Project Report of CS120

——by Wu Mingzheng, Yu Zengxu

Project 0:
In this introductory project, we explored the main tools of the course—ASIO4ALL and JUCE. Utilizing these tools, we implemented the functionalities of recording and replaying sound. At the beginning of the semester, with ample energy, we even incorporated a graphical interface using JUCEApplication. However, at that time, we had not anticipated the pressure of the upcoming course.

Project 1:
After gaining preliminary control over the sound card, Project 1 required us to transmit information using the audio channel. Initially, we encoded information using sine waves of different frequencies. Specifically, we employed eight frequency bands ranging from 3000 to 13500 to encode, transmitting information through the superposition of these sine waves. During decoding, we separated different frequency sine waves using FFT, thereby achieving a transmission speed of 8 bits per symbol. However, maintaining clock synchronization during transmission was crucial to correctly interpreting the signals.

To address this issue, we introduced a preamble before the data signal for alignment. Initially, we designed a preamble with a linearly increasing frequency over time, forming a square shape on the spectrum graph. This allowed us to identify the offset and achieve synchronization. However, this type of preamble proved challenging to detect, leading us to design a new one. The new preamble had a stepped frequency increase over time, with four frequency bands assigned. Aligning four times, if successful, allowed us to proceed with decoding. While this preamble solved certain problems, it lacked precision in long data transmissions due to alignment deviations. Considering this, we introduced packet fragmentation, dividing the original data into multiple packets, each with a preamble. Aligning at regular intervals ensured that there were no significant offsets affecting accuracy. Despite these efforts, our final transmission accuracy only reached approximately 80%. We discovered that sound signals are not ideal for signal transmission due to their instability, leading to unreliable Fourier transform results. For instance, we found that speakers output signals of different frequencies with varying intensities, complicating the decoding process. We even attempted using a reference wave to indicate the intensity of different frequency signals dynamically, but this method failed due to the unreliability of the reference wave.

Project 2:
Unfortunately, after completing Project 1, we discovered examples provided by the instructor, featuring a high-accuracy preamble. Additionally, with new transmission devices, we could directly transmit binary symbols (01) instead of using sine waves as before. After changing the preamble and installing the new transmission devices, we finally achieved stable transmission. In Project 2, we needed to implement bidirectional communication between two devices. As we had to manage data packet and ACK interactions, as well as control

whether to send packets or ACKs based on the channel state, we implemented a finite state machine.

Initially, we considered using STL's variant to implement the state machine, setting the state as a variant type with different types representing various states. This would allow us to handle states by overloading the same function, facilitating state transitions. However, due to our insufficient familiarity with this functionality and the required use of lambda functions, we opted for a more straightforward implementation. We began with a Moore-type state machine and later switched to a Mealy-type for convenient updates to timeouts and backoffs. To minimize conflicts, we first monitored the channel and parsed received data in a callback before attempting to send data. During debugging, we observed that running the program in debug mode resulted in unstable sound output, and on computers with poor CPU performance, excessive output also affected stability. This posed significant inconvenience during debugging. Furthermore, Intel Smart Sound Technology was incompatible with the USE sound card, puzzling us for a while until we disabled it, resolving the issue. Unfortunately, due to the high latency of our computer, we couldn't timely detect channel availability and send data during interference intervals, preventing us from completing Task 4. We analyzed that the lack of a USB interface on our computer required the use of an adapter, introducing some delay. In previous latency tests, we directly received through the microphone after broadcasting through the speaker, resulting in significantly lower latency than our actual delay.

Project 3:
During this semester, both of us enrolled in four major courses, and Project 3 coincided with the busiest period. As a result, we only managed to complete Task 1 and Task 2 (Task 3 was later addressed in Project 4). Despite lacking a strong foundation in multi-threaded programming, we found that basic multi-threaded programming was not overly challenging. After invoking the npcap interface, it took us three days to hastily complete the first two tasks.

An interesting incident occurred during the check. Mistakenly assuming that Task 7 required executing our program in the terminal, we placed the generated executable file at the top level of the environment variable, replacing the original system's ping and achieving a similar effect. It was only during the check that we learned this part needed to be accomplished using a virtual network card. At that time, we had no concept of virtual network cards, and it had not been mentioned in the lectures. Later, we discovered that using or not using a virtual network card to complete Projects 3 and 4 were two entirely different approaches. In some ways, using a virtual network card is simpler, but we had no knowledge of its existence before completing Project 3.

We hope that more TA sessions could be scheduled to guide students through the projects, providing advance notice of potential challenges and the technologies required (such as virtual network cards). This way, students won't feel like they are blindly navigating and hitting walls.

Project 4:

Due to time constraints, we were unable to complete Task 3 (NAT portion) in Project 3. This required us to address the NAT issue before starting this project. Unfortunately, we cannot obtain the final three points for the foundational part of Project 3. Initially, our plan was to use a virtual network card and, incidentally, tackle some bonus aspects of Project 3. However, after a preliminary attempt with a virtual network card, we discovered that even when NODE1 lost connectivity, numerous threads were still sending packets to the outside world. These packets had to be sent by the virtual network card, placing significant demands on the efficiency and stability of our physical layer transmission. After weighing the pros and cons, we decided to continue using the previous approach and manually craft DNS and ICMP packets.

Subsequently, we found that for some domain names, the length of DNS packets exceeded 300 bytes, amplifying the issue of unsynchronized clocks between the two computers at the physical layer. The most direct solution to this problem is packet fragmentation. However, we attempted an alternative method by anticipating the occurrence of offsets during decoding and offsetting the data in the opposite direction by one sample for decoding. We adopted a completely new encoding scheme using {1, 1, -1, -1, 0} and {-1, -1, 1, 1, 0} to encode 0 and 1. This encoding, while satisfying the requirement of the sum of amplitudes over one cycle being 0, allowed for a sample deviation, meaning that even if there was a deviation in one sample, it would not lead to decoding errors.

In Project 1, our task was to receive and transmit binary strings (01 strings). However, until this project, we had not yet changed the input format from binary strings to strings where 8 consecutive 0s or 1s are encoded into a byte. This caused significant inconvenience in our programming. We rewrote this section and quickly completed the creation of NAT and DNS/ICMP packets. For non-virtual network card approaches, the workload for completing Task 2 was substantial, and after careful consideration, we chose to abandon it.

During the checking process, despite repeated checks in our dormitory, transmission errors persisted. After more than two hours of struggle, we still couldn't identify the problem. We had heard that whether the computer is plugged in or not can affect the stability of the physical layer. So, we went back to get the computer charger. After plugging it in, and after several rounds of tests, we passed the check. In summary, because we changed the encoding method this time, stability decreased compared to before, but stable transmission was still possible when plugged in. In our previous projects, stable transmission was possible whether the computer was plugged in or not, making it difficult for us to identify the error this time. From this experience, it is evident that whether a computer is plugged in or not can impact physical layer transmission (this varies depending on the computer).

We hope that in this course, the instructions can highlight certain aspects that are prone to errors, such as the impact of plugging in the computer on transmission stability (unless this is part of the assessment). This could save students a significant amount of time, sparing them from prolonged struggles with the physical layer. And I would appreciate it if I could be informed that Project 4 Task 1 is dependent on Project 3 Task 3 before Project 3 ends.

Conclusion:
The projects in the computer networking course serve as an excellent complement to classroom instruction. In lectures, we often learn knowledge by reviewing slides and listening to the teacher's explanations, but our mastery of certain details may not be proficient, and our understanding of underlying principles may not be deep enough. However, through practical implementation, we not only review the knowledge learned in class but also gain insights into relevant expanded topics. Additionally, through repeated testing and practical exercises, our understanding of computer networking gradually deepens.