1 Physical Layer Encoding and Decoding

We are planning to convert the binary into another encoding system developed by us. For every 3 bit patterns we assigned a unique symbol. Since there are 8 such possible patterns 8 unique symbols are required. Now for just 1 length bits 0 and 1 we assigned two more unique symbols. So, In total we are using 10 different unique symbols. This reduces the number of symbols transferred from \( n \) to \( n/3 \). We had written a program that given a \( n \)-bit array it returns after the corresponding string of symbols in our system to be transferred. There is also another program that decodes the string into a binary string. The first program is present at sender side and the second program is present at receiver side. They run the corresponding programs in-order to encode and decode respectively.

Now since it reduces the number of symbols to be transmitted it helps in increasing the throughput. We are planning to use cards to represent the 10 different symbols and use them to send message to receiver so that he can decode them and convert into binary.

We are using the visual system as it is the most efficient and less error prone.

2 Link layer framing

We are again making use of cards in-order to identify the start and end of the frame. In-order to represent the start we lift all the 10 cards with two hands and also to inform receiver that the sending of message has ended we again lift all the cards.

We need not again add flags to distinguish the original message bits from the redundancy bits as in our design technique the number of data bits is having a one-to-one mapping to total message (data+redundancy). So, given the received message, from this mapping we can get the number of data bits and we can the first data bits from the received message.

3 Link Layer Reliability

In case we are sending the message but the receiver might not had received some symbol or he had not understood what we had sent or shown. In this case we do a similar operation as that of sending a NACK packet. He shows the cross hands signal saying us that there is something wrong on his part.
and asks us to resend the symbol. Once he shows the cross symbol we will again retransmit the entire packet. This makes sure that we are transmitting the message reliably.

4 Error Detection

In order to detect messages we are making use of two-dimensional parity check but in a modified way. We first divide the n-bit message into \( \lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil \) a table. We add some padding bits as 0s so as to make them a nearest perfect square. We will just add this as a part of our algorithm but not during the transmission. We know that a two dimensional parity can detect the presence of errors(one bit or two bit) and can correct any single bit errors. But, we are making a modification to this two dimensional parity check. Instead of checking the horizontal rows and vertical columns parity we check the diagonals from left to right, diagonals from right to left, and horizontal rows parity and add redundancy bits accordingly. With this we can detect and correct both single and double bit errors. More details can about this can be made clear with the figure-1 inserted in this report.

5 Error Correction Algorithm

We had written two algorithms one adds the redundancy bits to the original message. This algorithm is executed by sender before sending the message. The second algorithm is typically run at the receiver side. This algorithm first calculates the parity bits obtained for the message received. It compares the parity bits obtained with the received parity or redundant bits. If there are no mismatches at all then this implies that there are no errors, so
it just returns the same message received as final message.
If there are two mismatches one in diagonal parity-1 and another in diagonal parity-2 this implies that there is just one bit error and we can get its position by simple mathematical analysis. So, we can correct that one bit error and return the corrected message.
If there are 4 mismatches two either in diagonal parity-1 or diagonal parity-2 and another two in horizontal parity then this implies that there are two errors. we had written two functions that does some mathematical analysis and detects the positions of errors, correct them and return the correct message.
If there are 4 mismatches two in diagonal parity-1 and another two in diagonal parity-2 then this implies that there are two errors. we had written a function that does some mathematical analysis and detects the positions of errors, correct them and return the correct message.
There wont arise any case where there occurs three mismatches or one mismatch in this parity check algorithm devised by us. Thus we are taking care of all cases and can definitely corrects all one and two bit errors reliably.

6 Conclusion

Hence, with our design we can guarantee reliability and it can definitely correct all one and two bit errors not only for data size of 20 but for any data size. For a given \( n \), the size of data, the number of redundant bits that our design adds is \( 5\sqrt{n} - 2 \). Thus for a high value of \( n \) we are optimising the redundancy bits to \( O(\sqrt{n}) \). And from this method i.e; by adding \( 5\sqrt{n} - 2 \) number of redundancy bits we can definitely say that our algorithm corrects all 1 bit and 2 bit errors not only for \( n \leq 20 \) but \( \forall n \in N \).