# Computer Networks Project Summary

Junzhe Yu
*School of Information Science and Technology*
*Shanghaitech University*
Shanghai, China
yujzh1@shanghaitech.edu.cn

Zaizhou Yang
*School of Information Science and Technology*
*Shanghaitech University*
Shanghai, China
yangzzh@shanghaitech.edu.cn

***Abstract*—This project report summarizes our efforts in understanding, developing, and overcoming challenges in the course project. The primary emphasis is on the reliability and effectiveness of the physical and MAC layers. The project involved extensive testing, coding, and troubleshooting with a specific focus on improving link performance. The core objective of the project is to identify problems, enhance engineering capabilities, grasp key techniques and standards, and refine coding and development skills.**

***Index Terms*—computer networks, acoustic link**

## I. Overall Impression

Generally, we'd say this project is meaningful. We've encountered many challenging problems related to this project. They cultivated our engineering capabilities, and also made us familiar with some of the course content. Sometimes the tasks even encouraged us to dive deeper into some key techniques and standards. Finally, the core of the project is about coding and project developing. This project really enhanced our related skills.

To be honest, the project is quite tough for us. From time to time, there would be frustrating moments. Several times we kept performing tests for days until we located the problem. From hardware to software, there were too many things to adjust. Thus the problem could be hidden anywhere, especially when you make contact with existing implementations, including open source packets and the OS. Besides, even individual tasks can appear huge, it's not always easy to divide them into parts that can be effectively verified.

We suggest optimizing the coverage of course content in the projects. Throughout the four parts of the course project, too much emphasis is put on the reliability of the physical layer and MAC layer, because almost no task is possible without a high-performance link. We spent a great portion of our effort in troubleshooting and improving our link. The project even revisited the setup of the two layers to obtain the performance gain from the switch to cable connection. However, our curiosity of many other parts of Computer Networks isn't satisfied. For example, we haven't got a deep insight into the implementations of congestion control, routers, BGP, etc. We are still curious about how our packets go further.

## II. Project 1. Acoustic Link

### A. Tools

The biggest choice we had to make was presented to us right at the beginning of the course. We actually had no choice because we started early, and at that time only saw the document from former semesters. In terms of driving ASIO, the only suggestion was using the packet JASIOHost. So we used JASIOHost and Java.

It took a while to get familiar with our tools, including the APIs of JASIOHost and the settings of ASIO4ALL. We could then generate the frequencies at will.

### B. Modem

The next part we worked on is modem, i.e., modulation and demodulation. Since *Dhwani* [1] is provided in the project documents, we chose to adopt its settings for building an acoustic link.

The same as *Dhwani*, we use OFDM to modulate our bits and STFT to demodulate the signal. OFDM is good at fully utilizing the frequency band and combat multi-channel effect.

It's worth mentioning that the modem implementation, as well as many other elements in the physical and MAC layer, includes many parameters you want to tune carefully. Your project won't work after carefully adjusting all your parameters.

### C. Preamble and Aligning

As an indispensable module of a link, *Dhwani* [1] Also includes detailed setting of the preamble. At the receiving end, we calculate the correlation of the received and expected signal. Once the correlation value is greater than a threshold, we would make the judgement on the arrival of the data frame. For better aligning accuracy, we would also find the nearby maximum correlation.

Till now, everything was ready, so our tests began. However, many kinds of factors, especially real-world factors, made the testing process extremely tortuous. We were switching between different microphones, loudspeakers and sound cards till the system worked, unaware of their differences because the time was limited. Well, we didn't think the different device combinations made much of a difference. What surely made a difference, was never performing tests in a small room. When we suddenly realized we should test our system in a classroom where the check was supposed to be performed, things became way better probably because there's much less multi-channel effect.

The system was heavily tested, so naturally, the parameters were heavily tuned. The settings provided by *Dhwani* [1] was

not always good with our system. Looking back, it was quite worthwhile to develop a UI to adjust all the parameters at will during runtime.

## III. Project 2. Manage Multiple Access

### A. Protocol Stack

The first thing to do is to clarify the protocol stack, with communication between layers through interfaces to ensure a clear code structure. For example, we have implemented the following protocol stack: the ASIO layer, the physical layer, the MAC layer, and the APP layer.

### B. MAC Frame Design

- *Source MAC address*
- *Destination MAC address*
- *Type*: Indicate whether the data is an ACK or actual data
- *Sequence number*: Unique identifier assigned to each frame of data to prevent duplicate receptions caused by the loss of ACK.
- *CRC8*: Cyclic Redundancy Check with a 8-bit output, used to detect errors in header
- *Payload*: The data in the upper layer
- *CRC32*: Cyclic Redundancy Check with a 32-bit output, used to detect errors in payload

### C. Multi-threading Management

When introduce MAC layer, we need to consider how to manage the sending thread and receiving thread. Our basic idea is that sending is a top-down action while receiving is down-top. We utilize semaphores to manage the behavior of different threads. For example, when MAC layer intends to call physical layer to transmit frame, we block the detecting thread running in physical layer to release resources. Thus, we could avoid lag of voice to some extent.

### D. Back-off Strategy

We implement reliable transmission using busy waiting. When an ACK is not received after the timer has expired, it indicates a detected collision, then expire *binary exponential back-off* before resend. But when facing a jamming task, it is challenging to design a reliable and general algorithm that fully utilizes the idle window of the link. We have made many attempts, such as having one device not perform back-off while the other does, but this affects fairness. We also tried to have two devices implement alternating back-offs to ensure that each device can alternately use the idle window, but the performance still lacks stability.

### E. Precaution

- Disable AGC: AGC (Automatic Gain Control) is a speaker option in Windows that is very hidden and enabled by default. It can adaptively adjust the input volume. It is worth taking the time to check if AGC is turned off before testing, as even when you disable it, it may still be enabled by default when switching to another sound card.
- Ensure that the devices share a common ground, especially when using ASK modulation, as the issue may be related to voltage.

## IV. Project 3. To the Internet

In this project, we were required to deal with real Internet packets. We implemented a series of forwarding strategies between the Internet and our *Aethernet* to allow some network functionalities through acoustic connection.

### A. To Capture and Inject

In the document, the system interface *Pcap* and packages to manipulate the interface like *Pcap4J*, as well as the important debugging tool *Wireshark* are introduced. These tools enabled us to capture and inject packages.

### B. Get a Reply From a Server

To allow ICMP ping through the Aethernet to the Internet, we should first know how to correctly generate and forward requests with our toolchain.

There's definitely too much work if we start from nothing, constructing the packets byte by byte. Fortunately, *Pcap4J* has elegant APIs to construct all types of protocols as long as you tell its builders what to fill in each field. Also, your system can also generate proper packets.

If you also plan to use *Pcap4J*, please be cautious of all the checksums. Packets resolved from byte strings will have the option *CorrectChecksumAtBuild* set to false by default. Apart from checksums, MAC addresses are also not easy for you to notice. The server won't reply you unless all the important fields are filled in correctly.

### C. Better MAC Required

Once you solved all the problems related to the protocols, it's time to apply Acoustic link to deliver the packets you made.

The first issue was the length of the data frame. MAC layer shouldn't be able to probe into upper layers of the protocol stack to see the length of the payload. So, a *length* field must be included into the MAC layer to allow the receiving end to correctly determine the length it should expect.

The second was to continue optimizing the MAC layer to satisfy new requirements. The minimum RTT demanded by the tasks would be almost impossible for our implementation if there's always re-transmission for each MAC frame.

### D. Forwarding Logic

As we went deep into the implementation, we found that things weren't that simple as two nodes. Each node, in reality, comprised two nodes: one for the Internet adapter and the other for the Ethernet adapter.

Fortunately, this part of implementation was not that error-prone and was easy to verify. There weren't many unknown factor, almost everything was visible in our code.

## V. Project 4. Above IP

### A. Get a DNS response

Despite the experience we gained in modifying packages in Project 3, we are still trapped in the DNS server not responding. We finally found out that the *checksum* filed in UDP need to be corrected even the UDP layer is not modified. Since we ignored the fact that the *checksum* in UDP is related to the IP layer.

### B. Modify the TCP Packet

Unfortunately, we did not realize the need to implement a TCP layer. However, for the purpose of the assessment, modifying the sequence number and acknowledgment number was sufficient, and that became our approach. Moreover, we were not clear about the extent to which we needed to implement TCP.

## VI. Conclusion

In conclusion, the project proved to be a considerable challenge due to its scope and multiple variables. Despite the initial struggles, it significantly honed our engineering capabilities, understanding of key techniques, and familiarity with course content. The project focused heavily on physical layer and MAC layer reliability, which necessitated extensive efforts in troubleshooting and improving the link. Nevertheless, we acknowledge that there's an enormous terrain of Computer Networks that remains to be explored, especially regarding congestion control, routers, BGP, and more. Moving forward, it is recommended that future projects carefully balance the focus on various aspects of Computer Networks to provide a more extensive understanding of the field.

## References

[1] R. Nandakumar, K. K. Chintalapudi, V. Padmanabhan, and R. Venkatesan, "Dhwani: secure peer-to-peer acoustic NFC," in Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, in SIGCOMM '13. New York, NY, USA: Association for Computing Machinery, Aug. 2013, pp. 63–74. doi: 10.1145/2486001.2486037.