CS378: Computer Networks Lab
Hands-On PHY Layer : Design Document

22B1030 : Kajjayam Varun Guptha
22B1008 : Rithik Nuthakki
22B0948 : Rushikesh Rathamasetty
22B1046 : Neredimelli Sandeep

Introduction

In this lab, we implement a physical (PHY) network layer using audio systems to transmit and receive messages encoded as bit sequences. The goal is to reliably transfer bitstrings from the sender to the receiver through sound and correcting bit errors at the receiver end.

System Overview

Our system consists of two main components:

- **Sender Program**: Takes a Bitstring and position of bit errors as input, adds preamble to the input bitstring, encodes the Bitstring into a codeword, and flips the bits at mentioned positions.
- **Receiver Program**: Decodes the received audio signal, detects and corrects errors, and outputs the corrected bitstring.

Encoding Scheme

We have used an Extension of Hamming Code for transmitting the Bitstring, This technique ensures the correction of 1-bit errors and 2-bit errors in the code word

Hamming code

- Hamming Code is simply the use of extra parity bits to allow the identification of an error.
- All the bit positions with a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
- All the other bit positions are marked as data bits.
- Each data bit is included in a unique set of parity bits, as determined by its bit positions in binary form i.e. parity bit 2^i covers all the bit positions greater than 2^i whose binary representation includes a 1 at ith position from the right.
- We will check for even parity and set a parity bit to 1 and if the total number of ones in the positions it checks is odd, set a parity bit to 0 if the total number of ones in the positions it checks is even.

Encoding Process:

We will define the input string as an 'a', string formed by applying hamming code to the input string as 'b' and extra bit as 'p' and 'n' as the length of the input bitstring.

- We will add a preamble of 5 bits which represents n, this preamble sequence at the beginning of each transmission helps to find the actual length of the message.

- p will be set if the number of set bits(bits which are set) in b is odd.
- Encoded string can be formed by appending the preamble with a, b, and p.

Error Detection and Correction

We will implement the **Extended Hamming Code** for error detection and correction to handle potential bit errors during transmission.

Decoding Code:

- According to the above encoding we can say that the number of bits set in the p+b string is even, So we will check the number of bits set in the p+b string in the received string if it is even we can say that there is a two-bit error or zero-bit error else will have a one-bit error in p+b string.
- If there is a one-bit error, and apply hamming code to b again, we will get another set of parity bits, apply XOR to both these parity bits, and the applied number will be the index of the flipped bit in b, we can flip that bit and extract data bits in b and we can return it.
- If there is a two-bit error, There are two possibilities
  - * If there is a two-bit error in a and no-bit error in b+p in this case XOR of two sets of parity bits(as mentioned above) will be zero then we can extract the data bits in b and return it.
  - * If there is a two-bit error in b+p we can say that the XOR of two parity bits will be non-zero so we can return a.

Transmission and Reception

In this section, we describe how the data is transmitted from the sender to the receiver and how the receiver decodes the transmitted data. The process involves generating audio signals from a bitstring and interpreting these signals back into the original bitstring.

Transmission Process

The transmission process involves encoding a given bitstring into an audio signal that can be transmitted over a speaker. The steps are as follows:

- **Bitstring Preparation**:
  The input bitstring is first encoded in this encoding scheme:
  - * A `0` bit is represented by a low-frequency audio.
  - * A `1` bit is represented by a high-frequency audio.
- **Signal Generation**:
  The encoded bitstream is then converted into an audio waveform. This waveform is generated using the PyAudio library in Python.
  - * The duration and frequency of the tones for each bit are carefully chosen to ensure that they can be accurately captured by the microphone and decoded by the receiver.
  - * A preamble (e.g., a specific sequence of bits like `111000`) is added at the beginning of the transmission to allow the receiver to synchronize with the start of the bitstream.

Reception Process

The reception process captures the transmitted audio signal and decodes it back into the original bitstring.

- **Signal Capture**:
  The receiver captures the transmitted audio signal using a microphone. The captured signal is processed using the PyAudio library to obtain the waveform data.
- **Output**:
  After decoding and error correction, the receiver outputs the final bitstring.

Conclusion

This design document outlines our approach to implementing a reliable PHY layer using audio signals. Our focus will be on robust encoding, and effective error detection/correction.