Distributed data collection and storage algorithms for collaborative learning vision sensor devices with applications to pilgrimage

Salah A. Aly
Center of Research Excellence in Hajj and Umrah (HajjCore), The Custodian of the Two Holy Mosques Institute of Hajj Research, College of Computer and Information Systems, Umm Al-Qura University, Makkah, Saudi Arabia
Email: salahaly@uqu.edu.sa

Abstract: This work presents novel distributed data collection and storage algorithms for collaborative learning Wireless Sensor Networks (WSNs). In a large WSN, consider \( n \) sensor devices distributed randomly to acquire information and learn about a certain field. Such sensors have less power, small bandwidth, and short memory, and they might disappear from the network after certain time of operations. We propose two Distributed Data Storage Algorithms (DSAs), denoted by DSA-I and DSA-II, to solve this problem. In DSA-I, where the value of \( n \) is known for each learning sensor, we show that this algorithm is efficient in terms of the encoding/decoding operations. Furthermore, each node uses network flooding to disseminate its data throughout the network using mixing time approximately \( O(n) \). In DSA-II, it is assumed that dissemination of the data does not depend on the total number of network nodes, we show that the encoding operations take \( O(\mu^2) \), where \( \mu \) is the mean degree of the network graph and \( C \) is a system parameter. Performance of these two algorithms matches the derived theoretical results. Finally, these two algorithms can be used for monitoring and measuring certain phenomenon in camp tents located in the Minna field in south-east side of Makkah.

Keywords: data collection; storage algorithms; WSNs coverage; vision sensor devices; IT pilgrimage applications.


Biographical notes: Salah A. Aly (IEEE’02, ACM’03) received his PhD degree in Computer Science from Faculty of Engineering at Texas A&M University in May 2008. He worked in a research internship at Bell-Labs; Alcatel-Lucent in New Jersey, and as a Research Associate at Princeton University, USA. He serves as a regular reviewer for several IEEE/ACM conferences and journals including ISIT, ICC, Globecom, LCN, IEEE Transaction on Information Theory, JSAC, QIP, and International Journal of Quantum Information. His research interests include information security, storage algorithms, wireless sensor networks, and quantum information processing.

1 Introduction

The field of information technology has witnessed remarkable extensions especially after appearance of the World Wide Web two decades ago. In addition, this has been embarked by appearance of several communication networking branches, such as Wireless Sensor Networks (WSNs). WSNs consist of small devices (nodes) with low CPU power, small bandwidth, and limited memory. They can be deployed in isolated, tragedy, and obscured fields to monitor objects, detect fires or floods, measure temperatures, transmit media streams, etc. They can also be used in areas where human involvement is difficult to reach or it is danger for human being to be involved. There has been extensive research work on sensor networks to improve their services, powers, and operations (Stojmenovic, 2005). They have taken much attention recently due to their varieties of applications. We consider a model for large-scale WSNs where \( n \) data collection and storage sensor nodes are distributed uniformly and randomly in a field. These \( n \) nodes are deployed to collect information and transmit media streams (images, videos, texts) about a certain field, see Figure 1 and Aly (2012). These \( n \) sensor devices have a short time-to-live, limited memory, and might disappear from the network at anytime. Also, the nodes do not know locations of the neighbouring nodes, and they do not maintain routing tables to forward messages. We assume that the \( n \) sensing and data collection nodes generate independent packets that can be classified as initial or update packets sent at an arbitrary time. A packet initiated from a node \( u \) contains its \( ID_u \), time-to-live parameter, and sensed data. In addition, each storage node \( u \)
has a buffer size $M$ that can be divided into $m$ small buffers to save other neighbours’ data. Every storage node decides randomly and independently from which it will accept or reject packets. Also, a packet will be discarded once it travels through the network $O(n)$.

**Figure 1** A WSN consists of various collaborative sensor devices that are distributed randomly to monitor, collect data, and learn about Minna field in the East-South of Makkah. Approximately 50,000 camp tents are located in Minna to accommodate 3–5 million people for 4–8 days during the pilgrimage season, according to 2010 KSA statistics (see online version for colours).

The goal of this work is to develop an efficient method to randomly distribute and collect information from $n$ sensors to all $n$ storage nodes. In this case, a data collector with a high computational power can query any $(1 + e) n / m$ storage nodes for $e > 0$, and easily retrieve information about the $n$ sensor nodes with a high probability. Other versions of this problem has been solved by using coding in a centralised way (e.g. Fountain codes, MDS and linear codes) by adding some redundancy, where a node can send its data to a preselected set of other nodes in the network (Dimakis et al., 2006b; Lin et al., 2007b; Dimakis et al., 2008; Aly et al., 2011). Over a distributed random network, this is unreliable since we still need to find a strategy to distribute the information from the sources to a set of arbitrary storage nodes. Hence, a decentralised way solution is needed where the data collector and storage nodes are distributed randomly and independently. Therefore, the considered problem is a network storage rather than a network transmission problem. The later problem assumes that channel coding and modulation theory are used to handle the transmission from a source to a destination. The former problem requires distributed networking storage algorithms to assure protection of information against node failures or disappearance. It is assumed that all nodes trust each other data, and attackers are unable to break the nodes transitions. The motivations for this work are following:

1. We demonstrate a realistic model for WSNs, where nodes are distributed randomly with limited power and memory.
2. The encoding and decoding operations are done linearly.
3. Querying only $(1 + e) n / m$ a subset of the network nodes reveals information about all nodes.
4. The proposed storage algorithms have less computational complexity in comparison to the related work shown in Section 9.

This work is organised as follows. In Section 2 we introduce the network model. In Sections 3 and 5 we propose two storage algorithms and their analysis are provided in Sections 4 and 6, respectively. In Section 7 we present simulation studies of the proposed algorithms, and provide practical aspects in Section 8. In Section 9 we present a background and short survey of the related work. Finally, the work is concluded in Section 10.

**2 Network model and assumptions**

In this section we present the network model and problem definition. Consider a WSN $\mathcal{N}$ with $n$ sensor nodes that are uniformly distributed at random in a region $A = [L, L]^2$ for some integer $L \geq 1$. The network model $\mathcal{N}$ can be considered as an abstract graph $G = (V, E)$ with a set of nodes $V$ and a set of edges $E$. The set $V$ represents the sensors $S = \{s_1, s_2, \ldots, s_n\}$ that will measure information about a specific field, a sensor device is shown in Figure 2. Also, $E$ represents a set of connections (links) between the sensors $S$. Two arbitrary sensors $s_i$ and $s_j$ are connected if they are in each other transmission range.

**Figure 2** A wireless sensor device equipped with several sensor components to measure temperature, gas pollution, and CO$_2$ (see online version for colours).

We ensure that the network is dense, meaning with high probability there are no isolated nodes. Let $r > 0$ be a fraction, we say that two nodes $u$ and $v$ in $V$ are connected in $G$ if and only if the distance between them is bounded by the design parameter $r$, i.e. $0 < d(u, v) \leq r$. Put differently, let $z$ be a random variable represents existence of an edge between any two arbitrary nodes $u$ and $v$. Then

$$z = \begin{cases} 1 & d(u, v) \leq r \\ 0 & \text{otherwise} \end{cases}$$

(1)

One can guarantee such condition by assuming that the radius $r \geq O\left(\frac{1}{n^2}\right)$. 

2.1 Assumptions

We have the following assumptions about the network model $N$:

1. Let $S = \{s_1, \ldots, s_n\}$ be a set of sensing nodes that are distributed randomly and uniformly in a field. Also, they are the set of storage nodes. So, this assumption differentiates between our work and the problems considered in the work of Aly et al. (2008) and Lin et al. (2007b).

2. Every node does not maintain routing or geographic tables, and the network topology is not known. Every node $s_i$ can send a flooding message to the neighbouring nodes. Also, every node $s_i$ can detect the total number of neighbours by sending a simple flooding query message, and whoever replies to this message will be a neighbour of this node, see Figure 3. Therefore, our work is more general and different from the work done by Dimakis et al. (2005, 2007). The degree $d(u)$ of this node is the total number of neighbours with a direct connection.

3. Every node has a buffer of size $M$ and this buffer can be divided into smaller buffers, each of size $c$, such that $m = \lceil M/c \rceil$. Hence, all nodes have the same number of buffers, see Figure 4. Also, the first buffer of a node $u$ is reserved for its own sensing data.

4. Every node $s_i$ prepares a packet $\text{packet}_{s_i}$ with its ID, sensed data $x_i$, counter $c(x_i)$, and a flag that is set to zero or one.

\[ \text{packet}_{s_i} = (ID_{s_i}, x_i, c(x_i), \text{flag}) \]  

(2) The flag is set to zero when the sensors initiate data for the first time, otherwise it will be set to one for data update.

5. Every node draws a degree $d_u$ from a degree distribution $\Omega$. If a node decided to accept a packet, it will also decide on which buffer it will be stored.

When a node $s_i$ receives a packet, it will decide to either reject or accept it with a certain probability.

Figure 3 A WSN with $n$ nodes arbitrary and randomly distributed in a field. A node $s_i$ determines its degree $d(s_i)$ by sending a flooding message to the neighbouring nodes (see online version for colours)

3 Distributed storage algorithms

In this section we will present a networked DSA for WSNs and study its encoding and decoding operations. Other previous algorithms assumed that $k$ source nodes disseminate their sensed data throughout a network with $n$ storage nodes using the means of Fountain codes and random walks. However, in this work we generalise this scenario where a set of $n$ sources disseminate their data to a set of $n$ storage nodes. Also, in this proposed algorithm we use properties of WSNs such as broadcasting and flooding.

3.1 Encoding operations

We present a DSA-I for WSNs. DSA-I algorithm consists of three main steps: initialisation, encoding/flooding, and storage phases. Each phase can be described as follows.

1. Initialisation phase: Every node $s_i$ in $S$ has an ID $s_i$ and reading (sensing) data $x_i$. The node $s_i$ in the initialisation phase prepares a packet along with its info, a counter $c(x_i)$ that determines the maximum number of hops that will receive $x_i$, and a flag that is set to zero. We ensure that every message $x_i$ will have its own threshold value $c(x_i)$ set by the sender $s_i$ based on the set of neighbours $N(s_i)$. This value will depend on the degree $d(s_i)$. If the node $s_i$ has a few neighbours, then $c(x_i)$ will be large. Also, a node with large number of neighbours will choose a small counter $c(x_i)$. This means that every node will decide its own counter.

\[ \text{packet}_{s_i} = (ID_{s_i}, x_i, c(x_i), \text{flag}) \]  

(3) The node $s_i$ broadcasts this packet to all neighbouring nodes $N(s_i)$.

2. Encoding and flooding phase

- After the flooding phase, every node $u$ receiving the packet $s_i$ will accept the data $x_i$ with probability one and will add this data to its buffer $y$.

\[ y_u^i = x_u^i \oplus x_i. \]  

(4)
• The node \( u \) will decrease the counter by one as
\[
c(x_i) = c(x_i) - 1
\] (5)

• The node \( u \) will select a set of neighbours that did not receive the message \( x_i \) and it will send this message using multicasting.

• For an arbitrary node \( v \) that receives the message from \( u \), it will check if the \( x_i \) has been received before, if yes, then it will discard it. If not, then it will run a probability distribution whether to accept or reject it. If accepted, then it will add the data to its buffer \( y_v^+ = y_v \oplus x_i \) and will decrease the counter \( c(x_i) = c(x_i) - 1 \).

• The node \( v \) will check if the counter is zero, otherwise it will decrease it and send this message to the neighbouring nodes that did not receive it using multicasting.

3 Storage phase: Every node will maintain its own buffer by storing a copy of its data and other nodes’ data. Also, a node will store a list of nodes IDs of the packets that reached it. After all nodes receive, send and storage their own and neighbouring data, every node will be able to maintain a buffer with some data of the network nodes.

Algorithm 1  DSA-I Algorithm: DSA for a WSN where the data are disseminated using multicasting and flooding to all neighbours

**Input:** A sensor network with \( S = \{s_1, \ldots, s_n\} \) source nodes, \( n \) source packets \( x_i, \ldots, x_s \) and a positive constant \( c(s) \).

**Output:** storage buffers \( y_1, y_2, \ldots, y_n \) for all sensors \( S \).

**foreach node** \( u = 1:n \) **do**

  Generate \( d_u(s) \) according to \( \Omega_u(d) \) (or \( \Omega_u(d) \)) and a set of neighbours \( N(u) \) using flooding, see Appendix A;  

**end**

**foreach source node** \( s_i, i = 1:n \) **do**

  Generate header of \( x_i \) and \( \text{token} = 0 \);

  Set counter \( c(x_i) = \lfloor n/d(s_i) \rfloor \);

  Flood \( x_i \) to all \( N(s_i) \) uniformly at random,

  Send \( x_i \) to \( u \in N(s_i) \);

  with probability \( 1, y_u = y_u \oplus x_i \);

  Put \( x_i \) into \( u \)'s forward queue;

  \( c(x_i) = c(x_i) - 1 \);

**end**

**while packets remaining do**

**foreach node** \( u \) **receives packets before current round** **do**

  Choose \( v \in A(u) \) uniformly at random;

  Send packet \( x_i \) in \( u \)'s forward queue to \( v \);

  **if** \( v \) **receives** \( x_i \) **for the first time** **then**

    \( \text{coin} = \text{rand}(1) \);

    flip a coin to accept or reject a packet;

    **if** \( \text{coin} \leq \frac{1}{d_i(v)} \) **then**

    \( y_u = y_u \oplus x_i \);

    Put \( x_i \) into \( v \)'s forward queue;

    \( c(x_i) = c(x_i) - 1 \);

  **else**

    **if** \( c(x_i) \geq 1 \) **then**

    Put \( x_i \) into \( v \)'s forward queue;

    \( c(x_i) = c(x_i) - 1 \);

  **else**

    Discard \( x_i \);

  **end**

**end**

**end**

3.2 Decoding operations

The stored data can be recovered by querying a number of nodes from the network. Let \( n \) be the total number of alive nodes; assume that every node has \( m \) buffers such that \( M/c \) is the number of buffers, where \( c \) is a small buffer size, and \( M \) is total buffer size by a node. Then the data collector needs to query at least \((1 + \epsilon)n/m\) nodes in order to retrieve the information about the \( n \) variables.

4 DSA-I analysis

We shall provide analysis for the DSA-I algorithm shown in the previous section. The main idea is to utilise flooding and the node degree of each node to disseminate the sensed data from sensors throughout the network. We note that nodes with large degree will have smaller counters in their packets such that their packets will travel for minimal number of neighbours. Also, nodes with smaller degree will have larger counters such that their packets will be disseminated to many neighbours as possible.

The following lemma establishes the number of hobs (steps) that every packet will travel in the network.

Lemma 1: On average with a high probability, the total number of steps for one packet originated by a node \( u \) in one branch in DSA-I is given by

\[ O(n / \mu) \] (6)
Proof: Let \( u \) be a node originating a packet \( \text{packet}_u \) and it has degree \( d(u) \). For any arbitrary node \( v \), the packet \( \text{packet}_u \) will be forwarded only if it is the first time to visit \( v \) or the counter \( c(x_v) \geq 2 \). We know that every packet originated from a node \( u \) has a counter given by

\[
c(x_v) = \left\lfloor \frac{n}{d(u)} \right\rfloor
\]  

(7)

Let \( \mu \) be the mean degree of an abstract graph representing the network \( N \), see Definition A1 in Appendix. On average assuming every packet will be sent to \( \mu \) neighbouring nodes. Approximating the mean degree of the graph to the degree of any arbitrary node \( u \), the result follows.

The previous lemma ensures that if \( d(u) > n/2 \), then the node \( u \) will flood its packet only once \( c(u) = 1 \). In addition, nodes with smaller degrees will require to send their packets using large number of steps.

If the total number of nodes is not known, one can use a random walk initiated by the node \( u \) to estimate the total number of nodes. In Section 5, we will propose different algorithm that does not depend on estimating \( n \) or use random walks in a graph.

The following lemma shows the total number of transmissions required to disseminate the information throughout the network.

Lemma 2: Let \( N \) be an instance model of a WSN with \( n \) sensor nodes. The total number of transmissions required to disseminate the information from any arbitrary node throughout the network is given by

\[
O(n)
\]  

(8)

Proof: Let \( d(S) \) be the degree (number of neighbours with a direct connection) of a sensor node \( s \). On average \( \mu \) is the mean degree of the set of sensors \( S \) approximated to

\[
\frac{1}{n} \left( \sum_i^n d(S_i) \right)
\]

Every node does flooding that takes \( O(1) \) running time to \( d(S) \) neighbours. In order to disseminate information from a sensor \( s_i \), at least \( n/\mu \) steps are needed using Lemma 1. Also, every sensor \( s_i \) needs to send \( \mu \) messages on average to the neighbours. Hence the result follows.

The following theorem shows the encoding complexity of DSA-I algorithm.

Theorem 1: The encoding operations of DSA-I algorithm are the total number of transmissions required to disseminate information sensed by all nodes that is given by

\[
O(n^2)
\]  

(9)

5 DSA-II algorithm without knowing global information

In algorithm DSA-I we assumed that the total number of nodes are known in advance for each sensing storing node in the network. This might not be the case since arbitrary nodes might join and leave the network at various times due to the fact that they have limited CPU and short life time. Therefore, one needs to design network storage algorithm that does not depend on the value of total number of nodes.

In this section we will develop a DSA-II that is totally distributed without knowing global information. The objective is that each node \( u \) will estimate a value for its counter \( c(u) \); the number of steps in which each packet will be disseminated in the network. In DSA-II each node \( u \) will first perform an inference phase that will calculate value of the counter \( c(u) \). This can be achieved using the degree of \( u \) and the degrees of the neighbouring nodes \( N(u) \). We also assume a system parameter \( c_u \) that will depend on the network condition and node’s degree.

Inference phase: Let \( u \) be an arbitrary node in a distributed network \( N \). In the inference phase, each node \( u \) will dynamically determine value of the counter \( c(u) \). The node \( u \) knows its neighbours \( N(u) \). This is achieved in the flooding phase. Furthermore, the node \( v \) in \( N(u) \) knows the degrees of its neighbours.

The inference phase is done dynamically in a sense that every node in the network will separately decide a value for its counter. Nodes with large degrees will have a high chance of forwarding their data throughout the network to a large number of nodes.

Then encoding operations of DSA-II algorithm are similar to DSA-I algorithm except the former utilises an inference phase, where the number of forwarding steps are predetermined first. Assume \( v \) be a node connected to a source node \( u \). Let \( h_\ell \) be the degree of a node \( v \) without adding nodes in \( N(u) \) \( \cup \) \( u \). We can define the counter \( c(u) \) as

\[
c(u) = c_u \left[ \frac{1}{d(u)} \sum_{v \in N(u)} h_\ell \right]
\]  

(10)

Algorithm 2 DSA-II algorithm: DSA for a WSN without knowing global information where the data are disseminated using multicasting and flooding to all neighbours

**Input:** A sensor network \( N \) with

\[
S = \{s_1, \ldots s_i, \ldots\} \text{ source nodes, source packets } x_1, x_2, \ldots
\]

**Output:** storage buffers \( y_1, y_2, \ldots, y_n \) for all sensors \( S \).

**foreach node \( u \) in \( N \) do**

- determine a set of neighbours \( N(u) \) using flooding;
- determine a system parameter \( c_u \);

**end**

**Inference Phase**
foreach source node $u$ in $N$ do
    query the neighbors $A(u)$ of $s$, for their degrees;
    Let $v \in A(u)$ and $b_v$ be the $v$ degree without adding nodes in $A(u) \cup u$;
    if $dv = 1$ then
        Repeat inference phase at $v$;
        Repeat until $b_v \neq 1$ for some $v' \in A(v)$;
        Put $b_v = \sum_{v'} dv'$
    end
    $c(u) = c_s \left[ \frac{1}{d(u)} \sum_{v \in N(u)} b_v \right]$;
end
foreach source node $s$, in $N$ do
    Generate header of $x_s$ and $token = 0$;
    flood $x_s$ to all $A(s)$ uniformly at random, send $x_s$ to $u \in A(s)$;
    with probability 1, $x_s = x_s \oplus x_u$;
    Put $x_s$ into $u$’s forward queue;
    $c(x_s) = c(x_s) - 1$;
end
while source packets remaining do
    Run the encoding and flooding phase in DSA-I alg.;
end

Encoding and flooding phase:

- After the inference and initialisation phases, every node $u$ receiving the packet $y_u$ will accept the data $x_u$ with probability one and will add this data to its buffer $y$.

  $$y_u = y_u \oplus x_u.$$  \hfill (11)

- The node $u$ will decrease the counter by one as
  $$c(x_u) = c(x_u) - 1.$$  \hfill (12)

- The node $u$ will select a set of neighbours that did not receive the message $x_u$ and it will send this message using multicasting.

- For an arbitrary node $v$ that receives the message from $u$, it will check if the $x_u$ has been received before, if yes, then it will discard it. If not, then it will run a probability distributed whether to accept or reject it. If accepted, then it will add the data to its buffer $y_v = y_v \oplus x_u$ and will decrease the counter $c(x_v) = c(x_v) - 1$.

- The node $v$ will check if the counter is zero, otherwise it will decrease it and send this message to the neighbouring nodes that did not receive it.

Storage phase: Every node will maintain its own buffer by storing a copy of its data and other nodes’ data. Also, a node will store a list of nodes IDs of the packets that reached it. After all nodes receive, send and store their own and neighbours’ data, every node will be able to maintain a buffer with some data of the network nodes.

6 DSA-II analysis

We will also provide analysis for the DSA-II algorithm shown in the previous section. The main idea is to utilise flooding and the node degree to disseminate the sensed data from sensors throughout the network. We ensure that nodes with large degree will have smaller counters in their packets such that their packets will travel for minimal number of hops. Also, nodes with smaller degree will have larger counters such that their packets will travel to many neighbours as possible.

The following lemma establishes the number of hops (steps) that every packet will travel in the network.

Lemma 3: On average for a uniformly distributed network, the total number of steps for one packet originated by a node $u$ in one branch in DSA-II is given by

$$O(\mu - \lambda).$$  \hfill (13)

Proof: Let $u$ be a node originating a packet $packet_u$ and it has degree $d(u)$ and when the nodes are uniformly distributed in the network we can approximate $d(u)$ as $\mu$. We know that every packet originated from a node $u$ has a counter given by

$$c(u) = c_s \left[ \frac{1}{d(u)} \sum_{v \in N(u)} b_v \right].$$  \hfill (14)

We ensure that $c_s$ is inversely proportional to node degree so that nodes with small number of neighbours take large values of $c_s$ and vice versa. Also in case that node $v$ has only one neighbour other than the originating node $u$ we traverse through this node until we get at least one node $v'$ that has degree $b' > 1$.

On average assuming every packet will be sent to $\mu$ neighbouring nodes. We can approximate $b_v$ as $\mu - \lambda$ so we can rewrite the equation $\sum_{v \in N(u)} b_v / d(u)$ as $\lambda / (\mu - \lambda) / \mu$. For any arbitrary node $v$, the packet $packet_v$ will be forwarded only if it is the first time to visit $v$ or the counter $c(x_v) \geq 2$.

The following lemma shows the total number of transmissions required to disseminate the information throughout the network.
Lemma 4: Let \( \mathcal{N} \) be an instance model of a WSN with \( n \) sensor nodes uniformly distributed. The total number of transmissions required to disseminate the information from any arbitrary node throughout the network is given by

\[
O\left(\mu (\mu - \lambda)\right).
\]

Proof: Let \( d(s) \) be the degree (number of neighbours with a direct connection) of a sensor node \( s \). On average \( \mu \) is the mean degree of the set of sensors \( S \) approximated to \( \frac{1}{n} \left( \sum_{i=1}^{n} s(s_i) \right) \).

Every node does flooding that takes \( O(1) \) running time to \( d(s) \) neighbours. In order to disseminate information from a sensor \( s \), at least \( \mu - \lambda \) steps are needed using Lemma 3. Also, every sensor \( s_i \) needs to send \( \mu \) messages on average to the neighbours. Hence the result follows.

The following theorem shows the encoding complexity of DSA-I algorithm.

Theorem 2: The encoding operations of DSA-II algorithm are the total number of transmissions required to disseminate information sensed by all nodes and given by

\[
O\left(\mu (\mu - \lambda) n\right).
\]

7 Performance and simulation results

In this section we will simulate the DSAs, DSA-I and DSA-II, presented in the previous sections. The main performance metric we investigate is the successful decoding probability versus the decoding ratio.

Let \( \rho \) be the successful decoding probability defined as percentage of \( M \), successful trials for recovering all \( n \) variables (symbols) to the total number of trials. Also, let \( h \) be the total number of queries needed to recover those \( n \) variables. We can define the decoding ratio as the total queried nodes divided by \( n \), i.e. \( h/n \).

Definition 1: (decoding ratio) Decoding ratio \( \eta \) is the ratio between the number of querying nodes \( h \) and the number of sources \( n \), i.e.

\[
\eta = \frac{h}{n}.
\]

Definition 2: (successful decoding probability) Successful decoding probability \( P_s \) is the probability that the \( n \) source packets can be recovered from the \( h \) querying nodes.

In our simulation, \( P_s \) is evaluated as follows. Suppose the network has \( n \) nodes, and we query \( h \) nodes. There are \( \binom{n}{h} \) ways to choose such \( h \) nodes, we pick a set \( S \) of these choices uniformly at random. The set \( S \) was chosen large enough to give more normal results. So given the set \( S \) which is a ratio \( 0 < r < 1 \) of all possible combinations we define \( M \) as follow:

\[
M = r \cdot \binom{n}{h} = r \cdot \frac{n!}{h!(n-h)!}.
\]

Let \( M \) be the size of the subset these \( M \) choices of \( h \) query nodes from which the all \( n \) source packets can be recovered. Then, we evaluate the successful decoding probability as

\[
P_s = \frac{M}{M}
\]

We ran the experiment over a network with area \( A = [L, L]^2 \) and with different node densities. We evaluated the performance with various decoding ratios depending on the total number of nodes inside the network with incremental step \( \eta = 0.1 \).

For a decoding ratio \( \eta \) we select \( h \) nodes for our test. So we may have a large number of combinations to choose from, which may get order of \( 100^{100} \) combinations. So, we have to choose a fair portion \( r \) of these combinations \( N \ll r \ll M \) and average the results over these experiments.

Figure 5 shows the decoding performance of DSA-I algorithm with Ideal Soliton distribution with small number of nodes. We ran the experiment over a network with area \( A = [2, 2]^2 \) and with a node density \( 2.5 \leq \lambda \leq 12.5 \). We evaluated the performance with various decoding ratio \( 0.1 \leq \eta \leq 1 \) with incremental step \( \eta = 0.1 \).

From these results we can see that the successful decoding probability increases as the node density increases while the decoding ratio \( \eta \) is kept constant. We can deduce that the successful decoding probability is above 70% when the decoding ratio is about 20%–30%. Another observation is that with a node density \( \lambda > 8 \), the successful decoding probability \( P_s > 90% \).

Figure 6 shows the decoding performance of DSA-I algorithm with Ideal Soliton distribution with medium number of nodes. The network is deployed in \( A = [5, 5]^2 \) with node density \( \lambda \) ranges from 4 to 20. From the simulation results we can see that the decoding ratio increases with the increase of \( \lambda \) and approaches to 1 for \( \eta > 20% \) and \( \lambda \geq 12 \).
Figure 6 A WSN with \( n \) nodes arbitrary and randomly distributed in a field. The successful decoding ratio is shown for various values of \( n = 200, 400, 600 \) with the DSA-I algorithm.

Figure 7 shows the decoding performance of DSA-II algorithm with Ideal Soliton distribution with small number of nodes. We ran the first experiment over a network with area \( A = [2, 2]^2 \) and with a node density \( 2.5 \leq \lambda \leq 12.5 \), and evaluated the performance with various decoding ratio \( 0.1 \leq \eta \leq 1 \) with incremental step \( = 0.1 \), as shown in the figure the DSA-II algorithm archived similar results to the DSA-I algorithm with a successful decoding probability \( P_s > 70\% \) for a decoding ratio \( \eta \geq 0.4 \).

Figure 7 A WSN with \( n \) nodes arbitrary and randomly distributed in a field. The successful decoding ratio is shown for various values of \( n = 30, 40, 50 \) with the DSA-II algorithm.

Figure 8 shows that a caparison between the buffer size in DSA-I and DSA-II in a network deployed in an area \( A = [5, 5]^2 \), it can be concluded from the results that the buffer size approximately equals 10\% of the network size \( n \). From Figure 8 it can be seen that the buffer size is strongly related to the network density \( \lambda \).

Figure 8 A Caparison between DSA-I and DSA-II buffer size for various node densities in a medium size network. Increasing number of sensor nodes increases linearly the number of buffers.

8 Evaluation and practical aspects

In this section we shall provide evaluation and comparison analysis between DSA-I and DSA-II algorithms and related work in DSAs. Previous work focused on utilising random walks and fountain codes to disseminate data sensed by a set of sensors throughout the network. Also, global and geographical information such as knowing the total number of nodes, routing tables, and node locations are used. In this work we do not assume knowing such global information.

The main goal of this work is to design data collection algorithms that can be utilised in large-scale WSNs. We achieve this goal by disseminate data throughout the network using data flooding once at every sensor node, then adding some redundancy at other neighbouring nodes using random walks and packet trapping. Every storage node will keep track of other node’s IDs, from which it will accept/reject packets.

The main advantages of the proposed algorithms are as follows:

1. One does not need to query all nodes in the network to retrieve information about all \( n \) nodes. Only 20\%–30\% of the total nodes can be queried.
2. One can query only one arbitrary node \( u \) in a certain region in the network to obtain the information about this region.

8.1 Sensing new data

The proposed algorithms work also in the case of data update. Assume a node \( u \) sensed data \( x_u \) and it has been disseminated throughout the network using flooding as shown in DSA-I and DSA-II algorithms. In this case the flag value is set to zero; and a packet from the node \( u \) is originated as follows:

\[
packet_u = (ID_u, x_u, c(x_u), flag)
\]  

(20)
We notice that every node \( v \) stores a copy from this data \( x_u \) will also maintain a list of IDs including \( ID_u \).

Assume \( x'_u \) be the new sensed data from the node \( u \). Let us consider the case that the node \( u \) wants to update its values, then the node \( u \) will send update message setting the flag to one.

\[
\text{packet}_u = (ID_u, x'_u, \oplus x_u, c(x_u), \text{flag})
\]  

(21)

The new and old data are Xored in this packet. Every storage node will check the flag, whether it is an update or initial packet. Also, the node \( v \) will check if \( ID_u \) is in its own list. Once a node \( v \) accepts the coming update packet, it will update its target buffer as

\[
y'_v = y_v \oplus x'_u \oplus x_u
\]  

(22)

8.2 Practical aspects

The proposed algorithms can be deployed in large-scale WSNs, where geographic locations of sensor nodes are not known. Also, each sensor does not need to maintain routing tables about the neighbouring nodes. Such applications include WSNs disseminated in forests and burned fields, where monitoring and detecting fires, floods and disasters phenomena are required. It also can be deployed in crowd large fields, where a large number of nodes is scattered to collect data.

Figure 9: Wireless sensor devices are scattered in Minna field in the east of Makkah to gather and collect data about the environment. Such sensors are able to detect fires, gas pollution, and other disasters phenomena. They are needed to monitor the large number of camp tents in Minna field (see online version for colours).

The proposed data collection and storage algorithms certainly can be deployed in Minna and Arafat fields in the east south of Makkah during pilgrimage. Approximately 50,000 camp tents are located in Minna to accommodate 3–5 million people for 4–8 days during pilgrimage, according to 2010 KSA statistics. Figure 9 shows camp tents located in Minna field in the east of Makkah. The tents are supported by air-condition, electricity, and gas suppliers. The sensor devices are distributed randomly to measure gas pollution, detect fires, collect data and learn about the environment. The data storage devices receive collected data by the sensors and send it to the main server for further analysis. More details and practical aspects of this model will be explained in our future work.

9 Related work

Wireless vision sensor networks are small devices that can be scattered in a field or deployed in a network to measure certain phenomena. In this section, we present the previous work in network storage codes that is relevant to our work. Distributed network storage codes such as Fountain codes are used along with random walks to distribute data from a set of sources \( k \) to a set of storage nodes \( n \) (see Dimakis et al., 2006b; Aly et al., 2008). However, in this work we generalise this scenario where a set of \( n \) sources disseminate their data to a set of \( n \) storage nodes.

The most notable work in DSA for WSNs can be stated as:

- Dimakis el al. (2006a, 2006b, 2008) used a decentralised implementation of Fountain codes that uses geographic routing and every node has to know its location. The motivation for using Fountain codes instead of using random linear codes is that Fountain codes need \( O(k \log k) \) decoding complexity but random linear codes and RS codes use \( O(k^2) \) decoding complexity where \( k \) is the number of data blocks to be encoded. Also, one does not know in advance the degree \( d \) of the collector nodes (Lin et al., 2007a). The authors propose a randomised algorithm that constructs Fountain codes over grid network using only geographical knowledge of nodes and local randomised decisions. They also used fast random walks to disseminate source data to the storage nodes.

- Lin et al. (2007a, 2007b) studied the question “how to retrieve historical data that the sensors have gathered even if some sensors are destroyed or disappeared from the network?” They analysed techniques to increase ‘persistence’ of sensed data in a random WSN. They proposed two decentralised algorithms using Fountain codes to guarantee the persistence and reliability of cached data on unreliable sensors. They used random walks to disseminate data from a sensor (source) node to a set of other storage nodes. The first algorithm introduces lower overhead than naive random walk, while the second algorithm has lower level of fault tolerance than the original centralised Fountain code, but consumes much lower dissemination cost. They
proposed the first novel decentralised implementation of Fountain codes in sensor networks in an efficient and scalable fashion. The authors did not use routing tables to disseminate data from one sensor to a set of sensors. The reason is that a sensor does not have enough energy or memory to maintain a routing table which is scalable with the size of the network.

- Kamara et al. (2006) proposed a novel technique called growth codes to increase data persistence in WSNs, i.e. increasing the amount of information that can be recover at the sink. Growth codes are a linear technique that information is encoded in an online distributed way with increasing degree. They defined persistence of a sensor network as "the fraction of data generated within the network that eventually reaches the sink" (Kamra et al., 2006). They showed that growth codes can increase the amount of information that can be recovered at any storage node at any time period whenever there is a failure in some other nodes. They do not use robust or Soliton distributions; however, they propose a new distribution depending on the network condition to determine degrees of the storage nodes. The motivation for their work is that (a) positions of the nodes are not known, so a sensor node does not need to know positions of other nodes. (b) They assume a round time of update the nodes, meaning with increasing the time $t$, degree of a symbol is increased. This is the idea behind growth degrees. (c) They provide practical implementations of growth codes and compare its performance with other codes. (d) The decoding part is done by querying an arbitrary sink, if the original sensed data have been collected correctly then finish, otherwise query another sink node.

- The authors (Aly et al., 2008; Kong et al., 2010; Aly et al., 2011) studied a model for distributed network storage algorithms for WSNs where $k$ sensor nodes (sources) want to disseminate their data to $n$ storage nodes with less computational complexity. The authors used Fountain codes and random walks in graphs to solve this problem. They also assumed that the total number or sources and storage nodes are not known. In other words, they gave an algorithm where every node in a network can estimate the number of sources and the total number of nodes.

In this work we propose a different system for a WSN where all nodes act as sources as well as storage/receiver nodes (Aly et al., 2009). The encoding operations of a node to disseminate its data are linear and take less computational time in comparison to the previous work.

10 Conclusion

In this work we presented two DSAs for large-scale WSNs. Given $n$ storage nodes with limited buffers we demonstrated schemes to disseminate sensed data throughout the network with less computational overhead. The results and performance show that it is required to query only 20%–30% of the network nodes in order to retrieve the data collected by the $n$ sensing nodes, when the buffer size is 10% of the network size. Our future work will include practical and implementation aspects of these algorithms to better serve camp tents in Minna and Arafat fields located in the east south of Makkah, KSA.

Acknowledgements

This work is funded by a grant from the Long-Term National Plan for Science, Technology and Innovation (LT-NPSTI), No. 11-INF1708-10, the King Abdulaziz City for Science and Technology (KACST), Kingdom of Saudi Arabia. We thank the Science and Technology Unit at Umm A-Qura University for their continued logistics support.

References


Distributed data collection and storage algorithms


Appendix A

Given a network $N$, the mean degree of a node in $G$ can be defined as:

Definition A1: (Node Degree) Consider a graph $G = (V,E)$, where $V$ and $E$ denote the set of nodes and links, respectively. Given $u, v \in V$, we say $u$ and $v$ are adjacent (or $u$ is adjacent to $v$, and vice versa) if there exists a link between $u$ and $v$, i.e. $(u,v) \in E$. In this case, we also say that $u$ and $v$ are neighbours. Denote by $\mathcal{N}(u)$ the set of neighbours of a node $u$. The number of neighbours, with a direct connection, of a node $u$ is called the node degree of $u$, and denoted by $d(u)$, i.e., $|\mathcal{N}(u)| = d(u)$. The mean degree of a graph $G$ is given by

$$u = \frac{1}{|V|} \sum_{u \in V} d(u), \quad (A1)$$

where $|V|$ is the total number of nodes in $G$.

The Ideal Soliton distribution $\Omega_\alpha(d)$ for $k$ source blocks is given by

$$\Omega_\alpha(i) = \Pr(d = i) = \begin{cases} 
\frac{1}{k}, & i = 1 \\
\frac{1}{i(i-1)}, & i = 2, 3, \ldots, k 
\end{cases}, \quad (A2)$$

Let $R = c_0 \ln \left(k/\delta\right) \sqrt{k}$, where $c_0$ is a suitable constant and $0 < \delta < 1$.

The Robust Soliton distribution for $k$ source blocks is defined as follows. Define

$$\tau(i) = \begin{cases} 
\frac{R}{ik}, & i = 1, \ldots, \frac{k}{R} - 1 \\
\frac{R \ln(R/\delta)}{k}, & i = \frac{k}{R} \\
0, & i = \frac{k}{R} + 1, \ldots, k, 
\end{cases} \quad (A3)$$

and let

$$\beta = \sum_{i=1}^{k} \tau(i) + \Omega_\alpha(i). \quad (A4)$$

The Robust Soliton distribution is given by

$$\Omega_\beta(i) = \frac{\tau(i) + \Omega_\alpha(i)}{\beta}, \quad (A5)$$

for all $i = 1, 2, \ldots, k$. 