Data Compression Techniques in Wireless Sensor Networks

You-Chiun Wang

Abstract—*Wireless sensor networks (WSNs)* open a new research field for pervasive computing and context-aware monitoring of the physical environments. Many WSN applications aim at long-term environmental monitoring. In these applications, energy consumption is a principal concern because sensor nodes have to regularly report their sensing data to the remote sink(s) for a very long time. Since data transmission is one primary factor of the energy consumption of sensor nodes, many research efforts focus on reducing the amount of data transmissions through data compression techniques. In this chapter, we discuss the data compression techniques in WSNs, which can be classified into five categories: 1) The *string-based compression techniques* treat sensing data as a sequence of characters and then adopt the text data compression schemes to compress them. 2) The *image-based compression techniques* hierarchically organize WSNs and then borrow the idea from the image compression solutions to handle sensing data. 3) The *distributed source coding techniques* extend the Slepian-Wolf theorem to encode multiple correlated data streams independently at sensor nodes and then jointly decode them at the sink. 4) The *compressed sensing techniques* adopt a small number of nonadaptive and randomized linear projection samples to compress sensing data. 5) The *data aggregation techniques* select a subset of sensor nodes in the network to be responsible for fusing the sensing data from other sensor nodes to reduce the amount of data transmissions. A comparison of these data compression techniques is also given in this chapter.

Index Terms—compressed sensing, data compression, data aggregation, distributed source coding, Slepian-Wolf theorem, wavelet transformation, wireless sensor networks.

1 Introduction

A *wireless sensor network (WSN)* is composed of one or several remote sinks and a sheer number of sensor nodes. Each sensor node is a small, wireless device that can continually collect its surrounding information and report the sensing data to the sink(s) through a multi-hop ad hoc routing scheme [1]. WSNs provide a new opportunity for pervasive computing and context-aware monitoring of the physical environments. They are usually deployed in regions of interest to observe specific phenomena or track objects. Practical applications of WSNs include, for example, animal monitoring, agriculture transforming, health care, indoor surveillance, and smart buildings [2]–[6].

Because sensor nodes are usually powered by batteries and many WSN applications aim at long-term monitoring of the environments, how to conserve the energy of sensor nodes to extend their lifetimes is a critical issue. There are two common solutions to conserve the energy of sensor nodes. One solution is to take advantage of node redundancy by selecting a subset of sensor nodes to be active while putting others to sleep to conserve their energy [7]–[9]. The selected subset of active sensor nodes have to cover the whole monitoring region and maintain the network connectivity. In other words, these active sensor nodes have to make sure that the network still functions as well as the case when all sensor nodes are active. By selecting different subsets of sensor nodes to be active by turns, we can prevent some sensor nodes from consuming too much energy and thus extend the network lifetime. However, when node redundancy is not available (because of network deployment [10], [11] or sensor breakage, for example), such sleep-active mechanisms may not be applied.

Another solution is to reduce the amount of sensing data to be sent by sensor nodes, because transmission is one of the most energy-consuming operations of sensor nodes. Such a solution is especially useful when sensor nodes have to regularly report their sensing data to the sink(s) for a very long time. In order to reduce the amount of sensing data, we need to *compress* them inside the network. Depending on the recoverability of data, we can classify the data compression schemes into three categories: *lossless*, *loss*, and *unrecoverable*. A lossless compression means that after executing the decompression operation, we can obtain exactly the same data as those before executing the compression operation. Huffman coding [12] is one of the representative examples. A loss compression means that some detailed (and usually minor) features of data may be lost due to the compression operation. Most of the image and video compression schemes such as JPEG2000 [13] belong to this category. Finally, an unrecoverable compression means that the compression operation is irreversible. In other words, there is no decompression operation. For example, one can compress a set of numbers by taking their average value but each of the original numbers cannot be derived from this average value.

This chapter discusses the data compression techniques in WSNs, which can be classified into five categories:

1) The *string-based compression techniques* view sensing data as a sequence of characters and then adopt the data compression schemes used to handle text data to compress these sensing data. Inherited from these text data compression schemes, the string-based compression techniques can also provide lossless compression.
2) The *image-based compression techniques* organize a WSN into a hierarchical architecture and then adopt some image compression schemes such as wavelet transformation to provide multiple resolutions of sensing data inside the network. Some minor features of sensing data may be lost due to the compression operations

Y.-C. Wang is with the Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, Taiwan.
E-mail: wangyc@cs.nctu.edu.tw

and thus the image-based compression technique support loss compression.

3) The *distributed source coding techniques* compress sensing data inside the network according to the Slepian-Wolf theorem, which proves that two or more correlated data streams can be encoded independently and then be decoded jointly at a receiver with a rate equal to their joint entropy. Therefore, the distributed source coding techniques can support lossless compression.
4) The *compressed sensing techniques* indicate that any sufficiently compressible data can be accurately recovered from a small number of nonadaptive, randomized linear projection samples. Thus, they can exploit compressibility without relying on any prior knowledge or assumption on sensing data. With the above observation, the compressed sensing techniques can provide lossless compression.
5) The *data aggregation techniques* select a subset of sensor nodes to collect and fuse the sensing data sent from their neighboring nodes and then transmit the small-sized aggregated data to the sink. Because the original sensing data cannot be derived from these aggregated data, the compression of the data aggregation techniques is thus unrecoverable.

In this chapter, we give a comprehensive survey of recent research on each category of data compression techniques and then conclude the chapter by comparing these data compression techniques.

2 String-Based Compression Techniques

In text data, there have been many compression algorithms proposed to support lossless compression. The *Lempel-Ziv-Welch (LZW) algorithm* [14] is one popular example, which dynamically constructs a dictionary to encode new strings based on previously encountered strings. Fig. 1(a) gives the flowchart of the LZW algorithm, where a dictionary is initiated to include the single-character strings corresponding to all possible input characters. For example, by using the *American standard code for information interchange (ASCII)*, the dictionary will contain 256 initial entries. Then, the LZW algorithm scans each character of the input data stream until it can find a substring that is not in the dictionary. When such a string is found, the index of the longest matched substring in the dictionary is sent to the output data stream, while this new string is added into the dictionary with the next available code. Then, the LZW algorithm continues scanning the input data stream, starting from the last character of the previous string. Table 1 shows an example.

The LZW algorithm is computationally simple and has no transmission overhead. Specifically, because both the sender and the receiver have the same initial dictionary entries and all new dictionary entries can be derived from existing dictionary entries and the input data stream, the receiver can thus construct the complete dictionary on the fly when receiving the compressed data. With the above observation, the work in [15] develops an *S-LZW* algorithm (the abbreviation 'S' means "sensor") to support data compression in WSNs. S-LZW points out that in the LZW algorithm, the decoder must have received all previous entries in the block to decode a dictionary entry. However, since packet loss is usually common in a WSN,

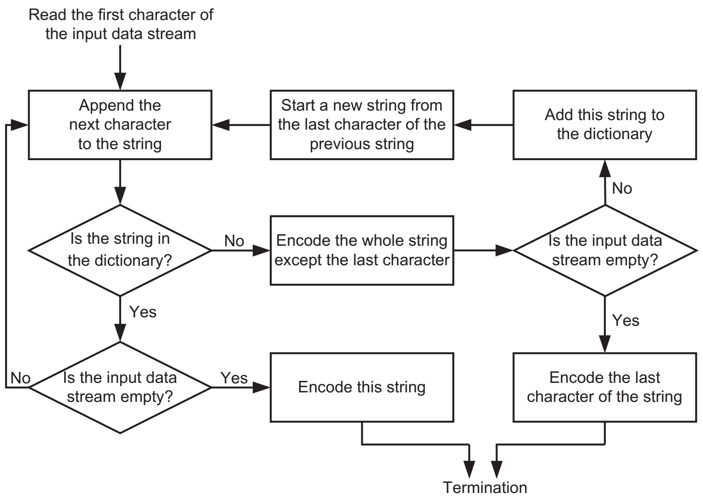

Read the first character of the input data stream
Append the next character to the string
Start a new string from the last character of the previous string
Add this string to the dictionary
Is the string in the dictionary?
No
Encode the whole string except the last character
Is the input data stream empty?
No
Yes
Yes
No
Is the input data stream empty?
Yes
Encode this string
Encode the last character of the string
Termination

Fig. 1: The flowchart of the LZW algorithm, where a dictionary is initiated to include the single-character strings corresponding to all possible input characters.

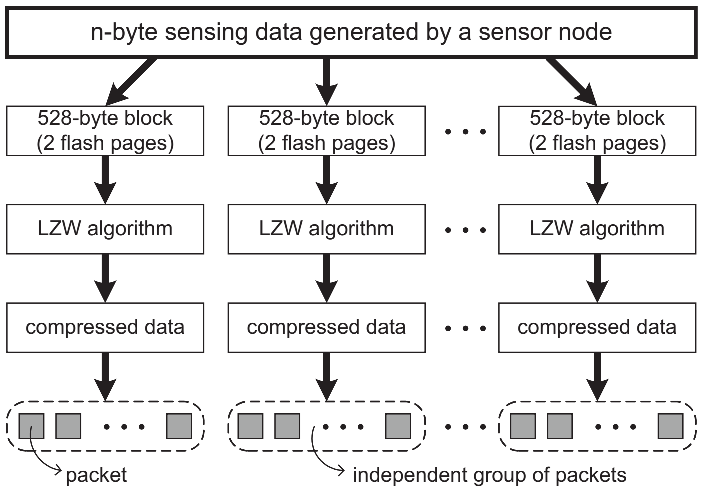

n-byte sensing data generated by a sensor node
528-byte block (2 flash pages)
LZW algorithm
compressed data
packet
independent group of packets

Fig. 2: The compression operation of S-LZW. The sensing data of each sensor node is first divided into blocks and each block of data will be compressed by the LZW algorithm separately. Each packet in a dashed block cannot be decompressed without proceeding the packets prior to it in the same group. However, the packets in different dashed blocks are independent with each other and thus can be decompressed separately.

S-LZW thus proposes dividing the data stream into small and independent blocks, as shown in Fig. 2. In this way, if a packet is lost due to collision or interference, S-LZW can guarantee that this packet only affects the following packets in its own block. According to the experiment results in [15], S-LZW suggests adopting a dictionary with size of 512 entries to fit the small memory of sensor nodes. Besides, S-LZW also suggests compressing sensing data into blocks of 528 bytes (that is, two flash pages) to achieve a better performance. In Fig. 2, the sensing data of each sensor node will be compressed and divided into multiple independent groups of packets. A packet in each group cannot be decompressed before proceeding the packets prior to it in the same group. On the other hand, any two groups of packets will not affect with each other and thus they can be independently decompressed. Therefore, the loss of a packet will affect at most one group of packets (that is, one block of data). To reduce such an effect, S-LZW suggests that each group contains at most 10 dependent packets.

Observing that sensing data may be repetitive over short intervals, the work of [15] also proposes a variation of S-

encoded substring	output data stream	new dictionary entry
x	120	256: xx
xx	120 256	257: xxx
x	120 256 120	258: xy
y	120 256 120 121	259: yx
xxx	120 256 120 121 257	260: xxxy
y	120 256 120 121 257 121	261: yz
z	120 256 120 121 257 121 122	262: zz
z	120 256 120 121 257 121 122 122	none

TABLE 1: An example of executing the LZW algorithm, where the input data stream is "xxxxyxxxyzz" and the ASCII coding scheme is adopted to initiate the dictionary.

LZW, called *S-LZW-MC* (the abbreviation "MC" represents "mini-cache"), by adding a mini-cache into the dictionary. In particular, the mini-cache is a hash-indexed dictionary of size 2^k, $k \in \mathbb{N}$, that maintains recently created and used dictionary entries. In S-LZW-MC, the last four bits of each dictionary entry are used as the hash index. Fig. 3 illustrates the flowchart of S-LZW-MC, which follows the LZW algorithm but adds some modifications (marked by dashed diagrams). When a dictionary entry is identified, S-LZW-MC will search the mini-cache for the entry. If this dictionary entry is in the mini-cache, S-LZW-MC encodes it with the mini-cache entry number and appends a '1' bit to the end. In this way, the entry has a length of $(k+1)$ bits. Otherwise, S-LZW-MC encodes the entry with the standard dictionary entry number and appends a '0' bit to the end, and then adds this entry into the mini-cache. Therefore, a high mini-cache hit rate can allow S-LZW-MC to make up for the one-bit penalty on searching the entries not in the mini-cache. Table 2 gives an example of executing the S-LZW-MC algorithm. By exploiting the mini-cache, S-LZW-MC could further reduce the computation time and improve the compression ratio, as compared with the S-LZW algorithm.

3 Image-Based Compression Techniques

An image is usually composed of many small *pixels* and this image can be modeled by a matrix whose elements are the values of these pixels. By conducting some wavelet transformation on the matrix, we can extract the important features from the image in the frequency domain [16]. Then, the image size can be significantly reduced by storing only these important features of the image.

The image-based compression techniques adopt the similar idea. They organize a WSN into a hierarchial architecture and consider the sensing data sent from all sensor nodes as an image containing multiple pixels. Then, a wavelet transformation is performed to extract the spatial and temporal summarization from these sensing data. Explicitly, when sensing data possess a higher degree of spatial or temporal correlation, the image-based compression techniques can further reduce the amount of sensing data. In this section, we introduce two representative frameworks of the image-based compression techniques: *DIMENSIONS framework* [17] and *multi-resolution compression and query (MRCQ) framework* [18].

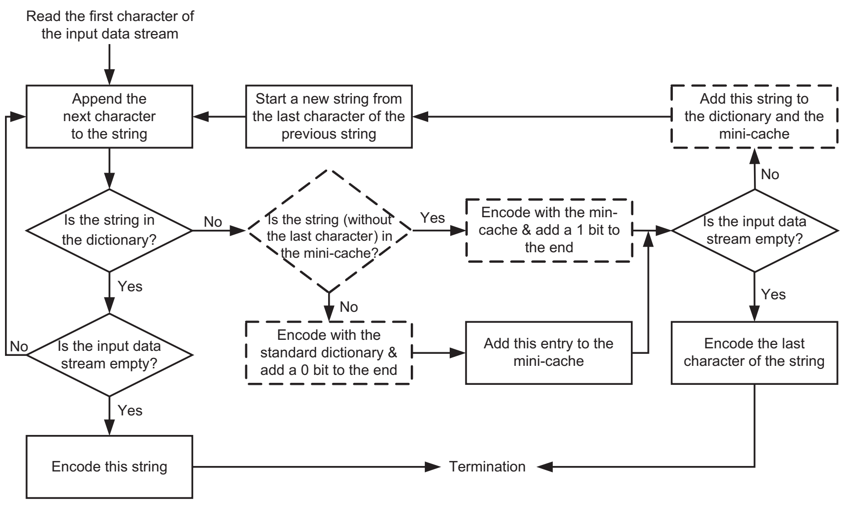
3.1 DIMENSIONS Framework

The DIMENSIONS framework exploits the wavelet transformation and quantization to reduce the amount of data transmissions of sensor nodes and support different resolutions of sensing data for users to query. The system architecture of DIMENSIONS is shown in Fig. 4(a), where the network is organized into multiple *levels*. A block in each level k contains four blocks in a lower level $(k-1)$. In each block of level k, one cluster head is selected to collect the sensing data sent from the corresponding four blocks in level $(k-1)$. Then, the cluster head will conduct the compression operation on these sensing data to extract their spatial summarization. Since each level k will pass only the spatial summarization of the sensing data to the higher level $(k+1)$, the data stored in each level will exhibit different resolutions. In particular, the data stored in a lower level possess a finer resolution while the data stored in a higher level possess a coarser resolution. In this way, the amount of sensing data transmitted by sensor nodes can be reduced while users can query more detailed information from the cluster heads in a lower level.

In DIMENSIONS, each sensor node will try to reduce the amount of its own sensing data by taking advantage of temporal correlation in the signal and a priori knowledge related to signal characteristics. In particular, each sensor node can adopt some real-time filtering schemes such as a simple amplitude threshold to extract the temporal summarization from its sensing data in the time domain. Only when the sensing reading exceeds a predefined *signal-to-noise ratio (SNR)* threshold will the sensor node transmit the sensing data to its cluster head.

On the other hand, for each cluster head of level k, it will collect the compressed data sent from the corresponding four cluster heads in level $(k-1)$, dequantize these data, and then compress the data again and send them to the cluster head in level $(k+1)$. Fig. 4(b) shows the compression operation conducted in each cluster head of level k. In particular, after obtaining the compressed data sent from the lower-level cluster heads, DIMENSIONS will adopt a Huffman decoder and a dequantization module to handle these data. Then, these data will be kept in a local storage and passed to a *three-dimensional discrete wavelet transform (3D-DWT)* module [19] to generate the spatiotemporal summarization of sensing data. Then, by using a quantization module and a Huffman encoder, this summarization can be further compressed. The compressed data will then be sent to the cluster head in level $(k+1)$. The above operations will be repeated until the data can be transmitted to the sink. Since the data passed through each level will be handled by 3D-DWT, the amount of data sent to higher levels may be reduced but their resolutions would be also degraded.

Explicitly, DWT is the core of the compression scheme in DIMENSIONS. The concept of DWT is shown in Fig. 4(c), where the original signal $x[n]$, $n \in \mathbb{N}$, will be handled by a sequence of low-pass and high-pass filters. Such a sequence of filtering is usually called a *Mallat-tree decompo-*



Read the first character of the input data stream
Append the next character to the string
Start a new string from the last character of the previous string
Add this string to the dictionary and the mini-cache
No
Is the string in the dictionary?
No
Is the string (without the last character) in the mini-cache?
Yes
Encode with the min-cache & add a 1 bit to the end
Is the input data stream empty?
Yes
No
Is the input data stream empty?
No
Encode with the standard dictionary & add a 0 bit to the end
Add this entry to the mini-cache
Yes
Encode the last character of the string
Yes
Encode this string
Termination

Fig. 3: The flowchart of the S-LZW-MC algorithm, where the dashed diagrams indicate the different designs with the LZW algorithm.

encoded substring	new output	new dictionary entry	mini-cache changes	data length (bits) LZW	mini-cache
x	120, 0	256: xx	0: 256, 1: 120	9	10
xx	0, 1	257: xxx	1: 257	18	15
x	120, 0	258: xy	1: 120, 2: 258	27	25
y	121, 0	259: yx	2: 121, 3: 259	36	35
xxx	257, 0	260: xxxy	1: 257, 4: 260	45	45
y	2, 1	261: yz	5: 261	54	50
z	122, 0	262: zz	3: 122, 6: 262	63	60
z	3, 1	none	none	72	65

TABLE 2: An example of executing the S-LZW-MC algorithm, where the same input data stream "xxxxyxxxyzz" in Table 1 is adopted.

sition. Fig. 4(c) shows a three-stage wavelet decomposition tree. The high-pass filter is denoted by H_p while the low-pass filter is denoted by L_p. At each stage i, the high-pass filter will generate detailed coefficients $d_i[n]$ while the low-pass filter associated with a scaling function will generate approximative coefficients $a_i[n]$. Each half-band filter (marked by "$\downarrow 2$") will generate signals spanning only half of the frequency band. According to the Nyquist sampling theorem [20], if the original signal has the highest frequency of ω radians, this signal requires a sampling frequency of 2ω radians. By lowering down the highest frequency to $\frac{\omega}{2}$ radians, this signal can be sampled at a frequency of ω radians. Therefore, the half of the samples can be discarded to reduce the data size. By the DWT operation, the time resolution becomes arbitrarily good at high frequencies while the frequency resolution becomes arbitrarily good at low frequencies. Then, the DWT result of the original signal $x[n]$ can be obtained by concatenating all of the coefficients (that is, $a_i[n]$ and $d_i[n]$), starting from the last stage of decomposition.

The DIMENSIONS framework can help reduce the amount of sensing data that sensor nodes have to regularly report to the sink. However, because each cluster head needs to execute the Huffman coding scheme, 3D-DWT, and the quantization operation, DIMENSIONS may incur a higher computation cost.

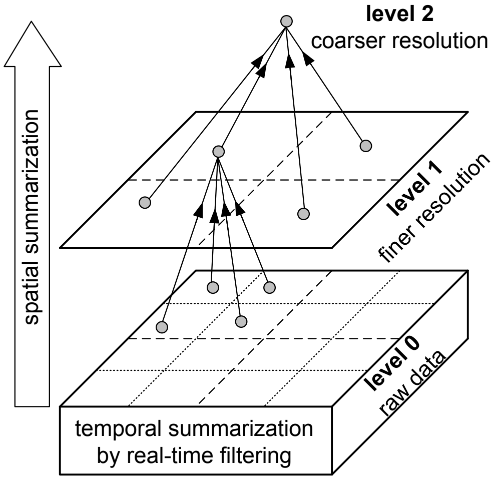
3.2 MRCQ Framework

In the MRCQ framework, sensor nodes are organized hierarchically and the objective is to establish multi-resolution summaries of sensing data inside the network through spatial and temporal compressions. In particular, the sink receives only the lower-resolution summaries while other higher-resolution summaries are kept in the network, which can be obtained through queries. To satisfy the above requirements, a hierarchical WSN architecture is proposed, as shown in Fig. 5(a). Specifically, the WSN is recursively partitioned into K blocks and is organized into d *layers*, where $K > 1$ and $d > 1$. Each block in layer $(i+1)$ contains K blocks in layer i. In each layer, one sensor node is selected in each block as the *processing node (PN)* to collect and compress the sensing data sent from the lower-layer blocks. In the lowest layer 1, the PN will compress the sensing data sent from the *leaf sensor nodes (LNs)*. The area handled by each layer-1 PN is partitioned into $k \times k$ pixels, where $k > 1$ is a small integer. Each pixel ideally contains one LN and the value of the pixel is the sensing data of that LN. However, when a pixel contains more than one LN, its value is the average of the sensing data of these LNs. On the other hand, when a pixel contains no sensor node, its value can be estimated by interpolating the values of its neighboring pixels.

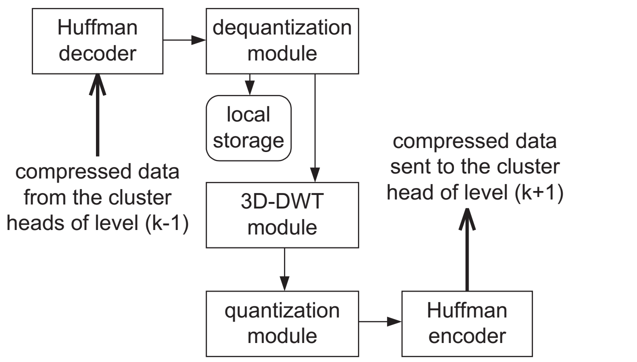
Sensing data are transmitted from LNs to the sink layer by layer. Data passing through each layer are compressed by its PNs using a spatial compression algorithm, which contains three components:

- Layer-1 compression: Each layer-1 PN collects the sensing data from its LNs and model these data by a matrix $M = (s_{i,j})_{k \times k}$, where $s_{i,j}$ is the value of a pixel (i,j). Then, the PN applies a *two-dimensional discrete cosine transform (2D-DCT)* scheme [21] on M to construct a new matrix $\widehat{M} = (t_{i,j})_{k \times k}$, where

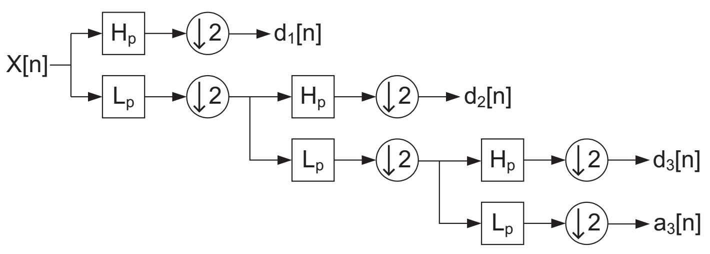
$$t_{i,j} = \frac{2C(i)C(j)}{k} \cdot \sum_{x=0}^{k-1}\sum_{y=0}^{k-1} s_{x,y} \cdot \cos\left(\frac{i\pi(2x+1)}{2k}\right) \cdot \cos\left(\frac{j\pi(2y+1)}{2k}\right), \quad (1)$$



(a)

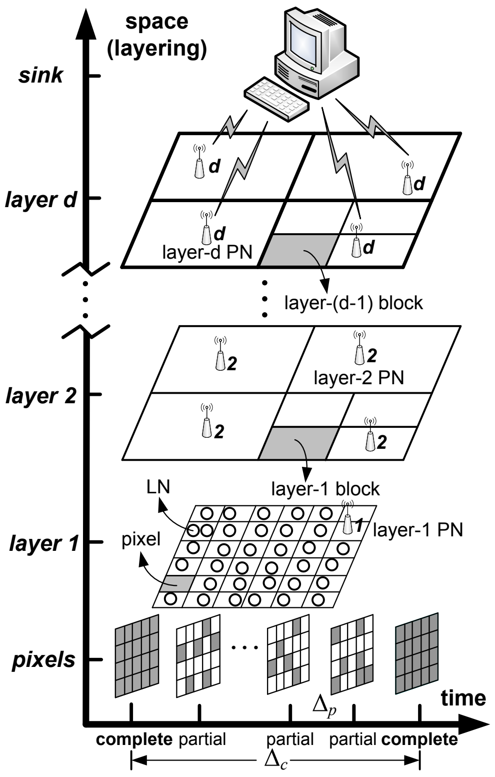

Huffman decoder
dequantization module
local storage
compressed data from the cluster heads of level (k-1)
3D-DWT module
compressed data sent to the cluster head of level (k+1)
quantization module
Huffman encoder

(b)

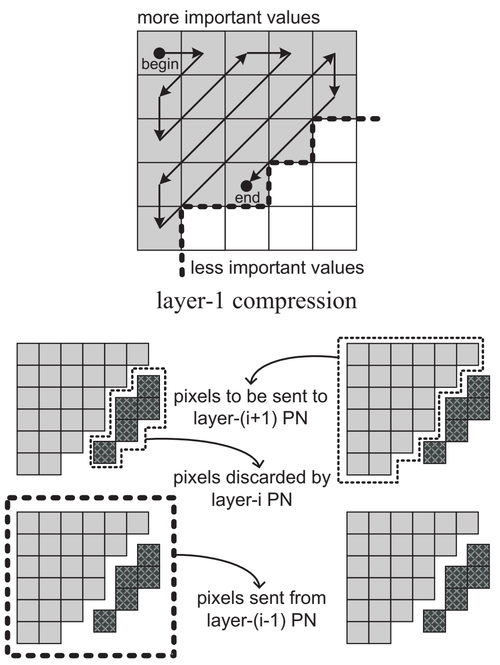


(c)

Fig. 4: The DIMENSIONS framework: (a) the hierarchical architecture to extract the spatial and temporal summarization from sensing data, (b) the compression operation conducted in each cluster head of level k, where the data will be handled by the Huffman coding, the quantization, and the 3D-DWT schemes, and (c) a three-stage wavelet decomposition tree to extract the detailed coefficients $d_1[n]$, $d_2[n]$, and $d_3[n]$ and the approximative coefficients $a_3[n]$ from the original data $x[n]$.



(a)


more important values
begin
end
less important values
layer-1 compression
pixels to be sent to layer-(i+1) PN
pixels discarded by layer-i PN
pixels sent from layer-(i-1) PN
layer-i compression

(b)

Fig. 5: The MRCQ framework: (a) the hierarchical WSN architecture to support multiple resolutions of sensing data, where each LN and layer-1 PN will compress its own data by the temporal compression algorithm, while each layer-i PN will compress the data sent from its K corresponding layer-$(i-1)$ PNs by the spatial compression algorithm, and (b) the spatial compression algorithm conducted in each layer-1 and layer-i PNs, where each layer-1 PN will conduct 2D-DCT on its $k \times k$ matrix and transmit only $\lceil r \cdot k^2 \rceil$ pixels to its layer-2 PN, while each layer-i PN will transmit only $\lceil r^i \cdot k^2 \rceil \cdot K^{i-1}$ pixels to its layer-$(i+1)$ PN.

where $C(i) = \frac{1}{\sqrt{2}}$ if $i = 0$ and $C(i) = 0$ otherwise. The 2D-DCT scheme can extract the features from a matrix M, where those significant features will appear in the upper left part of the transformed matrix $\widehat{M}$, while those insignificant features will appear in the opposite part. Therefore, we can preserve the most important features of M and truncate the lower right part of $\widehat{M}$ to achieve data compression. In particular, we can retrieve elements of $\widehat{M}$ from the upper left corner toward the lower right corner along the diagonal direction, as shown in Fig. 5(b), until $\lceil r \cdot k^2 \rceil$ elements are scanned, where $0 < r \leq 1$ is the compression ratio. Then, these scanned elements of $\widehat{M}$ will be sent to the layer-2 PN.

- Layer-i compression: A layer-i PN, $i \geq 2$, will collect the reduced matrices $\widehat{M}$ from its corresponding K layer-$(i-1)$ PNs, where each matrix $\widehat{M}$ contains only $\lceil r^{i-1} \cdot k^2 \rceil$ pixels. For each reduced matrix $\widehat{M}$, the layer-i PN transmits the first most significant $\lceil r^i \cdot k^2 \rceil$ pixels to the layer-$(i+1)$ PN and discards the remaining $\lfloor r^{i-1} \cdot k^2 - r^i \cdot k^2 \rfloor$ pixels. In this way, the layer-i

compression can incur only a small computation cost. Fig. 5(b) gives an example.

- Sink decompression: After collecting K^{d-1} reduced matrices from the highest layer-d, the sink first expands each reduced matrix to a $k \times k$ matrix $\widehat{M'} = (t_{i,j})_{k\times k}$ by appending sufficient zeros at the end. Then, the sink adopts the *inverse 2D-DCT* scheme to transform $\widehat{M'}$ to a matrix $M' = (s'_{i,j})_{k\times k}$, where

$$s'_{i,j} = \frac{2}{k}\sum_{x=0}^{k-1}\sum_{y=0}^{k-1} C(x)C(y)\cdot t_{x,y}\cdot \cos\left(\frac{x\pi(2i+1)}{2k}\right)\cdot \cos\left(\frac{y\pi(2j+1)}{2k}\right).$$

Note that since the sink appends zeros in the matrix $\widehat{M'}$, the recovered matrix M' will be an approximation of the original matrix M. Then, the sink can combine all of these K^{d-1} recovered matrices to obtain a coarser resolution of sensing data from the environment.

On the other hand, LNs and layer-1 PNs will compress their data by a temporal compression algorithm. In particular, the time axis is divided into *complete reporting intervals* of the same length Δ_c, as shown in Fig. 5(a). Each complete reporting interval is further divided into smaller *partial reporting intervals* of length Δ_p, where Δ_c is a multiple of Δ_p. In the beginning of each complete reporting interval, LNs and layer-1 PNs will report and compress data as we discussed earlier. During each complete reporting interval, differential compression will be conducted in the beginning of each partial reporting interval. Specifically, given an updating threshold δ_L, an LN will decide not to report if its current sensing data v_{new} differs from its previously reported data v_{old} by an amount no more than δ_L, that is, $|v_{\text{new}} - v_{\text{old}}| \leq \delta_L$. In this case, its layer-1 PN will use v_{old} as the current sensing data of that LN. Similarly, given another updating threshold δ_P, a layer-1 PN will decide not to report if the difference between its current matrix $M_{\text{new}} = (t_{i,j})_{k\times k}$ and its previously reported matrix $M_{\text{old}} = (s_{i,j})_{k\times k}$ satisfies the following inequality

$$\frac{1}{k^2}\sum_{i=1}^{k}\sum_{j=1}^{k}|s_{i,j} - t_{i,j}| \leq \delta_P.$$

In this case, its layer-2 PN will use M_{old} as the current sensing matrix of that layer-1 PN.

Compared to DIMENSIONS, MRCQ incurs less computation cost since the complicated 2D-DCT operation is conducted only at layer 1. Because k is a small constant, a small table can be maintained in each layer-1 PN to record the results of cosine operations for each (i, x) and (j, y) pair in Eq. (1) to reduce its computation cost. In addition, the proposed temporal compression algorithm is based on a simple differential idea. Therefore, MRCQ could be implemented on simple sensor platforms.

4 Distributed Source Coding Techniques

One foundation of the distributed source coding techniques is the *Slepian-Wolf theorem* [22]. Given two or more correlated data streams, each being encoded independently, and then decoded jointly at one receiver, the Slepian-Wolf theorem shows that it is feasible to achieve lossless encoding of these two data streams at a *rate* (that is, the number of bits used to encode each character) equal to their joint entropy. Fig. 6(a) gives an example, where there are two correlated data streams X and Y generated by making n independent drawings from a joint probability distribution $P(X = x, Y = y)$. Encoder 1 receives data stream X and then transmits a coded message to the decoder, where each character of X is encoded by a number of R_X bits. Similarly, encoder 2 receives data stream Y and then transmits a coded message to the decoder, where each character of Y is encoded by a number of R_Y bits. On receiving these two coded messages, the decoder will generate two n-vectors X^* and Y^*, which are the estimations of the original data streams X and Y, respectively.

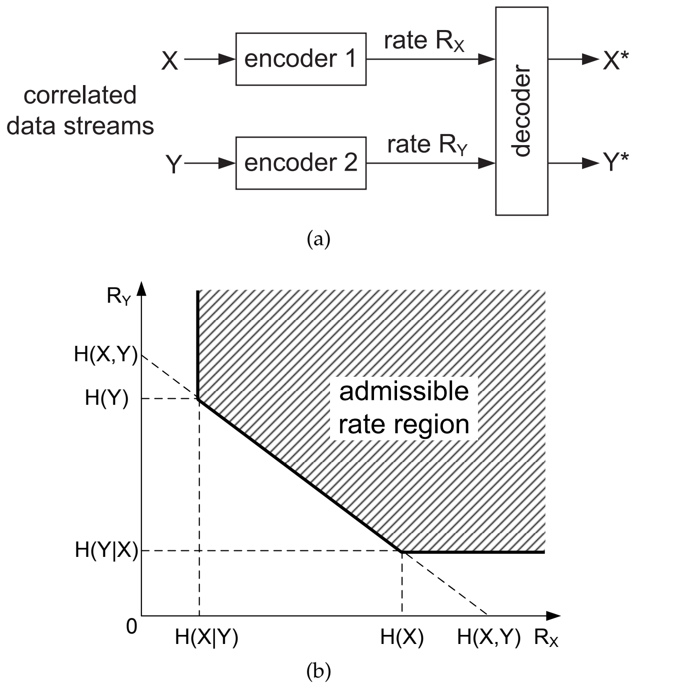


Fig. 6: The Slepian-Wolf theorem: (a) independent encoding and joint decoding of two correlated data streams X and Y, where encoder 1 will encode each character of X by a number of R_X bits while encoder 2 will encode each character of Y by a number of R_Y bits, and (b) the admissible rate region for the rate pair (R_X, R_Y), which should satisfy the inequalities of $R_X \geq H(X|Y)$, $R_Y \geq H(Y|X)$, and $R_X + R_Y \geq H(X, Y)$.

When n is sufficiently large, the probability that $X^* \neq X$ or $Y^* \neq Y$ can approximate to zero. That is, we can achieve lossless data compression of X and Y. In this case, the system is called an *admissible system*. The pair of rates (R_X, R_Y) for an admissible system is called an *admissible rate pair*. The closure of the set of all possible admissible rate pairs is called the *admissible rate region*. The admissible rate region can be calculated by measuring the entropies of random variables X and Y with joint probability distribution $P(X = x, Y = y)$:

$$H(X,Y) = -\sum_x\sum_y P(X=x, Y=y)\cdot \lg P(X=x, Y=y),$$

$$H(X) = -\sum_x P(X=x)\cdot \lg P(X=x),$$

$$H(Y) = -\sum_y P(Y=y)\cdot \lg P(Y=y),$$

$$H(Y|X) = -\sum_x P(X=x)\sum_y P(Y=y|X=x)\cdot \lg P(Y=y|X=x),$$

$$H(X|Y) = -\sum_y P(Y=y)\sum_x P(X=x|Y=y)\cdot \lg P(X=x|Y=y).$$

By the Slepian-Wolf theorem, the admissible rate region for the pair of rates (R_X, R_Y) is the set of points that satisfy the following three inequalities:

$$\begin{aligned} R_X &\geq H(X|Y),\\ R_Y &\geq H(Y|X),\\ R_X + R_Y &\geq H(X,Y). \end{aligned}$$

Fig. 6(b) shows the admissible rate region. The advantage of the Slepian-Wolf theorem can be observed by comparing it with the entropy bound for compression of single sources. In particular, separate encoders that ignore the source correlation can achieve rates of only $R_X + R_Y \geq H(X) + H(Y)$. However, by adopting the Slepian-Wolf coding, the separate encoders can exploit their knowledge of the correlation to achieve the same rates as an optimal joint encoder, that is, $R_X + R_Y \geq H(X,Y)$.

The Slepian-Wolf theorem provides a theoretical tool to characterize the amount of communications required for the distributed source coding in a network where correlated data streams are physically separated or each encoder has limited computation capability. The studies of [23], [24] give an example of applying the Slepian-Wolf theorem to compress sensing data in WSNs. Specifically, suppose that X and Y are the sensing readings of two sensor nodes, where X and Y are equiprobable binary triplets with $X, Y \in \{0,1\}^3$ and the Hamming distance between X and Y is no more than one. In this case, we have $H(X) = H(Y) = 3$ bits. Since X and Y differ at most in one position, for any given Y, there are four equiprobable choices of X. For example, suppose that Y is 111, then X belongs to the set $\{111, 011, 101, 110\}$. Thus, we can obtain that $H(X|Y) = 2$ bits. In other words, to jointly encode X and Y, it takes three bits to represent Y and two additional bits to index these four possible choices of X associated with Y. Therefore, at least $H(X,Y) = H(Y) + H(X|Y) = 5$ bits are required. In fact, the information Y is perfectly known at the decoder (for example, the sink) but not at the encoder (that is, the sensor that generates X). However, according to the Slepian-Wolf theorem, it is still possible to send only $H(X|Y) = 2$ bits rather than $H(X) = 3$ bits to decode X without any loss at the joint decoder. One solution is to first divide the set of all possible outcomes of X into four subsets $X_{00} = \{000, 111\}$, $X_{01} = \{001, 110\}$, $X_{10} = \{010, 101\}$, and $X_{11} = \{011, 100\}$ and then send two bits for the index i of the subset X_i that X belongs to. When generating the subsets X_i's, we should guarantee that each of these subsets has two elements with a Hamming distance of 3. Then, to jointly decode with i (and thus X_i) and information Y, we choose the X with $d_H(X,Y) \leq 1$ in subset X_i, where $d_H(X,Y)$ is the Hamming distance between X and Y. In this case, we can make sure of unique decoding because the two elements in each subset X_i have a Hamming distance of 3. Therefore, we can achieve the Slepian-Wolf limit of $H(X,Y) = H(Y) + H(X|Y) = 3 + 2 = 5$ bits in the above example with lossless decoding.

5 Compressed Sensing Techniques

The distributed source coding techniques allow sensor nodes to compress their sensing data without collaboration and negotiation but require prior knowledge of the precise correlation in the data. However, in many WSN applications, such prior knowledge is usually unavailable. Therefore, the compressed sensing (sometimes called *compressive sensing*) techniques are proposed by exploiting compressibility without relying on any specific prior knowledge or assumption on data [25]. The compressed sensing theory points out that any sufficiently compressible data can be accurately recovered from a small number of nonadaptive, randomized linear projection samples. In particular, given m-sparse data $\mathbf{x} = (x_{i,j})_{n\times 1}$ (that is, $\mathbf{x}$ has no more than m nonzero entries) where m is much smaller than the data length n, we can calculate a random projection matrix $\mathbf{A} = (A_{i,j})_{k\times n}$ with far fewer rows than columns (that is, $k \ll n$) to obtain a small compressed data set $\mathbf{y} = (y_{i,j})_{k\times 1} = \mathbf{A}\mathbf{x} + \epsilon$, where ϵ is the error caused by noise or other perturbations.

The concept of the above random projection is illustrated in Fig. 7, where a network consisting of $n = 16$ sensor nodes is considered [26]. Suppose that only one sensor node has a positive sensing reading while the remaining 15 sensor nodes have zero sensing readings. The objective is to use the minimum number of *observations* to identify which sensor node has the nonzero sensing reading. In other words, we have 1-sparse data $\mathbf{x} = (x_{i,j})_{16\times 1}$ and want to find out which entry is nonzero. By adopting the compressed sensing technique, we can compress data $\mathbf{x}$ by a random projection matrix $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4)^T$, where each row $A_l = (A_{i,j})_{1\times 16}$, $l = 1..4$, is called a *random vector*. In Fig. 7, the nodes in each A_l colored by grey will multiply their data values by '-1' while the nodes in each A_l colored by white will multiply their data values by '$+1$'. Each node in a random vector is colored by grey with a probability of 0.5. Then, we can calculate the compressed data $\mathbf{y} = (y_1, y_2, y_3, y_4)^T = \mathbf{A}\mathbf{x}$ (for ease of presentation, here we ignore the noise ϵ) and obtain the hypotheses of data H_1, H_2, H_3, and H_4 according to each y_i value. For example, if $y_i < 0$, the hypothesis data will be the same as those in the random vector. Otherwise, the hypothesis data will be the inverse of those in the random vector. By comparing these hypothesis data, we can find out which sensor node has the nonzero sensing data, as shown in Fig. 7. Here, about $\frac{n}{2}$ hypothesis of sensing data are consistent with each random projection observation. However, the number of hypotheses which are simultaneously consistent with *all* observations decreases exponentially with the number of observations. Therefore, only $\lg n = 4$ observations are required to determine which sensor node possesses the nonzero sensing reading.

The work of [26] adopts the compressed sensing technique to support lossless data compression in a WSN. Suppose that there are n sensor nodes in the network. Each sensor node will use its network address as the seed to feed into a pseudo-random number generator to locally draw a vector $\{A_{i,j}\}_{i=1}^{k}$, where $k \ll n$. In this way, the sink can also calculate the corresponding vector for each sensor node by feeding its network address as the seed to the same pseudo-random number generator. Then, for each sensor node at location j, $j = 1..n$, it will multiply its sensing reading x_j

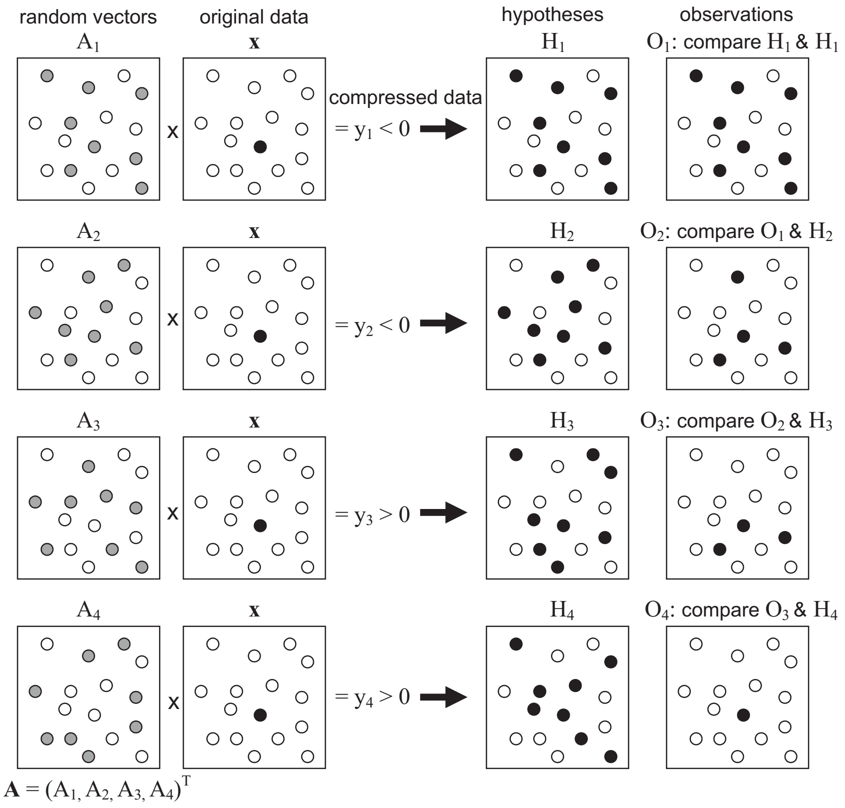


Fig. 7: An example of the random projection in the compressed sensing techniques, where a network with 16 sensor nodes is considered and only one sensor node (colored by black) possesses the positive sensing data.

by vector $\{A_{i,j}\}_{i=1}^{k}$ to calculate a k-tuple projection

$$P_j = (A_{1,j} \cdot x_j, A_{2,j} \cdot x_j, \cdots, A_{k,j} \cdot x_j)^T.$$

All sensor nodes will coherently transmit their respective projections P_j's in an analog fashion over the network-to-sink air interface using k transmissions. Due to the additive nature of radio waves, the corresponding received signal at the sink in the end of the kth transmission can be given by

$$\mathbf{y} = \sum_{j=1}^{n} P_j + \epsilon = \mathbf{A}\mathbf{x} + \epsilon,$$

where ϵ is the noise that may be caused by the receiving circuitry of the sink. Then, the sink can decode the received projection $\mathbf{y}$ to calculate the original sensing data $\mathbf{x}$.

6 Data Aggregation Techniques

Unlike other data compression techniques that exploit different compression theorems to provide either lossless or loss compression of sensing data, the data aggregation techniques consider reducing the amount of data transmissions in a WSN by *fusing* (or *aggregating*) these sensing data. In particular, the data aggregation techniques usually select a subset of sensor nodes (called *aggregation nodes*) to collect the sensing data sent from their neighboring sensor nodes and then adopt some aggregating schemes to fuse these sensing data such as taking their maximum, minimum, or average values. In this case, the amount of sensing data transmitted to the sink can be significantly reduced but such aggregating schemes are explicitly unrecoverable. Since these aggregating schemes are usually simple, the research on the data aggregation techniques usually aims at how to efficiently select these aggregation nodes to help reduce the overall data transmissions in a WSN [27]. In this section, we introduce the data aggregation techniques in WSNs. According to their network structures, these data aggregation schemes can be classified into four categories: 1) the *tree-based data aggregation schemes* organize the network into a tree structure for data collection and aggregation purpose, 2) the *cluster-based data aggregation schemes* group sensor nodes into clusters and then each cluster head will aggregate the sensing data within its cluster, 3) the *chain-based data aggregation schemes* make each sensor node transmit the sensing data to its nearest neighbor and thus the network will form a chain structure to aggregate the sensing data in the network, and 4) the *sector-based data aggregation schemes* adopt a ring-sector division concept to cluster sensor nodes such that the sensor nodes in the same sector will be assembled into one cluster.

6.1 Tree-Based Data Aggregation Schemes

The objective of the tree-based data aggregation schemes is to maximize the network lifetime by jointly optimizing data aggregation and routing tree formation [28]. These schemes organize the sensor nodes into a tree structure, where the data aggregation operation is conducted at the intermediate nodes along the tree and the aggregated data of the whole network will be eventually sent to the root node (that is, the sink). Fig. 8 gives an example of the tree-based data aggregation schemes, where each sensor node i will generate its sensing data s_i and each aggregation node j will fuse the data sent from the child nodes with its own sensing data s_j by an aggregation function $f(\cdot, \cdot, \cdot)$.

The work of [29] proposes an *energy-aware distributed heuristic (EADAT)* to maintain a data aggregation tree in a WSN. To construct such a tree, the sink (that is, the tree

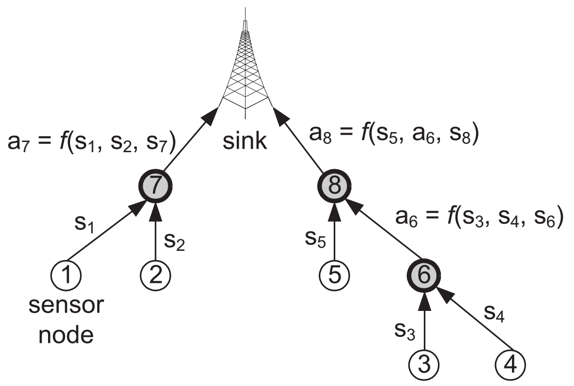


Fig. 8: The concept of the tree-based data aggregation schemes, where s_i is the original (uncompressed) sensing data, a_i is the aggregated data, and $f(\cdot,\cdot,\cdot)$ is the aggregation function. Each sensor node i will generate sensing data s_i. The sensor nodes colored by grey are aggregation nodes. Only intermediate nodes along the tree can become the aggregation nodes.

root) first broadcasts a control message *(ID, parent, power, status, hopcnt)* indicating the identification of the sensor node, the parent node along the tree, the residual power of the sensor node, the status of the sensor node (including *leaf*, *non-leaf*, and *undefined* states), and the hop counts from the sink. On receiving this control message for the first time, each sensor node sets up a timer that counts down every time when the channel becomes idle. Then, the sensor node selects the neighboring node with more residual power and a shorter path to the sink as its parent node along the tree. When the timer expires, the sensor node updates the control message and broadcasts the message to its neighboring nodes. The above process is repeated until all sensor nodes are added into the tree. After constructing the data aggregation tree, a residual power threshold P_{th} is used to maintain the tree. In particular, when the residual power of a sensor node becomes less than the threshold P_{th}, this sensor node will broadcast a *help* message to its child nodes. Then, one of its child nodes will replace this sensor node to maintain the tree structure.

The work of [30] develops a *power-efficient data gathering and aggregation protocol (PEDAP)* to maximize the network lifetime in terms of the number of *rounds*, where each round corresponds to the aggregation of sensing data transmitted from different sensor nodes. To achieve this objective, PEDAP tries to minimize the total energy consumption of sensor nodes in each round by calculating a minimum spanning tree over the network with link costs given by

$$c_{i,j}(k) = e_{circuit} \cdot 2k + e_{amp} \cdot k \cdot (d(i,j))^2,$$

where $c_{i,j}(k)$ is the energy cost to transmit k bits from node i to node j, $e_{circuit}$ is the amount of energy consumed by the transmitter/receiver circuitry per bit, e_{amp} is the amount of energy consumed by the transmitting amplifier per bit, and $d(i,j)$ is the distance between nodes i and j. The Prim's algorithm [31] is adopted to calculate the minimum spanning tree and then the data packets are transmitted to the sink through the tree edges of the minimum spanning tree. An energy-aware version of PEDAP is also proposed by considering the residual energy of sensor nodes. In particular, this energy-aware version modifies the link costs by $\frac{c_{i,j}(k)}{e_i}$, where e_i is the normalized residual energy of sensor node i and such a normalization is with respect to the initial energy of that sensor node. In this way, a sensor node remaining less energy will incur a larger link cost and thus the corresponding link may not be included in the minimum spanning tree. Therefore, the load among sensor nodes could be balanced.

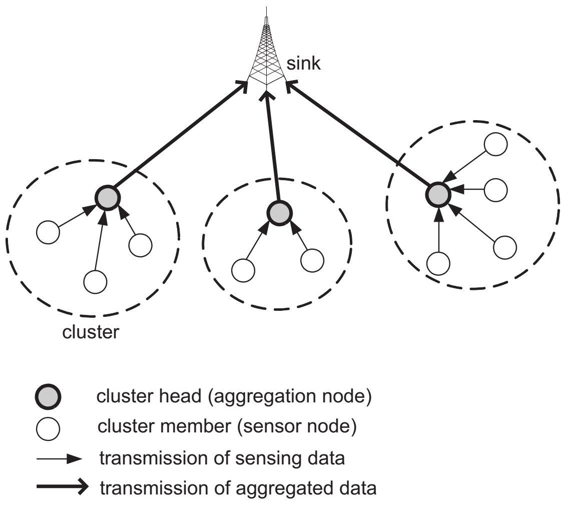

sink
cluster
cluster head (aggregation node)
cluster member (sensor node)
transmission of sensing data
transmission of aggregated data

Fig. 9: The concept of the cluster-based data aggregation schemes. Each cluster head is responsible for collecting and aggregating the sensing data sent from other sensor nodes in the cluster. Then, these cluster heads can transmit the aggregated data to the sink directly using a large-range transmission power or indirectly using a multi-hop routing protocol.

6.2 Cluster-Based Data Aggregation Schemes

The cluster-based data aggregation schemes first group sensor nodes into *clusters* and then select one *cluster head* in each cluster to aggregate the sensing data sent from other sensor nodes in that cluster. Then, these cluster heads can transmit the aggregated data to the sink *directly* through long-range transmissions or *indirectly* via multi-hop communications through other cluster heads. Fig. 9 illustrates an example of the cluster-based data aggregation schemes.

The work of [32] proposes a distributed cluster-based data aggregation scheme, called *low-energy adaptive clustering hierarchy (LEACH)*, which consists of a *setup phase* to organize the network into clusters and select the corresponding cluster heads and a *steady-state phase* to conduct data aggregation at these cluster heads. In the setup phase, a fraction α of sensor nodes will elect themselves as the cluster heads, where $0 < \alpha < 1$. In particular, each sensor node first generates a random number r_i between 0 and 1. If number r_i exceeds a threshold δ, this sensor node can become a cluster head, where the threshold δ is calculated by

$$\delta = \frac{\alpha}{1 - \alpha \cdot (r_i \bmod (\frac{1}{\alpha}))},$$

where the *mod* operation will return the remainder after division. Then, each elected cluster head will broadcast a message to announce that they are cluster heads. All other sensor nodes that are not cluster heads will join the clusters according to the received signal strengths from these received messages. Then, in the steady-state phase, each cluster member will send its sensing data to the cluster head. By aggregating these sensing data, each cluster head will transmit the aggregated data to the sink. Note that in LEACH, sensor nodes are assumed to have sufficiently large transmission powers so that the cluster heads can directly transmit their data to the sink.

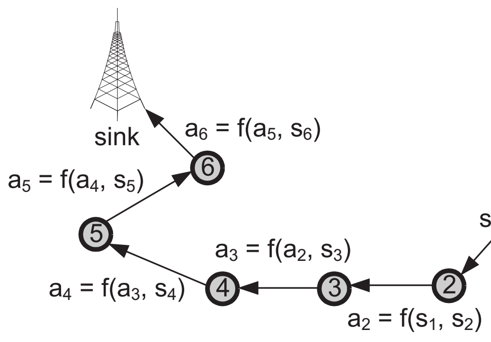


Fig. 10: The concept of the chain-based data aggregation schemes. Each sensor node i will generate its sensing data s_i. Except for the node(s) in the end of the chain (for example, node 1), all other nodes will serve as the aggregation nodes to fuse the data sent from their upstream nodes by the aggregation function $f(\cdot, \cdot)$.

The work of [33] proposes a *hybrid energy-efficient distributed clustering (HEED) scheme*, whose objective is to construct efficient clusters so that the network lifetime can be maximized. HEED assumes that sensor nodes can adjust their transmission powers and the selection of cluster heads is according to the combination of residual energy of sensor nodes and their link degrees. Specifically, HEED is executed iteration by iteration. In each iteration, a sensor node that does not join any cluster will calculate a probability

$$p_i = \alpha \cdot \frac{e_{residual}}{e_{max}},$$

to elect itself as a tentative cluster head, where α is the initial fraction of sensor nodes to serve as the cluster heads (which can be specified by applications), $e_{residual}$ is the current residual energy of the sensor node, and e_{max} is the maximum energy of a sensor node when fully charged. Each tentative cluster head will then broadcast a message to announce its existence. Other sensor nodes receive such messages will select the cluster head with the smallest cost, which is defined by the *average minimum reachability power (AMRP)* of that cluster head, to serve as their cluster heads. AMRP is the average value of the minimum levels of transmission powers required by all sensor nodes within the cluster to communicate with the cluster head, which provides an estimation of the energy consumption for communications. Every sensor node then increases its probability to $p_i = \min(2 \times p_i, 1)$ in the next iteration. The above process is repeated until each sensor node can join a cluster.

6.3 Chain-Based Data Aggregation Schemes

Unlike the cluster-based data aggregation schemes where sensing data are collected and aggregated by cluster heads, the chain-based data aggregation schemes make each sensor node transmit its sensing data to the *nearest* neighbor. In this way, the network will form a long chain that connects all sensor nodes, as shown in Fig. 10. Except for the node(s) in the end of the chain, all sensor nodes along the chain will become the aggregation nodes.

The work of [34] develops a *power-efficient data-gathering protocol for sensor information systems (PEGASIS)*, where sensor nodes are organized into a linear chain for data aggregation. Such a chain can be formed by adopting a greedy algorithm, where all sensor nodes are assumed to have the global knowledge of the network. In particular, the farthest sensor node from the sink will initiate the chain formation operation. Then, in each iteration, the nearest neighbor (closer to the sink) of each sensor node is selected as its *successor* along the chain. After forming the chain, each sensor node will receive the sensing data sent from its neighbor (farther from the sink), aggregate the data with its own sensing data, and then transmit the aggregated data to its successor along the chain. The above process will be repeated until the sink receives the aggregated data of the whole network.

PEGASIS greedily constructs the chain for data collection and aggregation, but may not guarantee to minimize the total energy consumption of sensor nodes. Therefore, the work of [35] proposes a chain-construction scheme that minimizes the total energy consumption by reducing the value of $\sum D^2$, where D is the distance between any two adjacent sensor nodes along the chain. Similarly to PEGASIS, this scheme also starts the chain-construction operation at the sensor node farthest from the sink. Then, in each iteration, a new sensor node is inserted into the chain such that adding this new sensor node can increase the minimum value of $\sum D^2$ of the current chain. The above process is repeated until all sensor nodes are inserted into the chain. In this way, this scheme incurs a time complexity of $O(n^3)$, where n is the number of sensor nodes.

6.4 Sector-Based Data Aggregation Schemes

The work of [36] proposes a sector-based data aggregation scheme, called the *semantic/spatial correlation-aware tree (SCT)* scheme. SCT considers a circular WSN centered at the sink and with radius of R, as shown in Fig. 11. To efficiently collect and aggregate sensing data, the network is divided into m concentric *rings*, where each ring has the same width of $\frac{R}{m}$. Each ring is further divided into *sectors* of the same size such that each sector contains approximately the same number of sensor nodes (assuming that all sensor nodes are uniformly distributed in the network). For each sector, an *aggregation node* is selected to collect and aggregate the sensing data sent from other sensor nodes in the sector. Then, an aggregation tree is constructed by connecting each aggregation node in the ith ring to its upstream aggregation node in the $(i-1)$th ring through the shortest path. After constructing the tree, each aggregation node can transmit the aggregated data to the sink.

In SCT, all sensor nodes are assumed to know their geographic positions. Then, the sink broadcasts a message containing its position, the total number of sensor nodes in the network, the radius of the network (that is, R), the number of rings (that is, m), and the desired number of sensor nodes in each sector, to the network to form the ring structure. Once receiving such a message, each sensor node can determine the ring and the sector that it should belong to. Besides, the sensor node can also determine the sector boundary. In particular, by adopting a polar coordinate system, the coordinate of each point can be denoted by (r, θ), where r is the distance between the point and the polar (for example, the position of the sink) and θ is the included angle with the polar axis. Then, the coordinates of the boundary of a sector in the ith ring can be represented by $((i-1)\cdot\frac{R}{m}, \alpha)$ and $((i-1)\cdot\frac{R}{m}, \beta)$, where α and β are the bounding angles of that sector. In SCT,

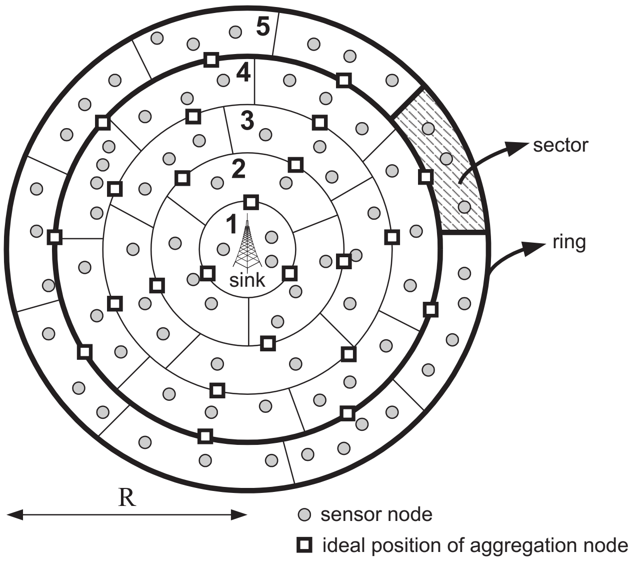


Fig. 11: The ring structure of SCT. The network is divided into multiple concentric rings and each ring is further divided into same-sized sectors. In each sector, an aggregation node is selected to collect and aggregate the sensing data sent from all other sensor nodes in that sector. The ideal position of the aggregation node in a sector should be located at the geometric center of the lower arc bounding that sector.

the ideal position of the aggregation node in a sector in the ith ring should be located at the geometric center of the lower arc bounding that sector, that is, the coordinate of $((i-1)\cdot\frac{R}{m}, \frac{\alpha+\beta}{2})$. However, if there is no sensor node located at this ideal position, the sensor node closest to that position will become the aggregation node in the sector. Then, all other sensor nodes can transmit their sensing data to the aggregation node by a location-based routing protocol.

7 Conclusion

Sensor nodes are usually battery-powered and thus how to conserve their energy is a primary concern in WSNs. In-network data compression can help reduce the amount of sensing data that sensor nodes have to regularly report to the sink(s) and therefore significantly reduce their energy consumption. This chapter provides a comprehensive survey of recent research on the data compression techniques in WSNs. Five categories of techniques, including string-based compression, image-based compression, distributed source coding, compressed sensing, and data aggregation, have been discussed. Table 3 gives a comparison of these data compression techniques. For data recoverability, the string-based compression, distributed source coding, and compressed sensing techniques provide lossless data compression. Some minor features of sensing data may be lost in the image-based compression techniques. On the other hand, the sensing data compressed by the data aggregation techniques are usually unrecoverable. For network structure, both DIMENSIONS and MRCQ will organize the network into a hierarchical architecture. EADAT and PEDAP need to maintain the data aggregation tree. LEACH and HEED will group sensor nodes into multiple clusters. PEGASIS and the work in [35] will form a long chain to collect the sensing data sent from all sensor nodes. SCT divides the network into a ring/sector structure. For compression theory, DIMENSIONS and MRCQ are based on the wavelet transformation. The studies of [23], [24] extend the Slepian-Wolf theorem while the work of [26] adopts the compressed sensing theorem to compress sensing data. Finally, for assumption, DIMENSIONS, MRCQ, and the studies of [23], [24] are based on the assumption that sensing data exhibit highly spatial and temporal correlation. In LEACH, the long-range communication capability of cluster heads is required to transmit their aggregated data directly to the sink. On the other hand, SCT considers a circular WSN where the sink is located at the center of the network.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Comm. Magazine*, vol. 40, no. 8, pp. 102–114, 2002.
[2] P. Zhang, C.M. Sadler, S.A. Lyon, and M. Martonosi, “Hardware design experiences in ZebraNet,” *Proc. ACM Int’l Conf. Embedded Networked Sensor Systems*, pp. 227–238, 2004.
[3] Y.C. Tseng, Y.C. Wang, K.Y. Cheng, and Y.Y. Hsieh, “iMouse: an integrated mobile surveillance and wireless sensor system,” *Computer*, vol. 40, no. 6, pp. 60–66, 2007.
[4] T. Wark, P. Corke, P. Sikka, L. Klingbeil, Y. Guo, C. Crossman, P. Valencia, D. Swain, and G. Bishop-Hurley, “Transforming agriculture through pervasive wireless sensor networks,” *IEEE Pervasive Computing*, vol. 6, no. 2, pp. 50–57, 2007.
[5] Y.M. Huang, M.Y. Hsieh, H.C. Chao, S.H. Hung, and J.H. Park, “Pervasive, secure access to a hierarchical sensor-based healthcare monitoring architecture in wireless heterogeneous networks,” *IEEE J. Selected Areas in Comm.*, vol. 27, no. 4, pp. 400–411, 2009.
[6] L.W. Yeh, Y.C. Wang, and Y.C. Tseng, “iPower: an energy conservation system for intelligent buildings by wireless sensor networks,” *Int’l J. Sensor Networks*, vol. 5, no. 1, pp. 1–10, 2009.
[7] M. Cardei and D.Z. Du, “Improving wireless sensor network lifetime through power aware organization,” *ACM Wireless Networks*, vol. 11, no. 3, pp. 333–340, 2005.
[8] Y. Zou and K. Chakrabarty, “A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks,” *IEEE Trans. Computers*, vol. 54, no. 8, pp. 978–991, 2005.
[9] Q. Zhao and M. Gurusamy, “Lifetime maximization for connected target coverage in wireless sensor networks,” *IEEE/ACM Trans. Networking*, vol. 16, no. 6, pp. 1378–1391, 2008.
[10] N. Heo and P.K. Varshney, “Energy-efficient deployment of intelligent mobile sensor networks,” *IEEE Trans. Systems, Man and Cybernetics – Part A: Systems and Humans*, vol. 35, no. 1, pp. 78–92, 2005.
[11] Y.C. Wang, C.C. Hu, and Y.C. Tseng, “Efficient placement and dispatch of sensors in a wireless sensor network,” *IEEE Trans. Mobile Computing*, vol. 7, no. 2, pp. 262–274, 2008.
[12] M. Nelson and J. L. Gailly, *The data compression book*, MIS Press, 1996.
[13] D.S. Taubman and M.W. Marcellin, *JPEG2000: fundamentals, standards and practice*, Kluwer Academic Publishers, 2002.
[14] T.A. Welch, “A technique for high-performance data compression,” *Computer*, vol. 17, no. 6, pp. 8–19, 1984.
[15] C.M. Sadler and M. Martonosi, “Data compression algorithms for energy-constrained devices in delay tolerant networks,” *Proc. ACM Int’l Conf. Embedded Networked Sensor Systems*, pp. 265–278, 2006.
[16] R.M. Rao and A.S. Bopardikar, *Wavelet transforms: introduction to theory and applications*, Addison Wesley Publications, 1998.
[17] D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, and R. Govindan, “Multiresolution storage and search in sensor networks,” *ACM Trans. Storage*, vol. 1, no. 3, pp. 277–315, 2005.
[18] Y.C. Wang, Y.Y. Hsieh, and Y.C. Tseng, “Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications,” *IEEE Trans. Computers*, vol. 58, no. 6, pp. 827–838, 2009.
[19] Wavelet image compression construction kit. [Online]. Available: http://www.geoffdavis.net/dartmouth/wavelet/wavelet.html
[20] H. Nyquist, “Certain topics in telegraph transmission theory,” *Proc. IEEE*, vol. 90, no. 2, pp. 280–305, 2002.
[21] N. Ahmed, T. Natarajan, and K.R. Rao, “Discrete cosine transfom,” *IEEE Trans. Computers*, vol. 1, no. 23, pp. 90–93, 1974.
[22] D. Slepian and J.K. Wolf, “Noiseless coding of correlated information sources,” *IEEE Trans. Information Theory*, vol. 19, no. 4, pp. 471–480, 1973.
[23] S.S. Pradhan, J. Kusuma, and K. Ramchandran, “Distributed compression in a dense microsensor network,” *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 51–60, 2002.

work	category	data recoverability	network structure	compression theory	assumption
S-LZW [15]	string-based compression	lossless	—	—	—
DIMENSIONS [17]	image-based compression	loss	hierarchy	wavelet transformation	correlated data
MRCQ [18]	image-based compression	loss	hierarchy	wavelet transformation	correlated data
reference [23]	distributed source coding	lossless	—	Slepian-Wolf theorem	correlated data
reference [24]	distributed source coding	lossless	—	Slepian-Wolf theorem	correlated data
reference [26]	compressed sensing	lossless	—	compressed sensing	—
EADAT [29]	data aggregation	unrecoverable	tree	—	—
PEDAP [30]	data aggregation	unrecoverable	tree	—	—
LEACH [32]	data aggregation	unrecoverable	cluster	—	long-range communication capability
HEED [33]	data aggregation	unrecoverable	cluster	—	—
PEGASIS [34]	data aggregation	unrecoverable	chain	—	—
reference [35]	data aggregation	unrecoverable	chain	—	—
SCT [36]	data aggregation	unrecoverable	ring/sector	—	circular WSN

TABLE 3: Comparison of data compression techniques in WSNs.

[24] Z. Xiong, A.D. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 80–94, 2004.

[25] D. Donoho, "Compressed sensing," *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[26] J. Haupt, W.U. Bajwa, M. Rabbat, and R. Nowak, "Compressed sensing for networked data," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 92–101, 2008.

[27] R. Rajagopalan and P.K. Varshney, "Data-aggregation techniques in sensor networks: a survey," *IEEE Comm. Surveys & Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

[28] C. Hua and T.S.P. Yum, "Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks," *IEEE/ACM Trans. Networking*, vol. 16, no. 4, pp. 892–903, 2008.

[29] M. Ding, X. Cheng, and G. Xue, "Aggregation tree construction in sensor networks," *IEEE Vehicular Technology Conf.*, pp. 2168–2172, 2003.

[30] H.O. Tan and I. Korpeoglu, "Power efficient data gathering and aggregation in wireless sensor networks," *ACM SIGMOD Record*, vol. 32, no. 4, pp. 66–71, 2003.

[31] R.C. Prim, "Shortest connection networks and some generalizations," *Bell System Technology J.*, vol. 36, no. 1, pp. 1389–1401, 1957.

[32] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Comm.*, vol. 1, no. 4, pp. 660–670, 2002.

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

[34] S. Lindsey, C. Raghavendra, and K.M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *IEEE Trans. Parallel and Distributed Systems*, vol. 13, no. 9, pp. 924–935, 2002.

[35] K. Du, J. Wu, and D. Zhou, "Chain-based protocols for data broadcasting and gathering in sensor networks," *Proc. IEEE Int'l Symp. Parallel and Distributed Processing*, 2003.

[36] Y. Zhu, R. Vedantham, S.J. Park, and R. Sivakumar, "A scalable correlation aware aggregation strategy for wireless sensor networks," *Information Fusion*, vol. 9, no. 3, pp. 354–369, 2008.