International Journal of Informatics and Communication Technology (IJ-ICT)
Vol. 10, No. 2, August 2021, pp. 104∼115
ISSN: 2252-8776, DOI: 10.11591/ijict.v10i2.pp104-115

An efficient forward error correction code for wireless sensor networks

Salah A. Alabady
College of Engineering, Computer Engineering Department, University of Mosul, Iraq

Article Info

Article history:

Received Aug 14, 2020
Revised Jan 27, 2021
Accepted Feb 3, 2021

Keywords:

Channel coding
Error correction code
FEC for IoT
LCPC
LDPC
RS

ABSTRACT

The main requirements in the design of wireless sensor network applications are to minimize energy consumption and maximize battery lifetime. Power is primarily consumed during wireless transmission and reception. Automatic repeat request (ARQ) and forward error correction (FEC) are the two basic methods to recover erroneous packets. As energy conservation is a major issue of concern in wireless sensor networks, repeat transmission because the error in the data received is not an option, and FEC would be preferred over ARQ. FEC is applied in situations where retransmissions are relatively costly or impossible. A successful data transmission means a higher energy saving and a long-life network. This paper presents a novel linear block forward error correction code for wireless sensor network applications called Low Complexity Parity Check (LCPC). The LCPC code offers lower encoding and decoding complexity than other types of codes. To validate the performance of the LCPC code, the proposed coding scheme was investigated at different values of data transmission with different types of modulations over Additive white Gaussian noise (AWGN) and Rayleigh fading channels. The simulation results show that the proposed code outperforms the renowned LDPC (8, 4), (255,175), and (576, 288) codes.

Corresponding Author:

Salah A. Alabady
College of Engineering, Computer Engineering Department
University of Mosul
Al Majmoaa Street, Mosul, Iraq
Email: eng.salah@uomosul.edu.iq

1. INTRODUCTION

In information and coding theory, error detection and correction are techniques that enable the reliable delivery of digital data over unreliable communications channels [1]. Transmission over the wireless channels especially in high reliability and high data rate wireless network applications is prone to noise and interference which causes the reception of erroneous packets at the receiving node. Automatic Repeat Request (ARQ) techniques are usually used in wireless sensor networks (WSNs) to tackle erroneous transmission, but they become inefficient with respect to energy and delay in hard environments [2]. WSNs consist of large number of small sensor devices with sensing, computing and communication capabilities. These sensors have limited resources such as limited memory, short communication range, low bandwidth, less CPU cycles and very limited energy [3]. The constrained energy has a direct impact on the lifetime of sensor networks. Moreover, the energy expended for computation is much less than the energy expended for communication. For example, in Berkeley motes, the energy consumed for transmitting a single bit is equivalent to the energy consumed for executing 800 CPU instructions. In general, the energy consumption in WSNs is involved in three main activities which

are sensing, processing and communication [4]. The largest amount of energy is consumed during the transmission and reception of information. Indeed minimizing the number of transmissions can prolong the lifetime of WSNs. Therefore, minimizing energy consumption is of most importance in designing WSNs.) ARQ techniques introduce high latency, where the repeated retransmission consumes considerable time, which leads to a high delay between the detection moment of an event at the sensor nodes and informing the base station about that event. Retransmission also consumes a large amount of energy from both transmitting and receiving nodes. To reduce the number of retransmission for lossy channels, Error Correcting Codes are commonly used [5].

Best networks must be able to transfer data from source to destination or from one node to another node with complete accuracy and reliability. Therefore, for reliable networks, errors must be detected and correct at the destination node without the need to send a retransmit request again from the sender node. By using the error detection and correction techniques the network performance will improve such as increase the throughput, decrease the BER besides reduction end-to-end delay. The main design interest for any applications of WSNs is the limited energy supply, limited computation capability, and communication range of sensor nodes as compared with other computing and communicating devices [4], [6]. The lifetime of WSN depends on the lifetime of the battery of individual sensor nodes. One way to conserve the energy in WSNs is to avoid retransmission due to error as far as possible and instead use an efficient error control scheme for error correction. Utilization of the Automatic Repeat Request (ARQ) is limited for sensor networks due to the additional retransmission energy cost and overhead.

Forward Error Correction (FEC) Code strategies allow the receiver to detect and correct errors within some bound. The advantage of FEC is that retransmission of data can be avoided. The most significant challenge in sensor networks is to overcome the energy constraints since each sensor node has limited energy to consume. Since data transmitted over the wireless media is vulnerable to corruption by noise, error control schemes are necessary to keep the Bit Error Rate (BER) low. Due to the stringent energy constraint, it is impossible to increase the signal power of the transmitted signal in wireless sensor networks. Hence an alternative way is to use the error control codes to reduce the BER.

Many approaches of error detection code can detect accidental changes in digital data over unreliable communication channels but without correcting these errors. Cyclic redundancy check (CRC) and frame check sequence (FCS) are some of the methods of the error detection which commonly used in the data link layer (e.g., frame header) and network layers (e.g., packet header)[7]. In both cases CRC and FCS, the errors can be detected but cannot be corrected [8]. The correction in these cases is attempted by retransmission of the original data once again. The frame or packet dropped if the retransmitted information still erroneous, which causes wasted time and bandwidth that leads to degradation in network performance. Another issue of this problem becomes (i.e., frame or packet is dropped) more critical in real-time applications, such as video conference and remote control in WSNs that depend on a live broadcast.

One type of error detection and correction is the Hamming code that can detect and correct single-bit error and detect double-bit errors but cannot correct. The reason that Hamming code cannot correct double bit error is limited of number of syndromes. Base on the H matrix of Hamming code that included 3 rows and 7 columns there are 8 values of the syndrome [7]. In the case of a single-bit error, there are 7 possibilities of error pattern when the codeword length equals 7 bits, in that case, each error pattern assigned by one syndrome vector. In the case of double-bit errors, there are 21 possibilities of error pattern when the codeword length equals 7 bits and the Hamming code did not have this number of the syndrome (i.e. 21), therefore each 3 error patterns assigned by one syndrome vector. This makes the correction operation is difficult and impossible because it cannot decide correctly which the error pattern is correct from the 3 error patterns. In addition, the Hamming code cannot detect more than two-bit errors (e.g., burst error) [8], [9]. Reed Solomon (RS) [10] code is one of the famous error correction codes, it is the subset of the Bose, Chaudhuri, and Hocquenghem (BCH) codes as well as linear block codes. A particular RS code specified as RS (n, k) with s-bit symbols. The number and type of errors that can be corrected in RS code depend on the characteristics of that code [11]. An RS decoder can correct up to t symbols that contain errors in a code word, where $2t = n\text{-}k$. To increase the capability of error correction, the number of the parity code must increase. This means, the value of t in RS code must be large. Likewise, the error correction capability of the low-density parity-check (LDPC) code also depends on the codeword length and the characteristic of the parity check matrix. The decoder gives a better performance with a larger code word and with good parity-check matrices. In practice, to achieve a better BER performance with LDPC codes and close to the channel capacity, the length of the LDPC codeword used is in the order of thousands of bits [12]. The matrix multiplication for that big codeword size demands huge memory,

computational requirements, and more complexity to the decoding [13]. In any way, the decoder fails to correct errors if the number of errors occurred is greater than the error correction capability of the decoder regardless of the number of iterations. N. Abughalieh, K. Steenhaut, and A. Nowe [14], ECC technique for sensor networks based on Turbo Codes is presented. The low complexity turbo encoder is deployed at the source nodes, while the iterative decoding process is shifted to the Base Station. The high processing power requirements make turbo decoders hard to run at the sensor nodes since the nodes have limited processing and memory resources [15]. Although there are various encoding schemes, such as Reed-Solomon codes, convolutional codes, and LT codes [16], none of them are directly applicable to the WSNs. Since a typical wireless sensor node currently has a low processing power and a small memory, it needs an energy-efficient error correction scheme. As Reed-Solomon and LT code require high processing power and storage, they may not ideal for WSNs. In addition, implementing coding techniques introduces a high delay in delivering packets to the base station. The delay comes from the coding and decoding processes that run on each node in the network while routing data to the base station.

The main difference between our proposed code and the other codes such as RS, BCH, and LDPC, is that a simple encoding and decoding method with low complexity moreover not require huge memory and not require iteration process [17]. To store the tables of syndrome vector and error pattern for single and double-bit error, we need 585 bits (around 74 bytes) from the memory size. The number of one element in the G and H matrices in our proposed code is lower than the number of zero elements. There are 12 one elements from 36 elements in the G matrix and 13 one elements from 45 elements in the H matrix. These low numbers of ones in the G and H matrices lead to reduce the number of addition and multiplication operations and this makes our proposed code low complexity compare with other types of codes. For example in Hamming code (7, 4) there are 13 one elements from 24 elements in the G matrix and 12 one elements from 21 elements in the H matrix. This means there is 54.16 % from one element in the G matrix and 57.14% from one elements in the H matrix, while there is 33.33% and 28.88% from one elements in the G and H matrices in our proposed code LCPC code [18]. On the other hand, there are 16 one elements from 21 elements in the H matrix in LDPC code (8, 4) this means there is 50% from one elements in the H matrix. We compare the performance of the LCPC code with other codes approaches such as Hamming, RS, and LDPC codes over AWGN and Rayleigh fading channels with BPSK, 4-QAM, and 16-QAM modulations. The simulation results show that the proposed LCPC code improves the BER performance compared with other code types and this leads to an increase in the throughput due to a decrease in the retransmission process. Moreover, LCPC code has less complexity and lower size memory requirement compared with RS and LDPC codes. Additionally, LCPC code does not require reiteration in the detection and correction error process. We used the same concept of Hamming codes for proposed the generator matrix G in the LCPC code. However, the parity matrix in the G and H matrices is different. In addition, the mean difference between the LCPC code and Hamming code is that the latter can correct the single-bit error and detect the double-bit error. While LCPC code can detect and correct a (1, 2, 3, 6, 7, 8, and 9) bits error from the codeword length 9 bits. The remainder of this paper is organized as follows. Section II provides a description of the proposed approach LCPC code. Section III, presents the performance of LCPC code and simulation results analysis with suggestions for some applications. Finally, section IV concludes.

2. PROPOSED LCPC (9, 4) CODE

In this section, we present the proposed linear block code LCPC (9, 4) that use to detect and correct single and double-bit errors. Moreover, the LCPC code can detect and corrects more than double-bit errors (Burst Error) in some cases that will explain later in this section. Our code has the capability to detect and corrects consecutive and non-consecutive bit errors. The LCPC code is defined as block code (9, 4) where the code word length n = 9 and the sample data length k = 4. The LCPC (9, 4) code includes four main units defined as follows; encoding, error detection, error correction, and decoding. This section presents the LCPC (9, 4) block code, which encodes symbol with length (k = 4) bits into a codeword length n = 9. Equations (1) and (2) show **G** and **H** matrices for LCPC (9, 4) respectively.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (1)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

The number of one's in **G** and **H** matrices for the proposed LCPC codes is lower than the number of zeros. Matrix **G** has 12 one out of 36 elements (4 rows and 9 columns) in the matrix. Matrix **H** has 13 one out of 45 elements (5 rows and 9 columns) in the matrix. The lower number of one in **G** and **H** matrices reduces the number of addition and multiplication operations in encoding and decoding processes and this lead to reduce code complexity [9]. Table 1 shows the comparison between LCPC and LDPC codes in terms of the number of one in **G** and **H** matrices as well as their percentage. LCPC codes have a lower number of one or percentage of one, thus having lower complexity. Because the complexity of multiplying a codeword with a matrix depends on the amount of 1's in the matrix [9].The complexity of LCPC code is $O(n)$, where n is the codeword length. Where n = 9 for LCPC (9, 4).

Table 1. Comparison for number and percentage of (1) in **G** and **H** matrices for LDPC, Hamming, and LCPC codes

Type of code Size of **H** matrix	Size of **G** matrix Number and percentage of (1) in the **H** matrix	Number and percentage of (1) in the **G**matrix
Hamming [1],[2]	4 × 7 = 28	13 (46.42%)
3 × 7 = 21	12 (57.14%)	
LDPC [8],[10]	4 × 8 = 32	16 (50%)
4 × 8 = 32	16 (50%)	
LCPC (proposed)	4 × 9 = 36	**12 (33.33%)**
5 × 9 = 45	**13 (28.88%)**	

2.1. LCPC encoding

The encoding process of LCPC codes is comparatively simple as it involves just matrices multiplication. The first step in the encoding units is the segmentation of the source data that are scheduled to send into samples; each sample includes k binary bits. The samples of source data denoted by $SD_i = (v_0, v_1, ... v_{k-1})$, $1 \leq i \leq j$, where j is the number of samples of the source data, v is a binary bit, and k = 4 is the lengths of a binary vector. To start transmit and encode the source data, two samples from the source data are take, for example (SD_1 and SD_2). The second step is the XOR operation that used to create SD_3 sample from SD_1 and SD_2 samples. We will mention later in this section the benefits of SD_3. The third step is the encoding that implemented using a polynomial function. In the encoding step the redundant bits r will be added to each sample. Length of the code word is equal to n , where $n = k + r$, and r = 5. A polynomial function $f\ (\alpha)$ can representation of the form,

$$f_n\alpha_n + f_{n-1}\alpha_{n-1} + ... + f_0 \quad (3)$$

where, $f_i \in$ GF (q), i = 0, 1, ... , n, and $q = 2^m$

In this case, the corresponding vector of a polynomial $f_{(\alpha)}$ would be as follows [$f_n\ f_{(n-1)}\ f_{(n-2)}$.... f_0]. The LCPC code is constructed using Galois Field, GF (2^5). As a rule, if the polynomial function used to construct the LCPC code, the first thing is choosing a generator polynomial $G\ (\alpha)$. The $G(\alpha)$ uses for encoding, where (α) is a primitive n^{th} root of the $G\ (\alpha)$ and a primitive element of GF (2^5). The generator polynomial $G(\alpha)$ that we have proposed depending on the Tanner graph as shown in Figure 1, Table 2 shows Code words of the LCPC (9, 4) code.

$$G(\alpha) = \alpha^5 + \alpha + 1 \quad (4)$$

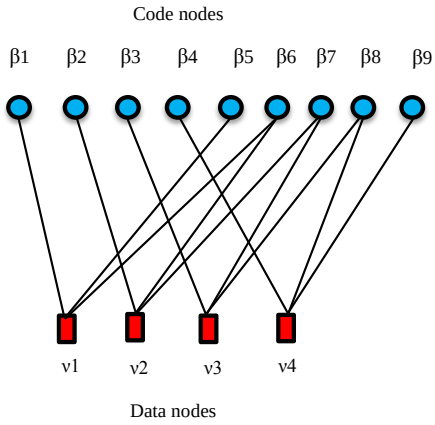

Figure 1. Tanner graph of a generator matrix G for the LCPC (9, 4) code.

To generate a code word polynomial CD_{Ti} (α), we multiply G (α) by a message polynomial SD_i (α). The generator polynomial $G(\alpha)$ is denoted by;

$$G(\alpha) = G_0 + G_1\alpha + ... + G_{n-1}\alpha^{n-1} + G_n\alpha^n \tag{5}$$

Depending on our proposed Tanner graph that shown in Figure 2, the parity check $H(\alpha)$ is defined by the follows:

$$H(\alpha) = \alpha^5 + 1 \tag{6}$$

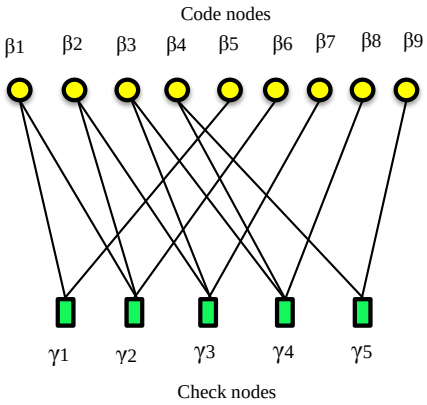

Figure 2. Tanner graph of a check matrix H for the LCPC (9, 4) code

The transmitted code word CD_{Ti} (α) can be expressed as:

$$CD_{Ti}(\alpha) = SD_i(\alpha) \times G(\alpha) \tag{7}$$

where,

$$CD_{Ti}(\alpha) = \beta_0 + \beta_1\alpha + ... + \beta_{n-1}\alpha^{n-1} \tag{8}$$

The message polynomial $SD_i(\alpha)$ that is a sample of source data can be expressed as:

$$SD_i(\alpha) = v_0 + v_1\alpha + ... + v_{k-1}\alpha^{k-1} \tag{9}$$

Table 2. Code words of the LCPC (9, 4) code

Message Binary	Message polynomial $SD_i(\alpha)$	Code word Polynomial $CD_{Ti}(\alpha)$	Code word Binary
0 0 0 0	0	0	0 0 0 0 0 0 0 0 0
0 0 0 1	1	$\alpha^5 + \alpha + 1$	0 0 0 1 0 0 0 1 1
0 0 1 0	α	$\alpha^6 + \alpha^2 + \alpha$	0 0 1 0 0 0 1 1 0
0 0 1 1	α +1	$\alpha^6 + \alpha^5 + \alpha^2 + 1$	0 0 1 1 0 0 1 0 1
0 1 0 0	α^2	$\alpha^7 + \alpha^3 + \alpha^2$	0 1 0 0 0 1 1 0 0
0 1 0 1	$\alpha^2 + 1$	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$	0 1 0 1 0 1 1 1 1
0 1 1 0	$\alpha^2 + \alpha$	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha$	0 1 1 0 0 1 0 1 0
0 1 1 1	$\alpha^2 + \alpha + 1$	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + 1$	0 1 1 1 0 1 0 0 1
1 0 0 0	α^3	$\alpha^8 + \alpha^4 + \alpha^3$	1 0 0 0 1 1 0 0 0
1 0 0 1	$\alpha^3 + 1$	$\alpha^8 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1$	1 0 0 1 1 1 0 1 1
1 0 1 0	$\alpha^3 + \alpha$	$\alpha^8 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$	1 0 1 0 1 1 1 1 0
1 0 1 1	$\alpha^3 + \alpha + 1$	$\alpha^8 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + 1$	1 0 1 1 1 1 1 0 1
1 1 0 0	$\alpha^3 + \alpha^2$	$\alpha^8 + \alpha^7 + \alpha^4 + \alpha^2$	1 1 0 0 1 0 1 0 0
1 1 0 1	$\alpha^3 + \alpha^2 + 1$	$\alpha^8 + \alpha^7 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1$	1 1 0 1 1 0 1 1 1
1 1 1 0	$\alpha^3 + \alpha^2 + \alpha$	$\alpha^8 + \alpha^7 + \alpha^6 + \alpha^4 + \alpha$	1 1 1 0 1 0 0 1 0
1 1 1 1	$\alpha^3 + \alpha^2 + \alpha + 1$	$\alpha^8 + \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + 1$	1 1 1 1 1 0 0 0 1

2.2. LCPC decoding

During the transmission, the codewords expect to expose attenuation and distortion due to interference, noise, and multipath fading in wireless network systems. Therefore, maybe there are errors in some bits in the codeword received at the receiver side. In the decoding unit, there are three procedures. First is syndrome vector computation for error detection. Second is determinate the error pattern depending on the syndrome vector. The third is error correction by adding the error pattern to the error codeword received. Error detection is the ability to detect errors in one bit or more than one bits. Whereas, error correction has an additional feature that enables identification and corrects the errors. The first step of the decoding process is error detection. The aim of the detection is to check is there are any errors in the codeword received (CD_{Ri}). The LCPC code used the parity check equation $H(\alpha)$ for this purpose. The relationship used for error detection in the codeword received can be expressed as (8).

$$SY(\alpha) = H(\alpha) \times CD_{Ri}(\alpha) \tag{10}$$

Where $SY = (\gamma_0, \gamma_1, \ldots \gamma_r)$ is the syndromes vector and $CD_{Ri}(\alpha)$ is the codeword received that denoted by (9). The codeword received defined as the codeword transmitted with error pattern $EP(\alpha)$. The value of SY depending on the type of error (single or double-bit errors) and on the position of the bit error in the $CD_{Ri}(\alpha)$. Then, depending on the SY value, we can determine the error pattern (EP). The EP is an error pattern vector that results from a noisy channel.

$$CD_{Ri}(\alpha) = CD_{Ti}(\alpha) + EP(\alpha) \tag{11}$$

By substituting (9), into (8), $SY(\alpha)$ can be rewritten as follows:

$$SY(\alpha) = H(\alpha) \times (CD_{Ti}(\alpha) + EP(\alpha)) \tag{12}$$

$$SY(\alpha) = H(\alpha) \times CD_{Ti}(\alpha) + H(\alpha) \times EP(\alpha) \tag{13}$$

$$SY(\alpha) = H(\alpha) \times EP(\alpha) \tag{14}$$

The multiplication between $H(\alpha)$ and $G(\alpha)$ or $CD_{Ti}(\alpha)$ is equal to zero. This is because of the property that any row in the H matrix is orthogonal to the rows of the G matrix, which is the inner product of a row in G with a row in H will be zero. From the (12) the syndromes vector (SY) is only dependent on the error pattern (EP). Therefore, the error detection is implemented by calculating the SY value to check is there any error in the three codewords received (CD_{R1}, CD_{R2} and CD_{R3}) for the three code words that transmitted (CD_{T1}, CD_{T2} and CD_{T3}) respectively. For any codewords received if SY is the null vector (all zeros) this indicates the received codeword is error-free. On the other hand, if the value of SY is non-zero, then the value

indicates some bit has been changed, and this means there are some errors (single, double, or more than double bits) in that code word received. The type of error is known from the *SY* value. There are two tables for syndrome vector values with the specific error patterns, one for the single-bit error and the second is for the double-bit errors. We assume the two tables for syndrome vector are stored in the memory at the receiver. The tow tables exploit 79 Byte form the memory size. If the *SY* indicates that there is a one-bit error for both CD_{R1} and CD_{R2}, the LCPC code can correct the two codewords received without the need to CD_{R3}. Also, when the *SY* indicates that there are two-bit errors with one error pattern for both CD_{R1} and CD_{R2} code words, also the LCPC code can correct the two codewords without the need to CD_{R3}. However, if the *SY* indicates that there are two-bit errors with two error patterns for one or both the code words (CD_{R1} and CD_{R2}), on that case the LCPC code used CD_{R3} (after correction CD_{R3})) to correct the two code words.

Additionally, to detect and correct single and double bit errors, our proposed code has the capability to detect and correct more than double-bit errors (3 to 9-bit errors) in some cases. The first case, if for example there is one or two-bit errors with one error pattern in the code words (CD_{R1} and CD_{R3}) and more than two-bit errors in CD_{R2}, the LCPC code can recover and correct CD_{R2} after correct CD_{R1} and CD_{R3}. Whereas the second case, if the *SY* value indicates there are one or two-bit errors with one error pattern in the code words (SD_2 and SD_3) and more than two-bit errors in CD_{R1} on that case the LCPC code can recover and correct CD_{R1} after correct (SD_2 and SD_3). However, the case that the LCPC (9,4) code can detect the errors but cannot correct it and need to retransmit the SD_1 and SD_2 again when there are more than two-bit errors in the two codewords received CD_{R1}, CD_{R2} and CD_{R3}. The previous cases explain the benefits of the send SD_3 with the SD_1 and SD_2. In the case of single-bit error correction and for 9 bits code word length, there is 9 probability of the *EP*. Whereas, in the case of two-bit errors for 9 bits code word length, there is 36 probability of the *EP*. The number of probability EP (*NoEP*) can be expressed as:

$$NoEP = n!/e!(n-e)! \tag{15}$$

One problem in the correction double bit errors in our proposed code is that we have 36 probability of the *EP* whereas the maximum probability of the *SY* is 32 (2^5). This is causes to overlap in some cases of the *EP* (i.e., there are two *EP* have the same value of *SY*). In case for example the CD_{R1} has two *EP* this means there are two-correction possibilities. The same thing for CD_{R2} or CD_{R3}. The worst-case happens when each one of the CD_{R1}, CD_{R2}, and CD_{R3} have two *EP*. We solve this problem by using the CD_{R3} in the correction process, and this is another reason to sent SD_3.

Table 3 shows the syndrome vectors and their associated error patterns in case a single bit error. We did not mention the *SY* and *EP* Table for double-bit errors because of the limited number of pages. The authors have more detailed documented for Tables, flow charts and pseudo-code for error detection and correction function of LCPC (9,4) code. The correction process is achieved by adding the *EP* value to the error core word received as shown in (12).

$$\widehat{CD_{Ti}}(\alpha) = CD_{Ri}(\alpha) + EP(\alpha) \tag{16}$$

where the $\widehat{CD_{Ti}}(\alpha)$ is the corrected codeword received. The decoder to obtain the original source data send SD_i implemented by masking the last four bits of the codeword.

$$SD_i = \widehat{CD_{Ti}}(\alpha) \wedge (111100000) \tag{17}$$

Table 3. Error pattern versus syndrome vector polynomial of LCPC (9, 4) code for a single bit error

Position number of single bit error	Error Pattern (EP)	Syndrome Vector (SY)
0 0 0 0 0 0 0 0 1	1	α
0 0 0 0 0 0 0 1 0	α	α^2
0 0 0 0 0 0 1 0 0	α^2	α^3
0 0 0 0 0 1 0 0 0	α^3	α^4
0 0 0 0 1 0 0 0 0	α^4	$\alpha + 1$
0 0 0 1 0 0 0 0 0	α^5	$\alpha^2 + \alpha$
0 0 1 0 0 0 0 0 0	α^6	$\alpha^3 + \alpha^2$
0 1 0 0 0 0 0 0 0	α^7	$\alpha^4 + \alpha^3$
1 0 0 0 0 0 0 0 0	α^8	$\alpha^4 + \alpha + 1$

3. PERFORMANCE OF LCPC CODE AND SIMULATION RESULTS ANALYSIS

This section presents the performance of the proposed code at the AWGN channel with BPSK modulation. In addition, this section provides a comparison of LCPC (9,4) code with Hamming [9], RS [19], and LDPC [20] codes using a different value of codeword length. Figure 3 shows the results of the BER performance of the LCPC (9,4) code with the RS codes, especially at the parity code (6, 8, 10, 16, and 32) bytes. We investigate the performance of our proposed LCPC (9,4) code with a different codeword length of RS codes. We have considered the BPSK modulation and AWGN channel. As the graph shows, the BER performance of the RS code is becoming better when the number of parity codes increases. For illustration, at parity code 32 byte, the RS code is better than LCPC code at the SNR greater than 6 dB. But at the low SNR for AWGN channel the LCPC (9,4) code is better than the RS code. As well as, increased the parity code will increase the complexity of detection and correction of the errors additionally this increased the time delay.

To explain how and why the performance of our proposed code (LCPC) is better than the Hamming and RS codes as the results shown in Figure 3. We refer to some details about the capability of the error detection and correction for those codes. The minimum Hamming distance defined as d_{min} = *n-k*. The number of errors that a block code can detect and correct is determined by its minimum Hamming distance d_{min}. This is defined as the minimum number of places where any two codewords different. In general, the number of errors *V* that can be detected for a block code is *V* = *d-1*. For example at *m* = 3, the codeword length *n* = 7, message length *k* = 4 and d_{min} = 3. Where *t* is the number of errors that a block code can correct.

$$t = \lfloor \frac{n-k}{2} \rfloor = 1 \tag{18}$$

Since, the Hamming code has a minimum Hamming distance *d* = 3, it can only correct 1 bit error for each 7 bits transmitted. Therefore, the percentage of error correction is (1/7 = 14.285%). Likewise, in case Reed-Solomon (RS) code the number and type of errors that can correct depends on the characteristics of the RS code. RS codes are a subset of Bose - Chaudhuri - Hocquenghem (BCH) codes and are linear block codes. RS codes are burst error-correction codes. A Reed-Solomon code is specified as RS (*n, k*) with *s*-bit symbols. This means that the encoder takes *k* data symbols of each *s* bits and adds parity symbols to make an $n = 2^s - 1$, symbol code word. There are *n-k* parity symbols of *s* bits. An RS decoder can correct up to *t* symbols that contain errors in a code word, where *2t* = *n-k*. If *s* = 3 bits, then *n* = 7, and when the number of parity equal to 3, then *k* = 4. The number of symbols that contain errors that RS code can correct is *t*, where *t* as shown in 14. So, based on *t* value the RS (7,4) code can only correct one symbol errors from the 7 codeword symbols that transmitted. When the symbol sizes are 3 bits, the worst case happened when only a one-bit error occurs in separate symbols. In this case, the percentage of error correction is (1 / 21 = 4.7619%), this value is small compared with the percentage of error correction in Hamming (7,4) code, and this explains the reason why the Hamming (7,4) code has a better BER performance compare with the RS (7,4) code, as shown in Figure 3.

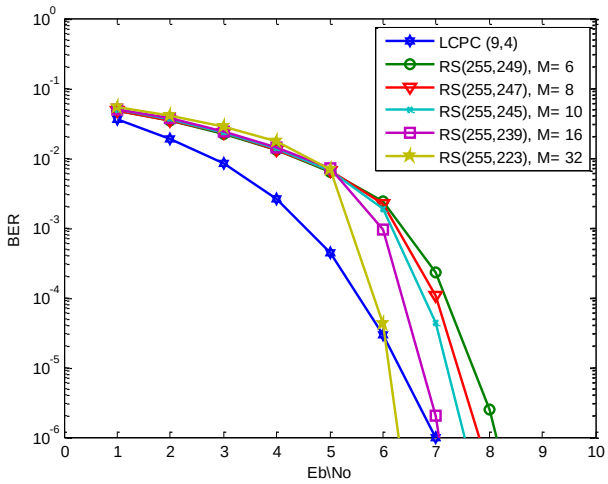

Figure 3. Comparison between LCPC (9, 4) code with different RS codes over AWGN channel.

The best case of error correction in RS (7,4) code occurs when there are only all bits in one symbol is errors. This means the percentage of error correction is the number of errors in one symbol over the total number of symbols bit transmitted, i.e. (3 / 21 = 14.285%). Compared with our proposed code LCPC the worst case happened when only a one-bit error in each codeword (9 bits). Based on our approach, there are three codewords transmitted each time. Two codewords contain the original source data (SD_1 and SD_2) and the other one codeword (SD_3) result of XOR the two original source data. The benefit of sending the third codeword (CD_{T3}) is in order to use it in a correction the error in the case that there are more than two-bit errors in one codeword that contains the original source data (CD_{T1} or CD_{T2}). The percentage of error correction is (2 / 18 = 11.11%) and the percentage of error correction is 22.22% when there are 2-bit errors in (CD_{T1} or CD_{T2}), the best percentage of error correction case happen when there are two-bit errors in one codeword and 9-bit errors in another codeword (11 / 18 = 61.11%). This explains how the LCPC code is better than Hamming, BCH, Golay, and RS codes.

To increase the capability of error correction, the number of the parity code must increase. This means, the value of *t* in RS code must be large. A popular Reed-Solomon code is RS (255, 223) with 8-bit symbols. Each block contains 255 code word bytes, of which 223 bytes are data and 32 bytes is parity. For this code: n = 255, k = 223, s = 8, $2t$ = 32, and t = 16. A large value of t means that a large number of errors can be corrected but requires more computing power than a small value of t. One symbol error occurs when 1 bit in a symbol is wrong or when all the bits in a symbol are wrong. RS (255, 223) decoder can correct 16 symbol errors. The worst-case occurred when 16-bit errors may occur, and each one of these bits is in a separate symbol (byte) so that the decoder can correct as maximum 16-bit errors from the 2040 bits (255x8). The percentage of error correction is (16/2040 = 0.784%). Whereas the best case occurred when a 16 complete byte error happens so that the decoder can correct a 128 bits (16x8) error from the 2040 bits. The percentage of error correction is (128/2040 = 6.274%).

As we see there are restrictions and limitations in the number of errors that the decoder can correct in each block and this limitation depending on the number of parity byte. The difference between the LCPC code and RS code is that the latter has the ability to correct a specific number of bit errors based on the parity number, whereas the LCPC code is not so. As an example in LCPC code, if we need to send 2040 bits (255 bytes), we segmentation these 2040 bits to around 227 blocks, each block contains 9 bits. So, the worst case is that the LCPC code can correct 1 bit in each one block (9 bits). We have 227 blocks, therefore the percentage of error correction is (227/2040 = 11.127%), and if there are two-bit errors in each block the percentage of error correction is (454/2040 = 22.25%). The best-case happened when there are half of 227 blocks that have two-bit errors, and another half of the 227 blocks have 9 bits error as the maximum. Therefore, the total number of errors that LCPC code can correct is equal to;(227 / 2) x 2 = 227 bits , and (227 / 2) x 9 = 1021 bits. The percentage of error correction is (227+1021)/ 2040 = 61.176%. Therefore, the proposed code LCPC is outperforming the renowned codes. Moreover, Figure 4 presents the comparison between LCPC (9, 4) code and other types codes such as Hamming, Golay, BCH (Soft, Hard), and RS codes with the Shannon limit over AWGN channels on BPSK modulation. The figure shows that the performance of LCPC (9, 4) code also is better than the other codes. As shown in Figure 4 to get a BER = 10^{-5}, we need 5.6 dB for a LCPC (9,4) code. The coding gain is 1.9, 2.5, 3.4, 3.6, 3.9 dB for Golay, BCH soft, Hamming, BCH soft, and RS codes respectively.The coding gain defined is the amount of additional SNR or Eb/No that would be required to provide the same BER performance for an uncoded signal.

Figure 5 presents the coding gain for LCPC (9, 4) code compared with the Hamming and RS codes in the AWGN channel. The figure evidently shows that the coding gain for LCPC (9, 4) code increases when the codeword length of Hamming and RS codes decrease. The minimum coding gain is equal to 1.4 dB at codeword length 255, whereas it is equal to 3.5 dB and 4 dB for Hamming and RS codes respectively at codeword length 7 when the BER equal to 10^{-5}. The saving power is one of the benefits of LCPC (9, 4) code. Therefore we can use our proposed code in wireless sensor network (WSN) because the power consumption and save battery life is very important things in WSN. Figure 6 presents the BER performance comparison between LCPC (9, 4) code and LDPC code (8, 4) at different types of decoding methods over AWGN channels using BPSK modulation. The figure shows that the performance of LCPC (9, 4) code is better than LDPC code when the bit flip decode method (BF) used and the coding gain is around 4 dB at BER equal 10^{-5}. Although, the BER performance of the LCPC (9, 4) code becomes near the LDPC code that used log domain and log domain simple decoding methods at low SNR. The performance of LCPC (9, 4) code becomes equal to the performance of the LDPC code that used the log domain decoding method when the SNR increased. In addition to better

performance of BER for our proposed LCPC code, the main advantage is low complexity of encoding and decoding comparing to the LDPC and RS codes and does not need reiteration in decoding for correcting the errors.

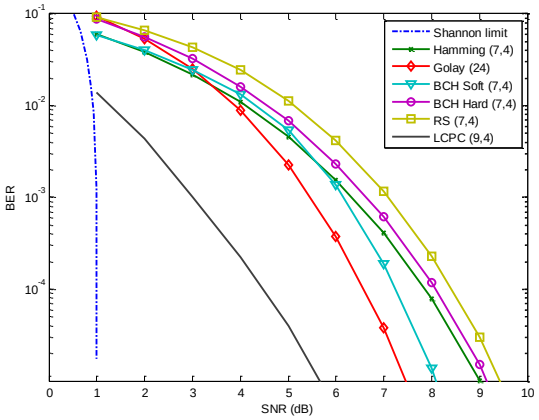

Figure 4. Comparison between LCPC (9, 4) code and another codes with Shannon limit over AWGN channel.

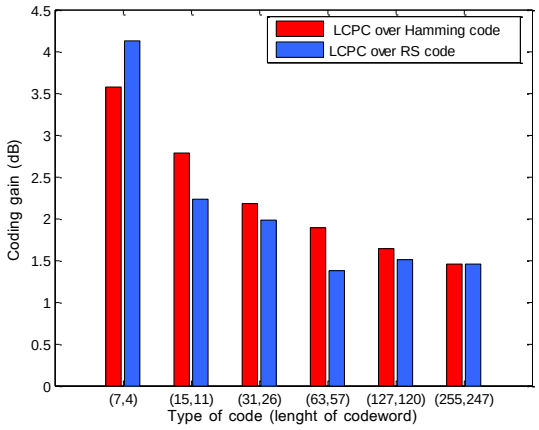

Figure 5. Coding gain for LCPC (9, 4) code over Hamming and RS codes at BER 10^{-5}.

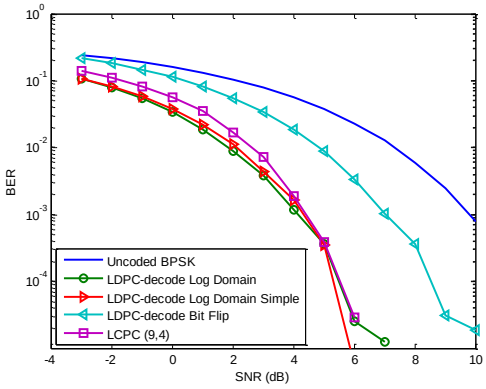

Figure 6. Comparison between LCPC (9, 4) code and LDPC (8, 4) at different types of decoding over BPSK AWGN channel at 8000 bits.

Figure 7 presents the BER performance comparison between LCPC (9, 4) code and Euclidean Geometry-Low Density Parity Check (EG-LDPC) (255,175) code [20] at different types of decoding methods over AWGN channels using BPSK modulation. These decoding methods include bit flipping decoding (BF), one-step majority-logic (MLG) decoding, weighted MLG decoding, and weighted BF decoding [20]. 7 shows that the BER performance of LCPC (9, 4) code is better than (EG-LDPC) (255,175) code at low SNR. On the other hand, at SNR 5 dB the BER performance of the LCPC becomes the same performance of the EG-LDPC code when the EG-LDPC BF and EG-LDPC weighted MLG decoding methods used. In addition, at SNR 3.5 dB the BER performance of the LCPC becomes the same performance of the EG-LDPC code when the EG-LDPC weighted the BF decoding method. Figure 8 shows the BER performance comparison between the LCPC (9, 4) code and LDPC (576, 288) in BPSK modulation over the Rayleigh fading channel at 41472 bits. The figure shows the LCPC (9, 4) code outperforms the LDPC (576, 288) code over the Rayleigh fading channel.

Although, some types of decoding for LDPC code have a better BER performance compare with our proposed code over AWGN channels such as EG-LDPC weighted BF decoding method. However, the LCPC

code has characteristics that distinguish it from LDPC codes such as; low complexity encoding and decoding, did not demand huge memory for matrix multiplication for that big codeword size, and did not need iterations in decoder compared with LDPC codes that need more than 20 times of iterations to correct the error code word. The benefits of the proposed code will improve the network performance such as increasing throughput, reducing end-to-end delay, and bit error rate for real-time applications.

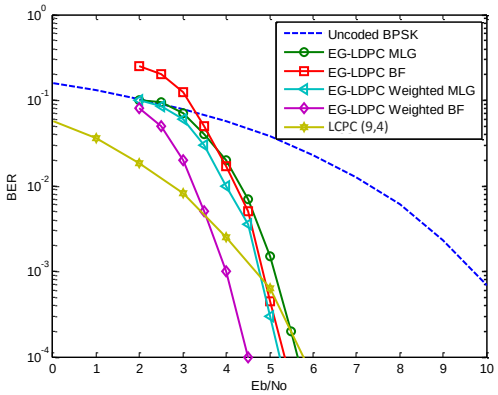

Figure 7. Comparison between LCPC (9, 4) code and LDPC (255,175) at different types of decoding over BPSK AWGN channel at 8000 bits.

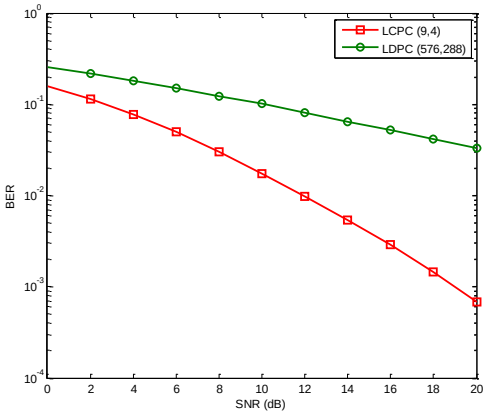

Figure 8. Comparison between LCPC (9, 4) code and LDPC (576, 288) at BPSK over Rayleigh fading channel at 41472 bit.

4. CONCLUSION

In this paper, an efficient FEC scheme for WSNs has been presented that called low complexity parity check (LCPC) code to avoid retransmission which saves energy. In addition, to detect and correct single and double bit errors, the LCPC code can correct more than two-bit errors in one codeword if the other two codewords have one or two-bit errors. The performance of the proposed LCPC code in the AWGN channel with BPSK modulation is investigated. Comparisons the BER performance are made between LCPC code and other codes such as Hamming, RS, BCH, Golay, and LDPC codes. The simulation results focus on the RS and LDPC codes, at different values of transmission data and different codeword lengths. The results indicate that the BER performance of the LCPC code outperforms the renowned codes. In addition, a good value of coding gain is get when comparing the BER performance of LCPC code with Hamming and RS codes. The results indicate the coding gain decrease when the codeword length increases. The LCPC code offers lower encoding and decoding complexity as compared to LDPC codes. Unlike LDPC, LCPC codes do not require any iteration process during decoding.

REFERENCES

[1] H. Soude, M. Agueh, and J. Mehat, "Towards an optimal reed solomon codes selection for sensor networks: a study case using tmotesky," *in Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 165-166, 2009.

[2] N. A. Alrajeh, U. Marwat, B. Shams, and S. S. H. Shah, "Error correcting codes in wireless sensor networks: an energy perspective," *Applied Mathematics and Information Sciences*, vol. 9, no. 2, pp. 809-818, 2015.

[3] S. S. Alhajji, and S. A. Alabady, "Slotted ALOHA Based p-Persistent CSMA Energy-Efficient MAC Protocol for WSNs," *International Journal of Computing and Digital Systems*, vol. 9, no. 1, pp. 1-10, 2020.

[4] Salah. A. Alabady, and S. Raed, "MHUCR: Mutli hop uniform clustering routing protocol for energy efficient WSN," *International Journal of Grid and Distributed Computing*, vol. 11, no. 6, pp. 95-106, 2018.

[5] N. Tekin and V. C. Gungor, "Lifetime analysis of error control schemes on wireless sensor networks in industrial environments," *in Proceedings of the 27th Signal Processing and Communications Applications Conference* (SIU), pp. 1-4, 2019.
[6] S. L. Howard, C. Schlegel, and K. Iniewski, "Error Control Coding in Low-Power Wireless Sensor Networks: When Is ECC Energy-Efficient?," *EURASIP Journal on Wireless Communications and Networking*, vol. 2006, pp. 1-15, 2006.
[7] Zarei B, Muthukkumarasay V, Wu XW, "A residual error control scheme in single-hop wireless sensor networks," *In IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 197-204, 2013.
[8] D. Kheirandish, A. Safari, and Y. Kong, "A novel approach for improving error detection and correction in WSN," *In IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1-4, 2014.
[9] M. A. M. Albashier, A. Abdaziz and H. A. Ghani, "Performance analysis of physical layer security over different error correcting codes in wireless sensor networks," *20th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Bali, pp. 191-195, 2017.
[10] R. A. Carrasco and M. Johnston, Non-binary error control coding for wireless communication and data storage: Wiley Online Library, 2008.
[11] R. Anane, R. Bouallegue and K. Raoof, "Extensive Simulation Performance Evaluation of MSK Scheme with Error Correcting Codes in Wireless Sensor Networks," *IEEE 29th International Conference on Advanced Information Networking and Applications*, Gwangiu, pp. 400-405, 2015.
[12] C. Wei, P. Chen, Y. S. Han and P. K. Varshney, "Local Threshold Design for Target Localization Using Error Correcting Codes in Wireless Sensor Networks in the Presence of Byzantine Attacks," *In IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1571-1584, 2017.
[13] A. Vempaty, Y. S. Han and P. K. Varshney, "Target localization in Wireless Sensor Networks using error correcting codes in the presence of Byzantines," *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, pp. 5195-5199, 2013.
[14] N. Abughalieh, K. Steenhaut, and A. Nowe, "Low power channel coding for wireless sensor networks," *In 17th IEEE Symposium on Communications and Vehicular Technology in the Benelux*, (SCVT2010), pp. 1-5, 2010.
[15] A. Vempaty, Y. S. Han and P. K. Varshney, "Target Localization in Wireless Sensor Networks Using Error Correcting Codes," *IEEE Transactions on Information Theory*, vol. 60, no. 1, pp. 697-712, 2014.
[16] M. Luby, "LT codes," *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, Vancouver, BC, pp. 271-280, 2002.
[17] Salah A. Alabady, and F. Al-Turjman, "Low Complexity Parity Check Code for Futuristic Wireless Networks Applications," *IEEE Access*, vol. 6, pp. 18398-18407, 2018.
[18] Salah A. Alabady, M. F. M. Salleh, and F. Al-Turjman, "LCPC Error Correction Code for IoT Applications," *Sustainable Cities and Society*, vol. 42, pp. 663-673, 2018.
[19] S. J. Johnson, Iterative error correction: turbo, low-density parity-check and repeat-accumulate codes: Cambridge University Press, 2010.
[20] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Transactions on Information Theory*, vol. 47, pp. 2711-2736, 2001.

BIOGRAPHIES OF AUTHORS

Salah A. Alabady received the B.Sc. degree in Electronics and Communications Engineering and the M.Sc. degree in Computer Engineering from Mosul University, Mosul, Iraq, in 1996 and 2004 respectively, and the Ph.D. degree in Wireless Networks from School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Pulau Penang, Malaysia, in May 2014. He was a Lecturer with the Computer Engineering Department, Mosul University, Mosul, Iraq, from 1999 until October 2010. Currently, he is Assistant Professor with the Computer Engineering Department, Mosul University, Iraq. His research interests include error correction codes, wireless network coding, joint network - channel coding, and cross-layer in wireless sensor networks. Asst. Prof. Dr. Salah has participated in many scientific activities as a reviewer in many respectable publishers such as IEEE, IET, Elsevier, and Springer, and many others. Currently, he has 45 published papers. He has supervised 5 MSc, and 2 PhD students.