160050005

160050008

160050028

160050043

Report

February 12, 2018

Physical Layer Encoding

- The message is being transmitted at the physical layer using light (colour) signals.
- Two colours have been used to encode bits, another one to signify the ending of a bit and another colour to signify the end of a packet.
- The transmitted message is recorded manually on the receiver side which is then fed to a computer to check for its validity.
- The receiver will send an ACK or a NACK message back to the sender, which are also encoded using colours. The following colour scheme has been used for encoding different parts of the message-
    - Sender to receiver
        - color1 : Red : bit 0
        - color2 : Green : bit 1
        - color3 : Yellow : intermediate for marking end of bit signal
        - color4 : Light Blue : for marking the start and end of packet
    - Receiver to sender
        - color1: Green (for 1 second) : ACK
        - color2: Red (for 1 second) : NACK

Link Layer Framing

1. The end of a frame is encoded by the colour blue.
2. Any message coming after that is a part of the next packet.
3. Each packet has the relevant error correction bits (16 in this case) just before the end of it.

Link Layer Reliability

- We are making the link layer transmission reliable by using an error correction scheme.
- We are using an error correction scheme though it involves transmission of a higher number of bits as the probability of there being an error in the transmission is quite high.
  - Here we are calculating the parity of every **4th bit, every 5th bit and every 7th bit**.
  - eg. Say we have a 15 bit message. We are using indexing from 0.
  - For this message, the parity for all bits having the same value mod 4 is considered. Eg. This will involve addition of an extra 4 bits. We do a similar process for numbers 5 and 7.
  - Now if there is an error in the 3nd and 9th bit (say). This error will reflect in the 3rd and 1st bit respectively corresponding to number 4, 3rd and 4th bit respectively corresponding to number 5, 3rd and and 2nd bit respectively corresponding to number 7.
  - We formed different cases and for each error involving almost two bits it is possible to deduce in which two (or one or zero) bits there are errors (using **Chinese Remainder Theorem**). Therefore, a total of 4+5+7 = 16 extra bits were required. We chose these numbers as the lcm of any two numbers among them is >=20 and their sum is least possible.
- The messages can be differentiated into different types as-
  - 0 errors - All the correction bits match with the input and thus it is correctly identified as valid
  - 1 bit error- For a 1 bit error, a change will be reflected in 3 bits in the correction scheme (one each corresponding to mod4, mod5 and mod7)
  - 2 bit errors- They can be divided into two types according if the error in two bits is same mod4,5 or 7. Note that both cannot be same modulo two of those numbers

due to the properties of the chosen number. In this case there will be an error in 4 bits.If they are not the same, for every bit 3 different correction bits will change and there will be a change in a total of 6 bits.

- Inspite of this, if there is an error in recording, a NACK is sent after which the message is retransmitted.
- Corner Cases:-
  - If there is an error in the correction bits (atmost two errors are allowed in the entire message), then the scheme will recognize that there is an error and will send a NACK, asking for retransmission as in our scheme there can be a change in only 3,4 or 6 bits.
  - If there in an error in 1 bit in the message and 1 bit in the correction scheme, then this case never happens previously and once again the message will be declared invalid.
- In general, if it is known that the input in at max. N bit long, the 3 numbers we choose can be around square_root(N). Therefore, using an extra O(3*square_root(N) bits), two bit errors can be corrected using this scheme.

Implementation

1. Sender

   The extra bits are calculated and appended on the sender side using a python program. Once the input is generated , a python program using graphics.py library generates a message according to the colour scheme.

2. Receiver

   On the receiver side, the recorded message is the given to a python program which checks its validity and corrects it if possible after which it generates a ACK or NACK (using a python library) according to the input. The option of sending an ACK or a NACK is given to the user as well.

3. Tester

Moreover, to test the validity of our error correction scheme, we ran a tester that generates all inputs and creates all two-bit errors and ran it on our error correction scheme. It worked alright!!!