International Journal of Engineering in Computer Science

E-ISSN: 2663-3590
P-ISSN: 2663-3582
www.computersciencejournals.com/ijecs
IJECS 202; 3(1): 31-34
Received: 11-03-2021
Accepted: 02-04-2021

Farah Ali Tahseen
Department of Computer Engineering, College of Engineering, University of Mosul, Iraq

Salah Abdulghani Alabady
Department of Computer Engineering, College of Engineering, University of Mosul, Iraq

Sukaina Shukur Alhajji
Department of Computer Engineering, College of Engineering, University of Mosul, Iraq

Corresponding Author:
Salah Abdulghani Alabady
Department of Computer Engineering, College of Engineering, University of Mosul, Iraq

Error detection and correction implementation for computer networks using Arduino

Farah Ali Tahseen, Salah Abdulghani Alabady and Sukaina Shukur Alhajji

DOI: https://doi.org/10.33545/26633582.2021.v3.i1a.44

Abstract
Error detection and correction, in particular in high-data-rate and high-reliability wireless network applications, has received significant concentration. Error detection and correction are techniques that enable secure transmission of digital data through unreliable communications networks in the information and coding theory. The use of error detection and correction techniques leads to an increase in the performance of the network by reducing the end-to-end delay, reducing the bit error rate, and thus increasing the throughput. Since OSI and TCP/IP models have no bits corrections operation, and the only error detection operation is performed in the data link layer and the transport layer using Cyclic Redundancy Check (CRC). We suggest adding a sub-layer between the physical layer and the data link layer responsible for detecting and correcting errors in this paper. The hardware design of the proposed sub-layer using the Hamming code is implemented using Arduino. As a result, without having to demand retransmission from the sender, we will have a network device that can detect and correct a bit of error in the frames.

Keywords: Wireless networks, error correction, error detection, forward error correction, hamming code, Arduino, system on chip (SoC)

Introduction
Generally, any real wireless system is riddled by errors in data transmission from time to time and from several various sources such as channel fading, interference, random noise, etc. Without the use of error control methods, accurate data transfers will be impossible. The Forward Error Correction (FEC) can detect and correct the errors, thus it is possible to avoid data retransmission [1].
The common feature about the communication channel is that the data is passed through the channel from the source to the recipient. The channel may be crowded in the event that whatever it received may not be the same as what has already been sent. Consequently, if binary data is sent through the channel when 0 is sent, it is received hopefully as 0, but often it is received as 1 (or as unacknowledged). The basic issue with coding theory is deciding which message has been sent based on what has been received [2]. The error detection and correction codes must have little complexity and smaller memory. A serious challenge in the Internet of Things (IoT) and Wireless Sensor Networks (WSNs) is energy-saving and energy usage. Avoiding or decreasing the amount of sending the error packets is really important. One of the strategies that used to decrease energy consumption is utilizing forward error-correcting methods like hamming code.
Over the past decade, several error detection and correction code schemes such as (Reed-Solomon code, Turbo codes, Bose-Chaudhuri-Hochquenghe (BCH) code, and low-density parity-check (LDPC) codes) were researched to ensure the robustness of the wireless networks, in addition to fulfilling the reliability of the data in a high data rate network. Every one of the developed codes does have its own advantage to be used as the channel coding scheme in a communication network [3].
The remaining part of the paper is coordinated as follows: the related work is given in section II. Hardware implementation is presented in section III. Section IV discusses the results. Eventually, section V concludes the paper.

Related work
The important role of error detection and correction has led researchers to develop a large

number of techniques in different areas. Sindhuaja Muppalla, and *et al*. present a novel Very Large Scale Integration (VLSI) implementation of Universal Asynchronous Receiver Transmitter (UART) designed to involve (8, 4) expanded hamming code known as Single Error Correcting Double Error Detecting (SEC-DED) code which can detect two errors and correct only one error. This design can increase the system's noise immunity to maximize data reception without errors. The entire architecture is implemented in the Xilinx ISE 12.3 simulator designed for the Xilinx Spartan 6 FPGA [4].

The authors in [5] proposed a code for error detection and error correction known as hamming code. This code can reveal errors in the same way that conventional codes do while also correcting them. The architecture of the hamming code and its process are described in [5]. The synthesis is performed employing Xilinx ISE 14.7i, then the simulation is performed using Questasim and the VERILOG language is used to develop HDL. Mohammed Abdul Kader, and *et al*. built up a wireless data acquisition scheme that can gather data using error detection and correction code from a wireless sensor array. The system transmitting gathers information from different sensors and sends it using linear block code. The receiving device is given the code word passed through the sender circuit, the error tested, and the error bits corrected if an error occurs and the data is decoded. Finally, the data is stored on the SD card. The system is being checked for data gathering from the sensor array [6].

Carry-Select Adder (CSA) and Pre-Computed Carry-Select Adder (PCSA) error detection modules were presented by Karthik Kumar and P Sathish Kumar with enhanced delay. These modules improve the overall system performance by decreasing the propagation delay. The characteristics of the suggested approach are integrated into the specification for detecting and correcting the stuck-at-faults and transient. The performance of the system is verified using Verilog HDL. The suggested architectures are implemented and synthesized in CADENCE [45 nm technology] for Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array FPGA (Virtex-6 family) [7].

N Karostiani and A B Pantjawati in [8] designing a (7, 4) Hamming code encoder trainer. The trainer was developed with various matrix generators to create code words in the channel encoder. The trainer is represented in two different stages: the manufacturing stage of hardware and Proteus 8 Professional stage of simulation. The Arduino Mega 2560 is utilized to design the trainer. A similar study was carried out with a Very High integrated circuit hardware Description Language (VHDL) in [9].

This paper is concerned with attaching a sub-layer between the physical layer and the data link layer in order to make a network device capable of detecting and correcting a bit of error in the frames.

Hardware implementation

Generally, the OSI and TCP/IP models lack error correction mechanisms as they can only detect errors that occur in the data link, the network, and the transport layers as explained in Figure (1). So, when an error is detected, the data must be transmitted again. In order to address this problem, we proposed a System on Chip (SoC) device that could detect and correct an error at the same time by adding a sub-layer between the physical layer and the data link layer to detect and correct an error known as Error Detection and Correction (EDC) sub- layer as shown in the Figure (2).

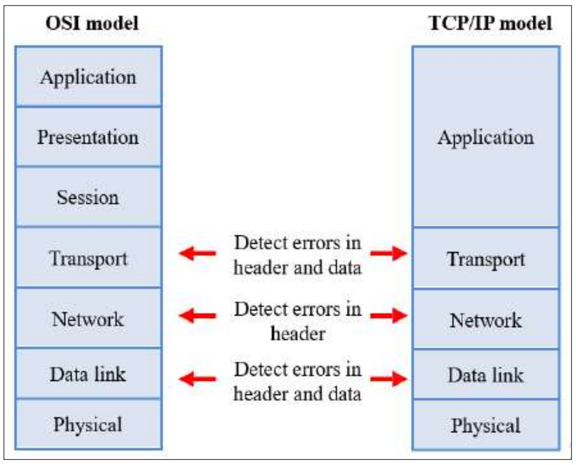

Fig 1: Presence of error detection mechanisms in OSI and TCP/IP models

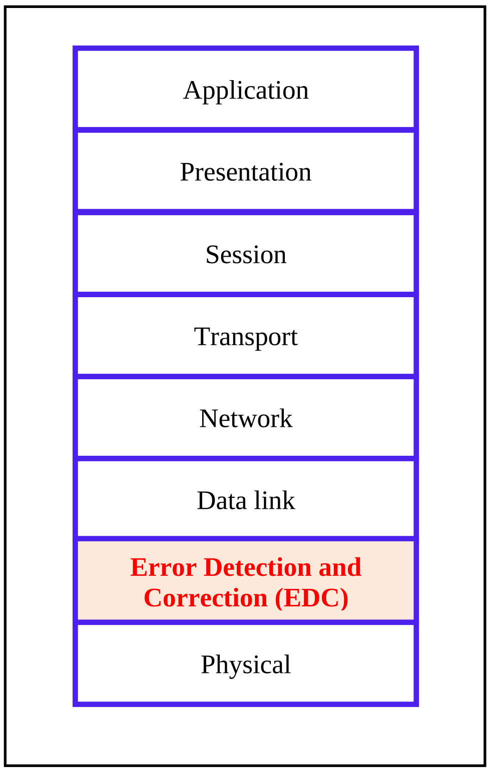

Fig 2: The error detection and correction (EDC) sub-layer

The error detection and correction (EDC) sub-layer is implemented by applying hamming code (4, 7) to Arduino. In order to build the system on a chip device, the following hardware components are required:

- Breadboard.
- Arduino Atmega or Uno.
- LEDs.
- Resisters 330 Ω.
- Dip switch 4 pins to generate the message.
- Dip switch 7 pins to add noise.
- Jumper wires.
- Arduino Integrated Development Environment (IDE) program.

The hamming code equations used are

Encoded message

$C = M.G$ (1)

Message received with noise

$$R = C + E \quad (2)$$

Syndrome vector to detect the error in message

$$Syn = R.\ H^T \quad (3)$$

Correct the error

$$Cc = R + E_P \quad (4)$$

Decoded the message transmitted

$$M = AND\ (1111000, Cc) \quad (5)$$

Where
M: Message.
G: Hamming code matrix (for encoding).
E: error (noise).
H: Parity Check Matrix Hamming code (for decoding).
E_P: error pattern.

$$G=\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$H=\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

The hardware components are connected as shown in Figure (3).

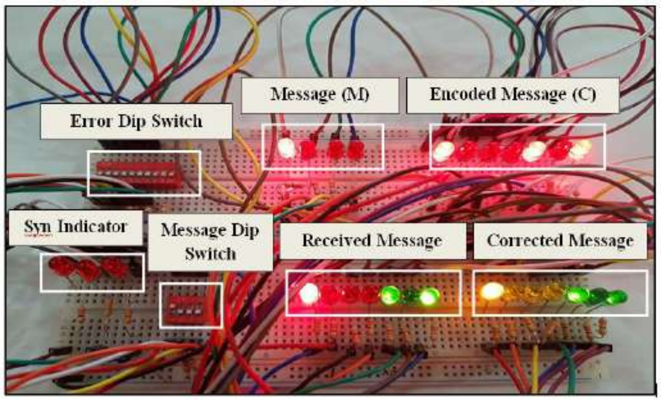

Fig 3: The hardware implementation

Results and Discussion

Three different scenarios have been taken to ensure that the designed system on the chip operates accurately and correctly. Since the message consists of four bits, there are sixteen possibilities for the data message. Table (1) shows the sixteen probabilities of the data message and the values of the encoded messages (*C*). While the possibilities for error (error pattern) and Syndrome vector (*Syn)* values are shown in Table (2).

Table 1: Message and C value

No	Message	C = M.G
0	0000	0000000
1	1000	1000101
2	0100	0100111
3	1100	1100010
4	0010	0010110
5	1010	1010011
6	0110	0110001
7	1110	1110100
8	0001	0001011
9	1001	1001110
10	0101	0101100
11	1101	1101001
12	0011	0011101
13	1011	1011000
14	0111	0111010
15	1111	1111111

Table 2: Error and SYN value

Number	Ep	SYN
0	0000000	000
1	1000000	100
2	0100000	010
3	0010000	001
4	0001000	110
5	0000100	011
6	0000010	111
7	0000001	101

First scenario

In this scenario, the message generated by the dip switch was (1000), as shown in Figure (4). The message was encoded using the hamming code to generate the encoded message *C* (1000101). We were supposed to send a message without any error, so the error dip switch was set (0000000). At the receiver side, we noticed that the received message (1000101) was the same as the transmitted message and the *Syn* indicator was (000) which means that there was no error and the received data was correct.

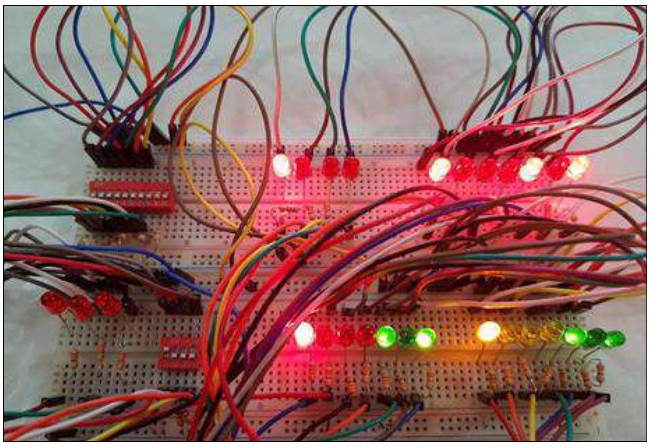

Fig 4: Data message 1000 with no error (First scenario)

Second scenario

In this scenario, the generated message was (1001), so the encoded message is *C* (1001110). We're supposed to have an error in the sixth bit so we set the *Ep* to (0000010). On the receiver side, we noticed that the received message (1101110) was not the same as the transmitted message and the *Syn* indicator was (111), which means that there was an error in the sixth bit as shown in Table (2) and the received data was incorrect. Thus, in order to correct the data, the designed chip corrected the error using Equation (5) to obtain the correct data (1001110) as shown in Figure (5).

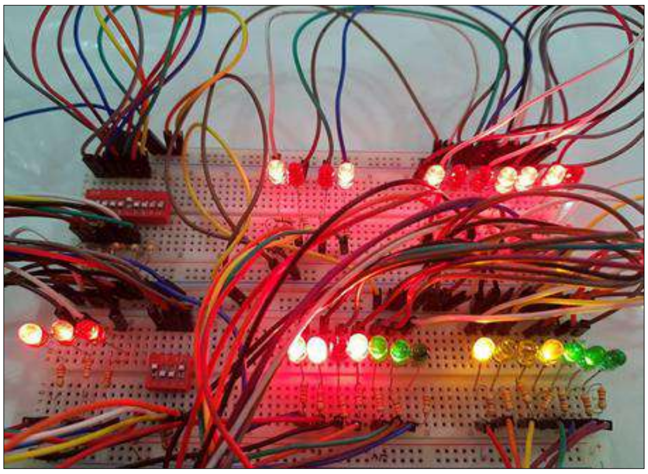

Fig 5: Data message 1001 with error in the sixth bit (Second scenario)

Third scenario

The generated message was (0010) in this scenario, so the encoded message is *C* (0010110). We're supposed to have an error in the fifth bit so we set the *Ep* to (0000100). On the receiver side, we noticed that the received message (0000110) was not the same as the transmitted message and the *Syn* indicator was (011), which means that there was an error in the fifth bit as shown in Table (2) and the received data was incorrect. Thus, in order to correct the data, the designed chip corrected the error using Equation (5) to obtain the correct data (0010110) as shown in Figure (6).

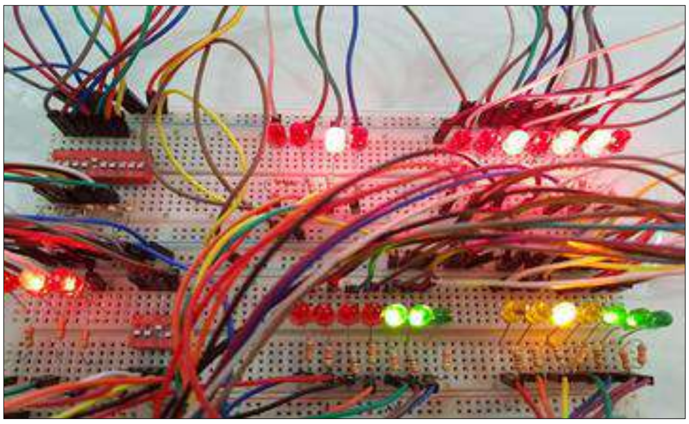

Fig 6: Data message 0010 with error in the fifth bit (Third scenario)

Table (3) summarized the three different scenarios

Table 3: Results summary

Scenarios	No	Message	Syn	Ep	C = M.G	R = C + Ep	*Cc* = R + Ep
1st	8	1000	000	0000000	1000101	1000101	1000101
2nd	9	1001	111	0000010	1001110	1001100	1001110
3rd	2	0010	011	0000100	0010110	0000110	0010110

Conclusion

The importance of transmitting data without error or with corrected errors without the need to retransmit incorrect data to energy-saving, reducing delays, increasing throughput, and improving network performance led to propose many error detection and correction mechanisms. The main issue with which the OSI and TCP/IP models have suffered is that there are no error correction techniques; they only have error detection techniques. To solve this problem, we designed a system on chip (SoC) device by adding an error detection and correction (EDC) sub-layer between the physical layer and the data link layer capable of detecting the error and correcting it. The device has been implemented using Arduino Atmega. The device was verified using different scenarios and the results showed the accuracy of the proposed device.

References

1. Salbiyono A, Adiono T. LDPC decoder performance under different number of iterations in mobile WiMax, presented at the International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS) 2010, P1-6.
2. Moon TK. Error correction coding, Wiley Online Library 2005.
3. Alabady SA, Al-Turjman F. Low Complexity Parity Check Code for Futuristic Wireless Networks Applications, IEEE Access 2018;6:18398-18407.
4. Muppalla S, Vaddempudi KR. A Novel VHDL Implementation of UART with Single Error Correction and Double Error Detection Capability, In 2015 International Conference on Signal Processing and Communication Engineering Systems, IEEE 2015, P152-156.
5. Raghavaiah B, Omprakash. Implementation of Hamming coding in Residue Number System, In 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), IEEE 2018, P1-5.
6. Kader MA, Uddin MM, Rahaman MA, Islam N, Razak MA. Data Transmission via Wireless Channel to Store in a Remote Device Employing Error Detection and Correction Code, In 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), IEEE 2019, P1-6.
7. Kumar AK, Kumar PS. High Speed Error-Detection and Correction Architectures for Viterbi Algorithm Implementation, In 2019 3rd International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), IEEE 2019, P1-6.
8. Karostiani N, Pantjawati AB. Design and Realization of (7, 4) Hamming Code Channel Encoder Trainer Using Arduino Mega 2560, In IOP Conference Series: Materials Science and Engineering 2020;850(1):12033-12042.
9. Sani US, Shahono IH. Design of (7, 4) Hamming Encoder and Decoder Using VHDL 1st International Engineering Conference 2015, P103–106.