions on Sensor Networks*, vol. 8, no. 7, pp. 7–31, 2011.

[2] D. Wang, J. Liu, and Q. Zhang, "On mobile sensor assisted field coverage," *ACM Transactions on Sensor Networks*, vol. 9, no. 3, 2013.

[3] E. C. Ngai, M. B. Srivastava, and J. Liu, "Context-aware sensor data dissemination for mobile users in remote areas," in *Proceedings of IEEE INFOCOM 2012 mini-conference.* IEEE, 2012.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comm. ACM*, vol. 38, no. 4, pp. 393–422, 2002.

[5] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proceedings of ACM MOBICOM99.* ACM, 1999.

[6] L. Ruiz-Garcia, L. Lunadei, P. Barreiro, and I. Robla, "A review of wireless sensor technologies and applications in agriculture and food industry: State of the art and current trends," *Sensors*, vol. 9, pp. 4728 – 4750, 2009.

[7] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, vol. 3, pp. 257 – 279, 2005.

[8] G. Zhou, J. Lu, C.-Y. Wan, M. D. Yarvis, and J. A. Stankovic, *Body Sensor Networks.* Cambridge, MA: MIT Press, 2008.

[9] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *IEEE Comput.*, vol. 37, no. 8 (Special Issue on Sensor Networks), pp. 41–49, 2004.

[10] P. Bahl, R. Chancre, and J. Dungeon, "SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks," in *Proceeding of the 10th International Conference on Mobile Computing and Networking (MobiCom'04).* New York, NY: ACM, 2004, pp. 112–117.

[11] F. Wang, J. Liu, and L. Sun, "Ambient data collection with wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, no. 10, 2010.

[12] A. Natarajan, M. Motani, B. de Silva, K. Yap, and K. C. Chua, "Investigating network architectures for body sensor networks," in *Network Architectures*, G. Whitcomb and P. Neece, Eds. Dayton, OH: Keleuven Press, 2007, pp. 322–328.

[13] S. Misra, S. C. Misra, and I. Woungang, *Guide to Wireless Sensor Networks.* Springer, 2009.

[14] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless reasearch," in *Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN).* IEEE, 2005.

[15] B. Warneke, B. Atwood, and K. Pister, "Smart dust mote forerunners," in *Proceedings of IEEE International Conference on Micro Electro Mechanical Systems (MEMS).* IEEE, 2001.

[16] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *Proceedings of the 5th international conference on Embedded networked sensor systems.* ACM, 2007.

[17] S. Olariu and I. Stojmenovic, “Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting,” in *Proceedings of IEEE INFOCOM.* IEEE, 2006.

[18] J. Wieselthier, G. Nguyen, and A. Ephremides, “On the construction of energy-efficient broadcast and multicast trees in wireless networks,” in *Proceedings of IEEE INFOCOM 2000.* IEEE, 2000.

[19] I. Beichl and F. Sullivan, “The metropolis algorithm,” *Computing in Science and Engineering*, vol. 2, pp. 65 – 69, 2000.

[20] V. Katiyar, N. Chand, and N. Chauhan, “Recent advances and future trends in wireless sensor networks,” *International Journal of Applied Engineering Research*, vol. 1, pp. 330 – 342, 2010.

[21] S. Rothery, W. Hu, and P. Corke, “An empirical study of data collection protocols for wireless sensor networks,” in *Proceedings of the workshop on Real-world wireless sensor networks (REALWSN 08).* ACM, 2008.

[22] Y. Wang, S. Jain, M. Martonosi, and K. Fall, “Erasure-coding based routing for opportunistic networks,” in *Proceedings of ACM SIGCOMM workshop (WTDN05).* IEEE, 2005.

[23] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter, “Luster: wireless sensor network for environmental research,” in *Proceedings of the 5th international conference on Embedded networked sensor systems.* ACM, 2007.

[24] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, “Sensorscope: Out-of-the-box environmental monitoring,” in *Proceedings of International Conference on Information Processing in Sensor Networks (IPSN).* ACM/IEEE, 2008.

[25] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macroscope in the redwoods," in *Proceedings of the 3rd international conference on Embedded networked sensor systems.* ACM, 2005.

[26] Y. Kim, T. Schmid, Z. M. Charbiwala, J. Friedman, and M. B. Srivastava, "Nawms: Nonintrusive autonomous water monitoring system," in *Proceedings of the 6th ACM conference on Embedded network sensor systems.* ACM, 2008.

[27] G. WernerAllen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proceedings of the 7th symposium on Operating systems design and implementation.* USENIX Association Berkeley, 2006.

[28] W. Z. Song, R. Huang, M. Xu, A. Ma, B. Shirazi, and R. LaHusen, "Air-dropped sensor network for real-time high-fidelity volcano monitoring," in *Proceedings of the 7th international conference on Mobile systems, applications, and services.* ACM, 2009.

[29] C. Hartung, R. Han, C. Seielstad, , and S. Holbrook, "Firewxnet: A multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments," in *Proceedings of the 4th international conference on Mobile systems, applications and services.* ACM, 2006.

[30] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proceedings of International Conference on Information Processing in Sensor Networks (IPSN).* ACM/IEEE, 2007.

[31] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, , and P. Zanon, "Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment," in *Proceedings of International Conference on Information Processing in Sensor Networks (IPSN).* ACM/IEEE, 2009.

[32] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking (MobiCom).* ACM, 2004.

[33] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking.* ACM, 2000.

[34] R. Farivar, M. Fazeli, and S. Miremadi, "Directed flooding: a fault-tolerant routing protocol for wireless sensor networks," in *Proceedings of Systems Communications.* IEEE, 2005.

[35] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of 44th Annual IEEE Symposium on Foundations of Computer Science.* IEEE, 2003.

[36] W. Yen, C. Chen, and C. Yang, "Single gossiping with directional flooding routing protocol in wireless sensor networks," in *Proceedings of 3rd IEEE Conference on Industrial Electronics and Applications.* IEEE, 2008.

[37] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proceedings of the 2nd international conference on Embedded networked sensor systems.* ACM, 2004.

[38] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks," in *Proceedings of Parallel and Distributed Processing Symposium.* IEEE, 2004.

[39] W. Z. Song, F. Yuan, and R. LaHusen, "Time-optimum packet scheduling for many-to-one routing in wireless sensor networks," in *Proceedings of IEEE MASS.* IEEE, 2006.

[40] L. Paradis and Q. Han, "Tigra: Timely sensor data collection using distributed graph coloring," in *Proceedings of Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom).* IEEE, 2008.

[41] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: Ultra-low power data gathering in sensor networks," in *Proceedings of International Conference on Information Processing in Sensor Networks (IPSN).* ACM/IEEE, 2007.

[42] K. S. J. Pister and L. Doherty, "Tsmp: Time synchronized mesh protocol," in *Proceedings of the IASTED International Symposium on Distributed Sensor Networks (DSN08)*, 2008.

[43] G. S. Ahn, E. Miluzzo, A. T. Campbell, S. G. Hong, and F. Cuomo, "Funneling-mac: A localized, sink-oriented mac for boosting fidelity in sensor networks," in *Proceedings of ACM SenSys.* ACM, 2006.

[44] C. T. Ee and R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks," in *Proceedings of ACM SenSys.* ACM, 2004.

[45] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel, "Context-aware sensor data dissemination for mobile users in remote areas," in *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks (IPSN 09)*, 2009.

[46] J. Beutel, B. Buchli, F. Ferrari, M. Keller, L. Thiele, and M. Zimmerling, "X-sense: Sensing in extreme environments," in *Proceedings of of Design, Automation and Test in Europe*, 2011.

[47] K. Martinez, P. Padhy, A. Elsaify, G. Zou, A. Riddoch, J. K. Hart, and H. L. R. Ong, "Deploying a sensor network in an extreme environment," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 06).* IEEE, 2006.

[48] S. Williams, M. Hurst, and A. M. Howard, "Development of a mobile arctic sensor node for earth-science data collection applications," in *American Institute of Aeronautics and Astronautics, Infotech@Aerospace,(Atlanta, GA)*, 2010.

[49] "Sonoma redwoods data," 2005, http://www.cs.berkeley.edu/ get/sonoma.

[50] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," in *Proceeding of SIGCOMM LA '01 Workshop on Data communication in Latin America and the Caribbean.* San Jose, Costa Rica: ACM, 2001, pp. 20–41.

[51] D. Wang, Q. Zhang, and J. Liu, "Partial network coding: Concept, performance, and application for continuous data collection in sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 3, pp. 1–22, 2008.

[52] A. Mainwaring, J. Polaster, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceeding of the 1st ACM international workshop on Wireless sensor networks and applications.* Atlanta, GA: ACM, 2002, pp. 88–97.

[53] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 2 – 16, 2003.

[54] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proceedings of the 22nd International Conference on Distributed Computing Systems.* Washington, DC, USA: IEEE, 2002, pp. 575–578.

[55] D. Wang, Y. Long, and F. Ergun, "A layered architecture for delay senstitive sensor networks," in *Proceedings of IEEE SECON 05.* IEEE, 2005.

[56] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Communication Surveys and Tutorials*, vol. 8, pp. 48 – 63, 2006.

[57] Z. Kong, S. Aly, and E. Soljanin, "Decentralized coding algorithms for distributed storage in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 261 – 267, 2010.

[58] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Trans. Info. Theory*, vol. 52, no. 6, pp. 2809–2816, 2006.

[59] A. Dimakis, V. Prabhakarna, and K. Ramchandran, "Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes," in *Proceeding of of the International Conference on Information Processing in Sensor Networks (IPSN05).* ACM, 2005, pp. 111–117.

[60] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications.* Prentice Hall, Upper Saddle River, NJ, 2004.

[61] N. Cao, Q. Wang, K. Ren, and W. Lou, "Distributed storage coding for flexible and efficient data dissemination and retrieval in wireless sensor networks," in *Proceedings of IEEE ICC 2010.* IEEE, 2010.

[62] D. Vukobratovic, C. Stefanovic, V. Crnojevic, F. Chiti, and R. Fantacci, "Rateless packet approach for data gathering in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 1169 – 1179, 2010.

[63] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An architecture for global-scale persistent storage," in *Proceedings of IACM ASPLOS 2012*. ACM, 2000.

[64] W. A. Burkhard and J. Menon, "Disk array storage system reliability," in *Proceedings of the 23rd Intenrational Symposium on Fault-Tolerant Computing*. IEEE, 1993.

[65] S. Acedanski, S. Deb, M. Medard, and R. Koetter, "How good is random linear coding based distributed networked storage," in *Proceedings of First Workshop Network Coding, Theory, and Applications (NetCod05)*, Riva del Garda, Italy, 2005.

[66] D. H. Wiedemann, "Solving sparse linear equations over finite fields," *IEEE Transaction on Information Theory*, 1986.

[67] C. Fragouli, J. Y. LeBoudec, and J. Widmer, "Network coding: an instant primer," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, 2006.

[68] R. Ahlswede, N. Cai, S. LI, and R. Yeung, "Network information flow," *IEEE Trans. on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[69] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *Proceedings of ACM SIGCOMM*, 2006, pp. 243–254.

[70] Y. E. Sagduyu and A. Ephremides, "Network coding in wireless queueing networks: Tandem network case," in *IEEE International Symposium on Information Theory*. Seattle, WA: IEEE, 2006, pp. 192–196.

[71] A. E. Kamal, “1+n network protection for mesh networks: Network coding-based protection using p-cycles,” *IEEE/ACM TRANSACTIONS ON NETWORKING*, vol. 18, no. 1, pp. 67–80, 2010.

[72] S. Aly and A. Kamal, “Network coding-based protection strategy against node failures,” in *Proceedings of the IEEE International Conference on Communications*, 2009, pp. 1–5.

[73] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” *IEEE Trans. Info. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[74] C. Gkantsidis and P. Rodriguez, “Network coding for large scale content distribution,” in *Proceedings of INFOCOM'2005*. IEEE, 2005, pp. 2235–2245.

[75] J. Widmer and J. Andboudec, “Network coding for efficient communication in extreme networks,” in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. Philadelphia, PA: ACM, 2005, pp. 284–291.

[76] X. Yang, X. Tao, E. Dutkiewicz, X. Huang, Y. Guo, and Q. Cui, “Energy-efficient distributed data storage for wireless sensor networks based on compressed sensing and network coding,” *IEEE Transactions on Wireless Communications*, vol. 12, pp. 5087 – 5909, 2013.

[77] I. H. Hou, Y. E. Tsai, T. F. Abdelzaher, and I. Gupta, “Adapcode: Adaptive network coding for code updates in wireless sensor networks,” in *Proceedings of the 27th IEEE International Conference on Computer Communications*. IEEE, 2008.

[78] A. Dimakis, V. Prabhakarna, and K. Ramchandran, “Distributed fountain codes for networked storage,” in *Proceeding of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2006.

[79] A. Shokrollahi, “Raptor codes,” *IEEE Transaction on Information Theory*, 2006.

[80] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, “A digital fountain approach to reliable distribution of bulk data,” in *Proceedings of SIGCOMM*, 1998.

[81] M. Luby, “Lt codes,” in *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, 2002.

[82] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, “Efficient erasure correcting codes,” *IEEE Trans. Info. Theory*, vol. 47, pp. 569 – 584, 2001.

[83] R. Blahut, *Theory and Practice of Error Control Codes*. Addison-Wesley, 1985.

[84] H. Weatherspoon and J. D. Kubiatowicz, “Erasure coding vs. replication: a quantitiative comparison,” in *Proceedings of IPTPS 02*, 2002.

[85] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, “Total recall: System support for automated availability management,” in *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[86] M. O. Rabin, “Efficient dispersal of information for security, load balancing and fault tolerance,” *Journal of the ACM*, vol. 36, p. 335C348, 1989.

[87] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983.

[88] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, “Informed content delivery across adaptive overlay networks,” in *Proceedings of SIGCOMM*, 2002.

[89] J. Considine, “Generating good degree distributions for sparse parity check codes using oracles,” in *Technical Report, BUCS-TR 2001-019, Boston University*, 2001.

[90] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, “Growth codes: Maximizing sensor network data persistence,” in *Proceedings of ACM Sigcom 06*, 2006.

[91] D. Munaretto, J. Widmer, M. Rossi, and M. Zorzi, “Network coding strategies for data persistence in static and mobile sensor networks,” in *Proceedings of International Workshop on Wireless Networks: Communication, Cooperation and Competition*, 2007.

[92] A. Oka and L. Lampe, “Data extraction from wireless sensor networks using distributed fountain codes,” *IEEE Transactions on Communications*, vol. 57, pp. 2607 – 2609, 2009.

[93] M. Rossi, G. Zanca, L. Stabellini, and R. Crepaldi, “Synapse: A network reprogramming protocol for wireless sensor networks using fountain codes,” in *Proceedings of 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks.* IEEE, 2008.

[94] Y. Lin, B. Li, and B. Liang, “Differentiated data persistence with priority random linear codes,” in *Proceedings of 27th International Conference on Distributed Computing Systems (ICDCS 07)*, 2007.

[95] P. Maymounkov, N. Harvey, and D. Lun, “Methods for efficient network coding,” in *Proceedings of 44th Ann. Allerton Conf. Comm.Control and Computing*, 2006.

[96] A. Jiang, “Network coding for joint storage and transmission with minimum cost,” in *Proceedings of IEEE International Symposium on Information Theory*, 2006.

[97] C. Gkantsidis and P. Rodriguez, “Network coding for large scale content distribution,” in *Proceedings of INFOCOM 05*, 2005.

[98] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *Proceedings of IEEE INFOCOM 2007*. IEEE, 2007.

[99] H. Tian, H. Shen, and T. Matsuzawa, "Randomwalk routing for wireless sensor networks," in *Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies.* IEEE, 2007.

[100] R. Beraldi, R. Baldoni, and R. Prakash, "A biased random walk routing protocol for wireless sensor networks: The lukewarm potato protocol," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 1649 – 1661, 2010.

[101] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Mixing times for random walks on geometric random graphs," in *Proceedings of SIAM Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, 2005.

[102] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equations of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, pp. 1087 – 1092, 1953.

[103] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, vol. 46, pp. 667 – 689, 2004.

[104] C. Avin and G. Ercal, "On the cover time of random geometric graphs," in *Proceedings of 32nd International Colloquium of Automata, Languages and Programming, ICALP05*, 2005.

[105] L. Filipponi, S. Santini, and A. Vitaletti, "Data collection in wireless sensor networks for noise pollution monitoring," in *Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems.* IEEE, 2008.

[106] P. V. Raju, R. V. R. S. Aravind, and B. S. Kumar, “Pollution monitoring system using wireless sensor network in visakhapatnam,” *International Journal of Engineering Trends and Technology (IJETT)*, vol. 4, pp. 591 – 595, 2013.

[107] K. K. Khedol, R. Perseedoss, and A. Mungur, “A wireless sensor network air pollution monitoring system,” *International Journal of Wireless and Mobile Networks (IJWMN)*, vol. 2, pp. 31 – 45, 2010.

[108] A. Traille, M. M. Tentzeris, S. Bouaziz, P. Pons, and H. Aubert, “A novel wireless passive temperature sensor utilizing microfluidic principles in millimeter-wave frequencies,” in *Proceedings of IEEE Sensors.* IEEE, 2011.

[109] C. Lin, P. Chou, and C. Chou, “Hcdd: hierarchical cluster-based data dissemination in wireless sensor networks with mobile sink,” in *Proceedings of IWCMC ’06*, 2006.

[110] E. Suli and D. Mayers, *An Introduction to Numerical Analysis.* Cambridge University Press, 2003.

[111] J. Gentle, *Numerical Linear Algebra for Applications in Statistics.* Springer, 1998.

[112] S. Lindsey and C. Raghavendra, “Pegasis: Power-efficient gathering in sensor information systems,” in *IEEE In Aerospace Conference Proceedings.* IEEE, 2002, pp. 1125–1130.

[113] A. Rahmati and L. Zhong, “Context-for-wireless: context-sensitive energy-efficient wireless data transfer,” in *Proceedings of MobiSys 2007*, 2007.

[114] X. Ye, J. Li, L. Xu, and M. Guizani, “A novel data collection scheme with cluster-based separate network coding for wsns,” in *Proceedings of Computing, Communications, and Applications Conference (ComComAp 2012).* IEEE, 2012.

[115] R. Liu, K. Wang, R. Jan, Y. Hu, and T. Ku, "An efficient cluster-based data dissemination scheme in wireless sensor networks," in *Proceedings of IEEE Vehicular Technology Conference (VTC Spring).* IEEE, 2011.

[116] O. Younis and S. Fahmy, "Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[117] J. Elson and D. Estrin, "Sensor networks : a bridge to the physical world," *Wireless Sensor Networks*, pp. 3 – 20, 2004.

[118] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," in *Proceedings of the IEEE.* IEEE, 2011.

[119] V. Kher and Y. Kim, "Securing distributed storage: Challenges, techniques, and systems," in *Proceedings of the Workshop on Storage Security and Survivability (StorageSS)*, 2005.

List of Publications

1. **Xiucai Ye** and Jie Li, "Distributed Separate Coding: Concept, Method, and Performance for Continuous Data Collection in Wireless Sensor Networks with a Mobile Base Station", *ACM Transactions on Sensor Networks (ACM TOSN)*, 2014, accepted.

2. Jie Li , **Xiucai Ye**, Li Xu, and Huaibei Liu, "Collecting all data continuously in wireless sensor networks with a mobile base station," *International Journal of Computational Science and Engineering (IJCSE)*, 2013, accepted.

3. **Xiucai Ye**, Jie Li, Li Xu, "Group Data Collection in Wireless Sensor Networks with a Mobile Base Station," *Proceedings of the 2013 IEEE Wireless Communication and Networking Conference (IEEE WCNC)*, pp. 1151-1156, April 7 - 10, 2013, Shanghai, China.

4. Jie Li, **Xiucai Ye**, and Li Xu, "A Novel Data Collection Scheme for WSNs," *Proceedings of 2012 IEEE 75th Vehicular Technology Conference (IEEE VTC 2012 Spring)*, pp. 1-5, Yokohama, Japan, May 6-9, 2012.

5. **Xiucai Ye**, Jie Li, Li Xu, and Mohsen Guizani, "A Novel Data Collection Scheme with Cluster-based Separate Network Coding for WSNs," *Proceedings of Computing, Communications, and Applications Conference (ComComAp 2012)*, pp. 18-23, HKUST, Hong Kong, China, January 11-13, 2012.

6. Jie Li, **Xiucai Ye**, and Yusheng Ji“A Novel Network Coding Scheme for Data Collection in WSNs with a Mobile BS,” *Proceedings of the 7th International Workshop on Databases in Networked Information Systems (DNIS 2011)*, Springer Lecture Notes in Computer Science (LNCS), No. 7108, pp. 296-311, Aizu-Wakamatsu, Japan, December 12-14, 2011.