Networks Lab 3

Akshath Reddy (22b0992), K Sai Aditya(22b0952), Nitin (22b0974), Gurram Suresh (22b0991)

August 2024

1 Physical Layer

The following is what happens at **sender**,

- We mapped 2 bit combinations to frequencies as follows,
  - '01' = 800 Hz
  - '00' = 850 Hz
  - '10' = 900 Hz
  - '11' = 950 Hz
- We split the bitstring which we are supposed to send in pairs and played its corresponding frequency for 0.4 seconds.
- If the total signal length is odd, an extra '0' bit is appended to the sender's bitstring, which is discarded at the receiver's end after checking the total length of the message bits so that we can send pairs of bits at once.
- Each bit pair of the bitstring is transmitted to the receiver's end over 0.4 seconds using sinusoidal waves corresponding to the mapped frequencies.So time per bit is 0.2sec.
- At the receiver side synchronization is based on human reaction speed, hence we added padding before the signal of the form "0011" and at the receiver side we handled padding by discarding starting bits till we hit 2 ones consecutively.
- Stopping the receiver side recording is based on the human on that side who is expected to stop as soon signal stops.

The following is what happens at **receiver**,

- The recording is stored and a bandpass filter is applied.
- The recording is split in 0.4 second chunks and Fourier transform is done in multiple subparts of that chunk , now the frequency corresponding to peaks in FFT graph of each subpart is pushed into a queue, later each element is popped, check which frequency is near to it and update counter, range_counter and append the bit strings as per the counter.
- The assumption here is that human on receiver side stops and starts at appropriate times.

2 Sender Bitstring Signal

The signal sent by the sender is (padding + length of message (5 bits) + additional bits + message) comprises the following components:

- **Padding Bits**: The bit sequence '0011' is prefixed to the data to mitigate errors caused by human reaction time and to synchronize the sender and receiver.
- **Length of message**: length of message in binary in 5 bits
- **Additional bits**: This comprises additional parity bits, xor checksums, ... which is further explained in error correction part of this document.
- **Message Bits**: The actual message data being sent from the sender to the receiver.

3 Error Detection and Correction

In our design, we break the message into 4-bit chunks (padding with zeros at the end if the length is not a multiple of 4). We apply the standard Hamming code method to each chunk, generating 4 parity bits. Additionally, we compute the XOR of all the 4-bit message chunks, which results in a 4-bit value, this is called intended XOR. Since this XOR depends on the message, we include it in the message itself rather than in the preamble. The final transmitted message after preamble has the following structure:

$$(\text{parity bits of each message chunk}) + (\text{parity bits of XOR}) + \text{XOR} + \text{message}.$$

Errors can be introduced anywhere within this message.

At the receiver, during the decode process, after parsing the data into the required format, we first check for errors in (parity bits of XOR) + XOR.

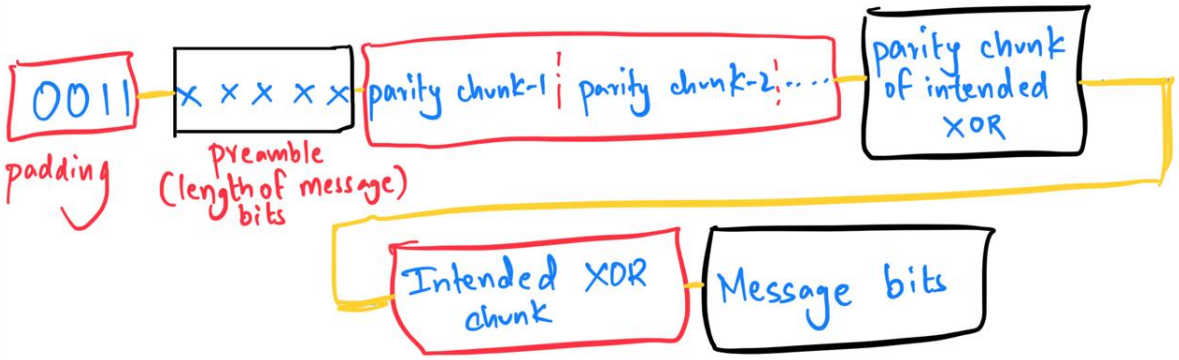

Figure 1: Frame

Case 1: No Errors in (parity bits of XOR) + XOR

If there are no errors in this part, then any errors (if present) are present only in the message bits and their corresponding parity bits.
Now we iterate through all message chunks and it's corresponding parity chunks,

Case 1.1: No Errors in the Message and Parity Bits

If there are no errors, we output the message as is.

Case 1.2: One Bit Error in same message chunk and its corresponding parity chunk

If there is a 1-bit error, we can use the standard Hamming correction method.

Case 1.3: Two bits errors in same message chunk and its corresponding parity chunk

If the 2 bits in error are in the same (message chunk + its parity chunk), we take the XOR of all message chunks .i.e observed XOR and compare it with the received XOR or intended XOR. The positions where there is a difference indicate the error locations. It is possible that the errors are in the parity chunk rather than the message chunk. After correcting the message chunk, we can apply the Hamming function to generate the correct parity bits, which can be compared with the received parity bits to identify the error indices.

Case 2: One Bit Error in (parity bits of XOR) + XOR

If there is a 1-bit error, we can identify and correct it directly using hamming correction method, and then follow the procedure as in Case 1.

Case 3: Two Bits in Error in (parity bits of XOR) + XOR

If there are 2 bits in error, we can assume the message bits are correct since there can be at most 2 bit errors. We can compute the XOR of the message chunks to obtain the correct XOR value and also apply the Hamming function on the XOR of the message chunks to obtain the correct parity bits for XOR. This allows us to identify the error indices.

4 MAC layer with CSMA

In this section, we describe the design and implementation of the Carrier Sense Multiple Access (CSMA) protocol with enhancements such as the Request to Send (RTS), Clear to Send (CTS), and acknowledgment (ACK) signals for unicast communication, as well as its behavior in broadcast communication. The goal of this design is to reduce data collisions in a shared communication medium, improving efficiency and reliability.

4.1 CSMA Overview

The CSMA protocol is widely used in network communication to manage access to a shared communication channel. It operates by allowing nodes to sense the medium (i.e., the carrier) before transmitting data. If the medium is busy, the node waits until it becomes idle before attempting to transmit. This carrier sensing mechanism helps in reducing the probability of collisions. However, collisions can still occur when multiple nodes sense an idle channel and transmit simultaneously.

The basic CSMA operation is enhanced with a collision detection and recovery mechanism, where nodes involved in a collision detect the event and retransmit after waiting for a randomized backoff period. This fundamental aspect of CSMA is critical for maintaining efficiency in a network where multiple devices may compete for the same medium.

4.2 Message Frame

- Here the frame is different from the previous frame.
- The frame consists of src. mac address + dest. mac address + message type + content.
- message type will represent is it an RTS, CTS, message or ACK.
  - 00 – RTS
  - 11 – CTS
  - 10 – message
  - 01 – ACK
- src and dest. mac addresses are as per the given input.

4.3 Unicast Communication with RTS/CTS

In unicast communication, where data is transmitted from one sender to one specific receiver, we introduce the RTS/CTS mechanism. This enhancement reduces the likelihood of collisions, particularly in wireless networks with hidden node problems. The implementation follows these steps:

1. **Request to Send (RTS):** Before transmitting data, the sender node sends a small control message called an RTS to the intended recipient. This message signals the sender's intent to transmit data and reserves the communication channel.
2. **Clear to Send (CTS):** Upon receiving the RTS, the recipient node responds with a CTS message if the channel is free and it is ready to receive data. This CTS message is also overheard by neighboring nodes, signaling them to defer their transmissions, preventing potential collisions.
3. **Message Transmission:** Once the sender receives the CTS, it begins transmitting the data message to the recipient.
4. **Acknowledgment (ACK):** After successfully receiving the data message, the recipient sends an acknowledgment (ACK) back to the sender, confirming successful transmission.

This four-step handshake process significantly reduces the chances of collisions by ensuring that both the sender and receiver agree on the transmission timing. Furthermore, neighboring nodes are made aware of ongoing communication through the RTS/CTS exchange, reducing the chance of simultaneous transmissions.

4.4 Broadcast Communication

For broadcast communication, the RTS/CTS handshake mechanism is not utilized. The sender directly transmits the message to all nodes in the network without requesting permission via RTS or waiting for a CTS response. This approach simplifies the transmission process . Additionally, acknowledgments (ACK) are not required, as the broadcast is not targeted at a specific node but meant for all. Before transmitting, the sender still performs carrier sensing to check if the communication channel is clear. If the medium is idle, the sender proceeds with broadcasting the message to all nodes. This process ensures efficient communication.

4.5 Collision Handling and Backoff Mechanism

Despite the use of RTS/CTS, collisions may still occur under certain conditions. For example, two nodes may transmit RTS messages simultaneously, or there may be collisions in broadcast transmissions. To handle these situations, the CSMA protocol employs a collision detection and backoff mechanism. If a collision is detected (e.g., no CTS is received after sending RTS), the sender waits for a random backoff period before attempting to retransmit.

4.6 Design Considerations

The design of this CSMA protocol implementation takes into account the following considerations:

- **Hidden Node Problem:** The use of RTS/CTS addresses the hidden node problem by ensuring that neighboring nodes are informed about ongoing communications and refrain from transmitting during the data transfer.
- **Scalability:** The protocol can scale to handle larger networks, but performance may degrade if the backoff mechanism is not properly managed, especially in high-collision environments.
- **Broadcast Communication:** The lack of a collision avoidance mechanism in broadcast communication is a trade-off to simplify the protocol and reduce control overhead. This works well in networks where broadcast messages are infrequent.

4.7 Implementation Details

The implementation of this CSMA protocol follows a modular design, separating the unicast and broadcast communication paths. The RTS/CTS exchange is implemented as a function that both the sender and receiver call when engaging in unicast communication. In contrast, broadcast communication bypasses this mechanism entirely, sending the message directly to all nodes.

The modular design allows for easy modification and optimization of individual components, such as the backoff algorithm or the RTS/CTS mechanism, to adapt the protocol to specific network environments.

5 Remark

We have written code for which will work 3 members, but since our laptops are not working well, we will try our best to implement for 3 members, else we may implement for 2.