CS378: Computer Networks Lab Fall 2024
Lab 3: Hands-On MAC Layer Design Document

Bathala Shashank(22b1041), Koyyana Suhas(22b1072),
Dayyala John Joseph(22b1063), Harideep Gandu(22b0987)

September 10, 2024

1 Encoding Technique

1.1 Hamming Code

Using the given input bit string (**a**) we can convert it into a new bit string (**b**) which comprises of the data bits (D) and parity bits (P) using **Hamming code**. Let the length of **a** be n, then the length of **b** will be n + $[\log_2 n]$ + 2 (where [.] indicates Greatest Integer Function). The new string **b** is generated by filling the data bits of **a** in the non-exponential (w.r.t 2) positions of **b** and the parity bits of **a** in the exponential positions (w.r.t 2 i.e, 1, 2, 4, 8, ...) of **b** which means **b** will be of the form $P_1P_2D_3P_4D_5D_6D_7P_8...$, where $D_3 = a_1, D_5 = a_2, D_6 = a_3$, ... the data bits are placed continuously at the non-exponential positions in **b**. Whereas a parity bit P_j (where $j = 2^i$) indicates the even parity of the string $D_{i_1}D_{i_2}D_{i_3}...$ and so on, where $i_1, i_2, i_3, ...$ indicates the positions in **b** where each i_r has the i^{th} bit set to 1.

Even Parity: If the number of 1's are even then the parity is 0, otherwise 1.

$$P_j = D_{i_1} \oplus D_{i_2} \oplus D_{i_3}...$$

For example, $P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12}$ as the 2^{nd} bit for 5, 6, 7, 12 is set to 1.

1.2 Preamble + Codeword = Encoded string

Preamble: It indicates the length of the input data string only. From the given problem statement as $n \leq 20$ and the integer 20 is of 5 bits, so we always give the length of the input string in 5 bits.

Codeword: It comprises of Input string (**a**), parity bit (p), Hamming code (**b**). The parity bit indicates the even parity of the string **b** i.e, if the the number of 1's in the string **b** are even then p = 0, otherwise p = 1.

$$Codeword = a + p + b$$

where + indicates concatenation. The reason behind the addition of p and **a** can be seen in the error detection/ correction part. The upcoming discussion comprises only of the transmitted string with atmost 2 bits flipped.

2 Error Detection & Correction

2.1 Number of bit flips

If the total number of 1's in (b + p) are even i.e, even parity of (b + p) is 0 then there might be definitely either 0 (or) 2 flips in (b + p), since before flipping itself (original string) we have the even parity of (b + p) as 0 (p $\oplus$ even-parity of b = p $\oplus$ p = 0). If the even parity of (b + p) is 1 then definitely there is one bit flip in (b + p). For correction it can be classified into two different cases.

2.2 Zero/Two Flips

We try to check whether parity bits tally with that of data bits in b (from the Hamming code method) since we know that parity bits always occupy the exponential positions in b we can easily identify parity, data bits in b at the receiver.
Case-(i): If the parity bits and data bits tally each other according to the Hamming code method then definitely there are no flips in b which implies 2 flips should occur in a. Using original Hamming code (b) we can extract the original input string and original transmitted message.
Case-(ii): If the parity bits and data bits doesn't tally each other according to the Hamming code method then definitely there are 2 flips in (b + p) which implies there are no flips in a. Hence we have the original input string, using this we can extract the original Hamming code (**b**) as well as its parity p.

2.3 One Flip

Here also we try to check whether parity bits tally with that of data bits in b (from the Hamming code method). We can have two different cases here.
Case-(i): If the parity bits and data bits tally each other then definitely there will be no flip in b which implies p is flipped. From the original Hamming code (b) we can extract the original input string and original transmitted message.
Case-(ii): If the parity bits and data bits doesn't tally each other according to the Hamming code method then definitely there will be one bit flip in b. Now we compare the parity bits in b (at the receiver) $...P_8P_4P_2P_1$ and the new parity bits from the data bits in b using Hamming code method $...P_8^*P_4^*P_2^*P_1^*$. If we have a difference between P_{2^j} and $P_{2^j}^*$ then definitely the j^{th} bit of i is set to 1 (where i is the position in b where there is a bit flip) since i falls in the P_{2^j} category (as i has j^{th} bit set to 1) and when i^{th} position in b gets flipped then P_{2^j} also gets flipped. Hence i can be calculated by,

$$i = (...P_8P_4P_2P_1) \oplus (...P_8^*P_4^*P_2^*P_1^*)$$

After evaluating i we can extract the original Hamming code (b) by flipping the i^{th} position in b at the receiver. From the corrected Hamming code (b) we can extract the original input string and original transmitted message.

3 Reception

3.1 Frequency Shift Keying

We are using the implementation of **Frequency Shift Keying (FSK)** communication system using **PyAudio** for the transmission of encoded bit string. **Frequency Shift Keying** is a digital modulation technique where information is transmitted by varying the frequency of a carrier wave between discrete values. In **binary FSK**, two distinct frequencies represent binary digits: one frequency for '0' and another for '1'. The carrier wave's frequency shifts according to the bit being transmitted, allowing the signal to convey digital data.

3.2 Sender Code-Receiver Code

The sender code generates an **FSK signal** based on evaluated encoded bit sequence and transmits it via audio output. Generates sinusoidal tones for bits '0' and '1' and concatenates these tones based on the bit sequence.
The receiver code captures the transmitted audio signal, performs frequency analysis, and decodes the bit sequence. **Epsilon** is used to identify the presence of specific frequencies in the received signal (there might be some difference in frequencies). Captures audio data in chunks and appends it to a buffer and concatenates buffer data to form the complete audio signal.

4 Changes in Design : #'s

Due to some calibration issues in the **Laptop's Microphone**, **Surroundings** there might some problem in sound detection. For example the device may detect some chunks of 0's, 1's even without the sound produced from the sender, then there will be a problem for detection of the transmitted message. To overcome that we are adding few (3-4) **hashes** before and after the transmitted message at the sender and # has a frequency different from 0 and 1. Hence the receiver is written in such a way that it listens and stores only the bit message between two #'s so that we can ensure that our bit message is **clean** (receives the message accurately).

5 MAC Layer

5.1 Headers

Since we have 4 different destinations 0, 1, 2, 3 we can represent them in 2 bits, and we have 3 different senders 1, 2, 3 we can represent them in 2 bits, and also we have 3 different types of messages **Data**, **RTS**, and **CTS** we can represent them in 2 bits. Hence we store the **signal type** in 2 bits, **sender** in 2 bits, **receiver** in 2 bits and we store the entire input message. Therefore we transmit **"message-type + sender + receiver + input message"** for data type and **"message-type + sender + receiver"** for RTS and CTS.

5.2 MAC protocol - CSMA/CA

Introduction: This outlines the design of a **Medium Access Control (MAC)** protocol using **Carrier Sense Multiple Access (CSMA)** with **RTS (Request to Send)** and **CTS (Clear to Send)** for collision avoidance. The protocol is enhanced by detecting collisions through frequency changes, where RTS and CTS signals are used to manage medium access, ensure that no two nodes transmit simultaneously, and handle collisions when they occur.

Protocol Overview: Whenever a node wants to send a message, it first sends an RTS (Request to Send), this RTS message is transmitted at the same frequency as the data bits (1's and 0's) used in the PHY layer. The receiver then checks for collisions by detecting the frequency of the incoming message. If there is no collision, the receiver responds with a CTS (Clear to Send) message, which grants permission for the sender to proceed with data transmission.
If a collision is detected, no CTS is sent. The sender waits for a random time and then attempts the RTS transmission again, reducing the likelihood of repeated collisions.

Collision Avoidance Process:

- **Case of Collision**: If a node detects no CTS after sending an RTS (indicating a collision), it waits for a random time (using an exponential backoff algorithm) before resending the RTS. The random backoff period reduces the chances of repeated collisions between the same nodes.

- **Case of No Collision**: Here the receiver corresponding to the RTS message sends a **CTS** with the the information of sender and receiver meanwhile the other nodes ignore the RTS (since they are not the receivers). Now the sender gets clear that the medium is free and it will get ready to send the data message meanwhile the other nodes also listen to the **CTS** but they can sense that they are not the senders and hence they will wait for a fixed time (say 5 sec).

Frequency-Based Collision Detection: The receiver distinguishes between a clear RTS message and a collision by monitoring the frequency of the received RTS signal. If the receiver detects interference or frequency changes (indicating multiple nodes are transmitting simultaneously), it assumes a collision and does not send a CTS. Nodes that do not receive a CTS assume their RTS caused was involved in a collision and wait for a random period before retrying.

5.3 Auxiliary Transmissions

RTS (Request to Send): A control message sent by the sender node to inform other nodes that it wants to send a data message.
RTS Transmission: When a node (sender) intends to transmit data, it first sends an RTS (Request to Send) to the intended receiver. The RTS message includes: Sender Node ID, Receiver Node ID, Message Type: RTS. This transmission is used to reserve the medium for communication, informing other nodes of the sender's intention.

CTS (Clear to Send): A response from the intended receiver after receiving an RTS, indicating that the medium is free for data transmission.
CTS Transmission: After receiving the RTS, the intended receiver checks if the medium is free (i.e., no collision). If the medium is free, the receiver responds with a CTS (Clear to Send) message, allowing the sender to proceed with data transmission. The CTS message includes: Receiver Node ID, Sender Node ID, Message Type: CTS. The CTS ensures that only the intended receiver responds, while other nodes overhearing the RTS will remain silent.