Tele-Communication System Design

CS 252 Lab03

Kanishk Garg
(210050080)

Yash Ruhatiya
(210050169)

Pulkit Goyal
(210050126)

Priyanshu Yadav
(210050125)

January 29, 2023

Contents

1 Design

1.1 Physical Layer Encoding

In physical layer encoding, we encode the bits 0 and 1 with two different sounds, with a small time gap between two consecutive bits. We will keep the sounds as distinct as possible to avoid bit errors.

1.2 Link Layer Framing

Initially, we are using the start flag representing 5 bits and another 5 bits to encode the message length. Then we add the payload (main message). At last, row parity and the column parity of 5 bits each are added (Refer to Error Correction section for futher clarification).

Frame Structure

Start flag (11111)	Message Length	Message body	Row parity	Column parity
	5 bits	0-20 bits	5 bits	5 bits

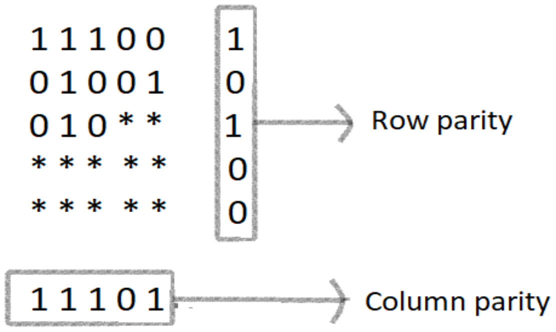
1.3 Link Layer Reliability

We ensure reliability over the network by sending a 2-bit acknowledgement to the sender upon performing error detection on the message received, as described in the section below. For re-transmission, we encode the ack as follows:

- 00 - Both messages have been received successfully
- 01 - Re-transmit the first message
- 10 - Re-transmit the second message
- 11 - Re-transmit both messages

1.4 Error Correction

We use 2d-parity for error correction, as we are guaranteed at most one-bit error. The part of calculating row and column parity for a message has been automated through python code. The sizes of row and column parity have been fixed to 5, each row will have five entries, and as the total number of message bits is less than 20, there would be at most five rows.



2d-parity for message 1110001001010

We use the row and column parities (by row-parity, we denote the xor of bits in a row, and similarly for a column) on the receiver side to detect errors. If the error is at most 1 bit, it can be corrected, and this task is automated. But if the number of errors exceeds 2, the receiving side has made a mistake in listing down the message, so we ask for re-transmission.

2 Implementation

2.1 Sender's side

The code written has a function encode, which on being given the correct message to be sent, calculates the message length in binary and the row and column parities and then appends them to form a frame whose design has been given in section 1. We transmit both frames with no gap in between to be as efficient as possible.

We have used python code for message transmission. Once given the message to be sent, the code generates sounds with a short rest time between consecutive bits.

2.2 Receiver's side

Both receivers listen to the sound on the receiving side and manually note down the message to cross-verify. Depending on whether the messages match or not, we either start processing the message or ask for re-transmission.

In the case of processing, we give the received messages as a single input to the decode function, which then splits it in two using the message length and start and end flags. After this step, the function extracts the message body, column and row parities for each message. With the extracted body, the function again calculates the parities and based on the mismatch, it outputs the erroneous rows and columns.

If the number of bit errors is 1, then the function flips the bit at that position. But if the number of bit errors is more than 1, it means that the receiving side made a mistake and hence asks for re-transmission.

In the case of re-transmission, the above process repeats, the only difference being that the sender side is not introducing any error this time.