Extension of Error Correction Capability of Two Dimensional Single Error Correction - Double Error Detection Encoding Technique

T. R. Nikolić[1], G. S. Nikolić[1], M. K. Stojčev[1], G. S. Jovanović[1], B. D. Petrović[1]

Abstract – **Wireless Sensor Networks (WSNs) are energy constraint networks that require reliable data transfer at a low cost of energy. There are two basic approaches to recover erroneous packets, ARQ and FEC. FEC is used with aim to increase link reliability and reduce number of packet retransmission. Given that most bit errors during data transmission are single-bit or double-bit errors and multiple-bit errors are present but rare, extension of the conventional error correction capability of 2D-SEC-DED code has been involved and performance evaluation are derived.**

Keywords – **Wireless Sensor Networks, Forward Error Correcting Code, 2D SEC-DED Coding Scheme**

I. Introduction

Wireless Sensor Networks (WSNs) are composed of large number of spatially distributed, autonomous and battery-powered small devices, called Sensor Nodes (SNs). SNs have the ability to sense the physical environment, process the obtained information and communicate using the radio interfaces [1]. An SN in a WSN is typically equipped with transducer, radio transceiver, microcontroller unit, and power source (usually battery). The lifetime of a WSN depends on how efficiently the battery life of each SN is in use. The radio transceiver is very expensive in terms of energy consumption, while the other components in the SN data processing consume significantly less.

Errors in data transmitted between SNs appear as a consequence of noise, interference, signal fading, etc. To provide reliable data communication two techniques are used, one is Automatic Repeat reQuest (ARQ) and the other one is Forward Error Correction (FEC). In ARQ the sender node adds error detecting code, usually Cyclic Redundancy Checks (CRC) or parity check codes to the data. In FEC the source node encodes data using some error correcting code which add redundancy to the packet and lets the receiver to correct errors in data packet if present, thus making retransmission outdated, since the communication activity is the most power hungry when compared to sensing and data processing.

This paper is focused on block codes. Among the widely used block codes in WSNs are Hamming, single error correction - double error detection (SEC-DED), CRC, erasure codes. Hamming codes have very simple structure and high code rate, and therefore they are used in FEC schemes when errors are random and the error rate is low.

In this paper we propose extension of error correction capability of two dimensional SEC-DED encoding technique. The extension deals with software manipulations of the received packet. In this manner, all single- and double-errors within the data field of the received packet are corrected.

[1]T. R. Nikolić, G. S. Nikolić, M. K. Stojčev, G. S. Jovanović, and B. D. Petrović, are with University of Nis, Faculty of Electronic Engineering, Niš, Serbia, E-mail: tatjana.nikolic@elfak.ni.ac.rs

II. SEC-DED Codes

SEC-DED codes are constructed by extended Hamming codes with an extra parity bit. The Hamming distance of SEC-DED is 4, which allows the decoder to distinguish between single-bit and two-bit errors. Thus the decoder can detect and correct a single-error and at the same time detect (not correct) a double error. Our observations, which are in accordance with [2], show that most bit errors are single- or double-bit errors, while burst and multiple bit errors rarely occur. Thus, it is likely that an encoding scheme that corrects single and double-bit errors can reduce a significant portion of the errors.

During one complete period of the adopted protocol implemented in our WSN, data gathered from several sensor elements are stored into the microcontroller memory as a matrix of order $k = k_r \times k_c$ bits (Information Bits Field, IBF), where k_r corresponds to the number of sensor elements in SN (sensed data) and k_c to the number of bits per element (ADC resolution of sensed data), see Fig. 1.

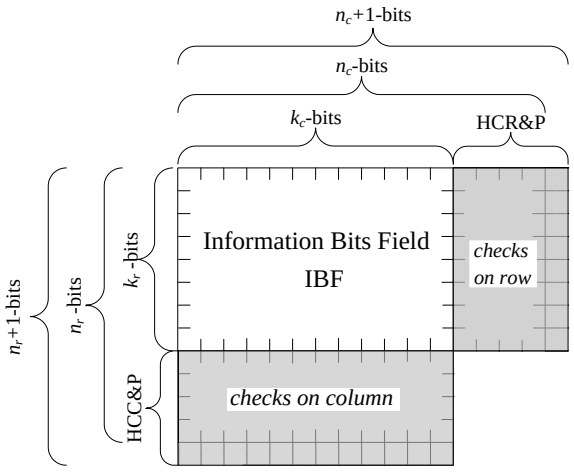

Fig. 1. 2D-SEC-DED code

2D SEC-DED code is two-dimensional code. We parameterize this code by the following expression

$$2D\ SEC - DED^{d_{min}}_{k_{ec}+h_{op},k_{er}+v_{op}}[k_r, k_c] \quad (1)$$

where d_{min} is the minimal Hamming distance, $k_{ec}(k_{er})$ is the number of parity check bits per row (column), $h_{op}(v_{op})$ is the number of overall parity check bits per horizontal (vertical) direction. For more details, related to creating 2D SEC-DED code, the reader can consult Reference [3].

In our solution we use synchronous rendezvous protocol called M-RPLL, which uses identical packet formatting and control mechanisms already described in [4-5], but implements different data encoding (2D-SEC-DED instead of CRC). The format of the control message (RTS, CTS, and ACK/NACK) for M-RPLL is presented in Fig. 2. If we put in expression (1) $k_r = 1$, $k_c = 11$, $k_{er} = 0$, $k_{ec} = 4$, $h_{op} = 1$, $v_{op} = 0$, and $d_{min} = 4$, the 2D SEC-DED code is transformed into one dimensional (16, 11) SEC-DED perfect code, see Fig. 1. Such encoding scheme employs single error correcting and double error detecting (SEC-DED) for the Data field. The format of the data message DATA is presented in Fig. 3. If we put in expression (1) $k_r = 11$, $k_c = 11$, $k_{er} = 4$, $k_{ec} = 4$, $h_{op} = 1$ and $v_{op} = 1$, we obtain (16, 11) SEC-DED perfect code per row and column, thus forming 2D-SEC-DED encoding scheme according to the procedure for generating a systematic code. In our design, 11 sensor elements are installed within the SN ($k_r = 11$), each sensed with 11-bit ADC resolution ($k_c = 11$).

III. Error Correction Capability of 2D SEC-DED

When conventional 2D SEC-DED error correction method is used [6], the wireless receiver, after accepting the control message, checks the SEC-DED code in two steps. In step 1, the single parity bit (overall parity) h_{op} is checked. If it is bad, the receiver assumes that 1-bit error occurred, and it uses the other parity bits (Hamming code bits, k_{ec}), in step 2, to correct the error, according to the Table I. It may happen, of course, that three or even five bits are bad, but the SEC-DED code cannot detect all such errors. If the h_{op} is good, then there are either no errors, or two bits are bad. The receiver switches to step 2, where it uses the other parity bits (Hamming code bits, k_{ec}), to distinguish between these two cases. Again, there could be four or six bad bits, but this code cannot handle this cases.

To correct single- and double-errors at low latency cost (by software), we extend detecting and correcting processes in the algorithm presented in [3] with additional two macro-steps (Macro-Step_4 and Macro-Step_5), see Fig. 4. Each macro-step performs already defined activities, but the number of iterations is increased.

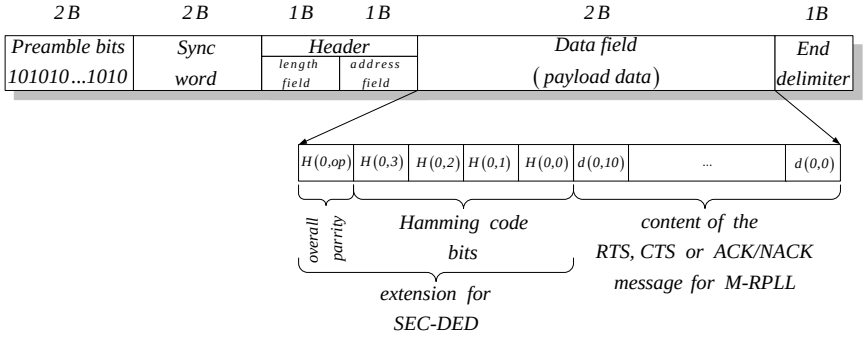

Fig. 2. Format of control messages for M-RPLL

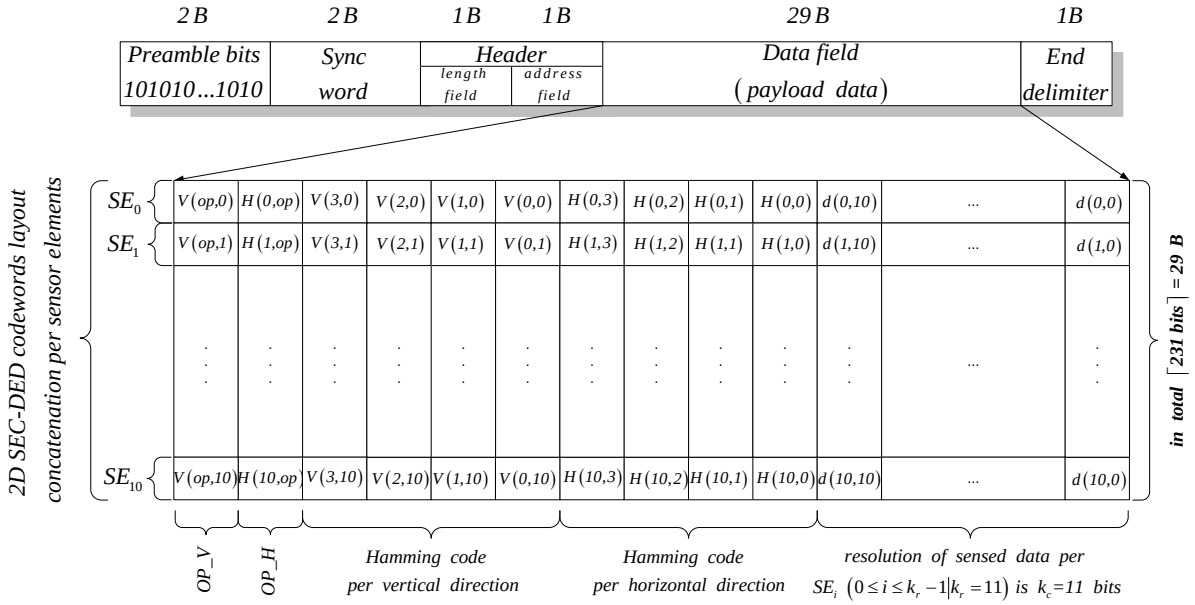

Fig. 3. Format of a data message for M-RPLL using systematic code

TABLE I
ERROR DETECTION FLAGS

Syndrome	Overall Parity	Error Type	Description
0	0	no error	No error
≠0	1	single error	Correctable (Syndrome point to incorrect bit position)
≠0	0	double error	Not correctable
0	1	parity error	Correctable (Parity error has occurred)

By using such procedure the obtained effects are the following: 1) Macro-Step_1 – error detection based on the syndrome and OP_H, and single bit error correction per horizontal direction; 2) Macro-Step_2 – error detection based on the syndrome and OP_V, and single bit error correction per vertical direction; 3) Macro-Step_3 – error detection based on the syndrome and OP_H, and single bit error correction per horizontal direction, again; 4) Macro-Step_4 – error detection based on the syndrome, OP_H and OP_V, and double bit error correction per elements at intersection points in matrix; and 5) Macro-Step_5 – error detection based on the syndrome and OP_H/OP_V, and single bit error correction per horizontal/vertical direction, again.

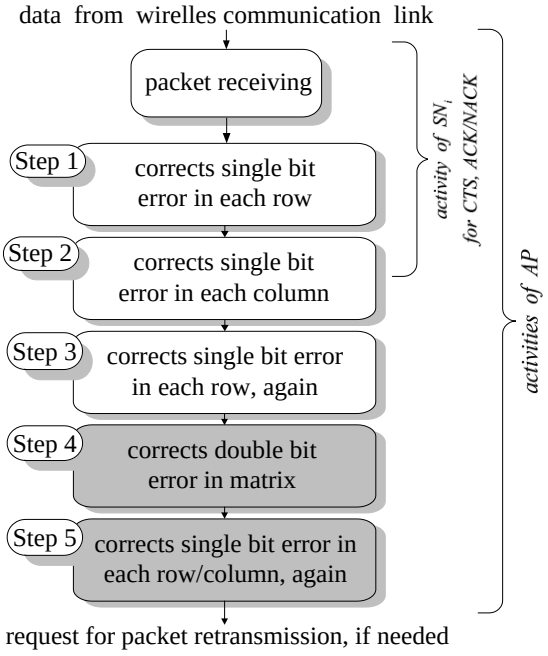

Fig. 4. Detecting and correcting processes by sensor node (SN_i) and access point (AP)

As can be seen from Fig. 4, the sender SN_i performs activity specified in Macro-Step_1, only (deals with error detection and correction of SEC-DED coded messages CTS and ACK/NACK). The AP executes Macro-Step_1 for RTS message, and activities from Macro-Step_1 up to Macro-Step_3 for 2D SEC-DED DATA coded message. Accordingly, after performing the three aforementioned steps, all single bit errors can be detected and corrected in any row or column. With the analysis carried out, all double errors are only detected but not corrected.

The patterns of possible double errors for a randomly selected data bit sequences, which cannot be corrected with the conventional algorithm (three Macro-Steps), are sketched in Fig. 5. Double bit errors in the IBF field can occur in any row or column, but for easier viewing, the pattern examples are reduced to matrix of order 4x4.

	pattern 1					pattern 2					pattern 3			
	1	2	3	4		1	2	3	4		1	2	3	4
1	x	x				x	x				x	x		
2	x	x				x	x	x			x	x		
3							x	x					x	x
4													x	x

Fig. 5. Possible positions of double errors

Our goal now deals with correction of all detected double bit errors. This is achieved by extending the previous algorithm, i.e. by introducing two additional steps. In Macro-Step_4, firstly the possible positions of all uncorrected multiple errors are determined. They are located in the intersection of those rows and columns of the matrix in which multiple errors are detected. In this way all the positions on which exists an error are detected, but besides them, some positions on which there are no real errors are marked. After that, all bits whose positions were detected as erroneous were inverted. With this bit inversion in any row or column, two real errors are corrected, but a new single error is involved. In accordance with the fact that double errors have a much higher likelihood than triple, we assume that in any row or column there can be two errors, while triple errors are very rare. So we can assume the error that was injected in the Macro-Step_4, of the proposed algorithm, is the third error in the order. Bearing in mind that some correct data bits have been inverted in this step, the values of these bits must be restored to old correct values. Therefore, to remedy this fault, we introduce Macro-Step_5, as additional. The conducted processing of this step is equivalent to Macro-Step_1 or Macro-Step_2 because the newly injected single bit errors can be corrected per horizontal or vertical direction.

IV. SIMULATION RESULTS

For data encoding, injecting, detecting and correcting errors during data transmission, a software simulation model, created in MATLAB, was used. Error-free, single- and double-/multiple- errors within a data field where injected by the MATLAB *randerr* function by using Binomial probability model for BER (bit error rate) as parameter. Let note that the most appropriate probability model of bit error from data/computer communication and networking point of view is the Binomial probability model [7]. In essence the Binomial function deals with integer valued discrete function and therefore it is appropriate model of bit error. Probability $P(i,m)$ that there are i errors in a code or message of size m ($i \leq m$) is given by a binomial frequency function, defined as:

$$P(i,m) = C_i^m \alpha^i \cdot (1-\alpha)^{m-1} \quad (2)$$

where $\alpha = BER$, $C_i^m = \binom{m}{i} = \frac{m!}{i!(m-i)!}$, $i \in \{0,1,2\}$, $i = 0$ corresponds to error-free transfer, $i = 1$ to single bit error transfer, and $i = 2$ to double-/multiple-bit error transfer. The *randerr* function [8] generates binary matrices with a specified number of zeros and ones and is a meant for testing error-control coding. The command $= randerr(N, M, [0,1,2; P_0, P_1, P_2])$, generates an $N \times M$ binary matrix having the property that each row (see Fig. 3) contains zero $'1's$ with probability P_0, one $'1'$ with probability P_1, etc. The error probability model, derived using Eq. (2), is presented in Table II. As can be seen from Table II most bit errors are single-bit or double bit errors and burst/multiple errors are present but rare. Thus, it is likely that an encoding scheme that corrects single and double-bit error can reduce a significant portion of the errors.

TABLE II
ERROR PROBABILITY MODEL

BER	Probability P_i per row [%]		
	P_0	P_1	P_2
$\alpha = 10^{-3}$	81.131	16.973	1.896
$\alpha = 10^{-4}$	97.931	2.046	0.023
$\alpha = 10^{-5}$	99.791	0.208	0.001
$\alpha = 10^{-6}$	99.979	0.020	~0.001

Notice: P_0 - error-free, P_1 - single-bit error, P_2 - double-/multiple-bits error

TABLE III
PACKET DELIVERY RATIO FOR M-RPLL PROTOCOLS

	BER				
	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
N	262800				
P_SE	262642	213288	43063	4745	434
P_D/M	259279	31081	345	4	0
PDR [%]	0.07	99.75	99.99	99.9999	~100

Notice: N - total number of transmitted packets, P_SE - # of packets with single error, P_D/M - # of packets with double-/multiple-errors

By conducting the error recovery procedure defined in Fig. 4, and the number of injected errors defined by random function (Table II), we obtain the results given in Table III. By analyzing the results, we can conclude the following:

1) For performance estimation we used a metric Packet Delivery Ratio (PDR) defined as coarse-grain indicator point to the ratio of the correctly received packets at the receiver to the total number of packets sent by the sender.

2) For BER=10^{-4}, 100% of a single-bit error and 99.99% of double-/multiple-bit errors are corrected. Let note that, from fault tolerant point of view, the involvement 2D SEC-DED/SEC-DED encoding technique is equivalent to implementation of Triple Modular Redundant (TMR) scheme at word-level voting, i.e. instead of using three packet replicas we use only one.

3) The proposed 2D SEC-DED implementation is effective when the BER (<10^{-2}) is not high and most errors are single-bit. When most of the error are bursty the proposed scheme is not as effective in reducing packet losses as is the case when erasure encoding is implemented, but at cost of higher energy consumption and computationally complexity.

V. CONCLUSION

During data transfer in WSN errors appear. In order to solve this problem in an efficient manner FEC technique is used. One of the most common FEC techniques is 2D-SEC-DED. Its main drawback is inability to correct double errors in information bits of the packet. In this paper, we propose an extension of the conventional 2D-SEC-DED procedure. The obtained simulation results show that all double errors in information field can be corrected. Having in mind that communication activity is the most power hungry one, by using 2D-SEC-DED code we drastically decrease the number of packet retransmissions and thus prolong the lifetime of the sensor node.

ACKNOWLEDGEMENT

This work was supported by the Serbian Ministry of Education and Science, Project No TR-32009 – "Low power reconfigurable fault-tolerant platforms".

REFERENCES

[1] I. F. Akyildiz, M. C. Vuran, Wireless Sensor Networks, Chester UK, John Wiley & Sons Ltd; 2010

[2] F. Koushanfar, et all, Fault Tolerance in Wireless Sensor Networks, in: Handbook of Sensor Networks, by I. Mahgoub and M. Ilyas (eds.), Section VIII, Ch. 36, pp. 36.1-13, CRC Press, Boca Raton, 2004

[3] Goran S. Nikolic, et all, "Reliable data transfer Rendezvous protocol in wireless sensor networks using 2D-SEC-DED encoding technique", Microelectronics Reliability, Volume 65, October 2016, pp 289-309

[4] M. R. Kosanović, M. K. Stojčev, "RPLL - Rendezvous Protocol for Long-Living Sensor Node", Facta Universitatis Series: Electronics and Energetics, Vol. 28, No. 1, (2015), pp. 85-102

[5] M. Brzozowski, K. Piotrowski, P. Langendoerfer, "A Cross-Layer Approach for Data Replication and Gathering in Decentralized Long-Living Wireless Sensor Networks", ISADS 2009, In Proc. of the 9th ISADS, 2009. pp. 49-54

[6] S. Lin, D. J. Costello, Error Control Coding, 2nd ed. Upper Saddle River, New Jersey: Pearson Prentice Hall; 2004.

[7] C. T. Bhunia, Information Technology Network and Internet, New Age International Publisher, New Delhi, (2005), pp. 173

[8] Mathworks, *randerr*, Generate bit error patterns, at: http://www.mathworks.com/help/comm/ref/randerr.html?requestedDomain=www.mathworks.com, (accessed 2018)