Computer Network Project Report[⋆]

Ziqi Wang[1][2021533045] and Jiaying Du[2][2021533037]

[1] ShanghaiTech University wangzq@shanghaitech.edu.cn
[2] ShanghaiTech University dujy2@shanghaitech.edu.cn

Abstract. This report presents a comprehensive overview of our journey through the Computer Networks course (CS120) at ShanghaiTech University. It encapsulates our experiences, challenges, and achievements across four distinct projects, ranging from basic network programming to the integration of advanced networking concepts with real-world applications. Each project not only enhanced our technical skills but also provided valuable lessons in teamwork, problem-solving, and the practical implications of theoretical knowledge in computer networking.

Keywords: Computer Networks · Audio based Networks · Cpp programming · Teamwork · Collaboration · Software Engineering Standards · Project Management · Technical Skills

1 Introduction

The field of computer networks is a cornerstone of modern communication and information technology. In the Computer Networks course at ShanghaiTech University, we embarked on a journey that took us through the intricacies of network programming, the development of network-based applications, and the exploration of advanced networking concepts. This report outlines our experiences in four major projects, highlighting the challenges we faced, the solutions we developed, and the invaluable lessons we learned. Our journey through these projects not only solidified our understanding of fundamental networking principles but also allowed us to appreciate the complexities and real-world applications of these concepts.

2 Audio part

2.1 Our Insights and Accomplishments in Computer Networking

We have acquired proficiency in digitizing and modulating signals for acoustic transmission over wired mediums. This includes understanding the composition of network packets, encompassing a preamble, header, and payload. Through the manipulation of input audio buffers, we have developed the capability to demodulate and extract packets from acoustic signals. Our knowledge extends to

[⋆] Course delivered by ShanghaiTech University error detection and correction, particularly employing Cyclic Redundancy Check (CRC) for rapid integrity verification of packets. Furthermore, in the realms of network diagnostics and domain name resolution, we have advanced our skills in constructing and managing a bespoke audio-based network infrastructure.

2.2 Challenges Encountered

High Delay on Windows Audio We choose to use the RTAudio, a set of C++ classes that provide a common API for real-time audio input/output across platforms. We didn't figure out how to add ASIO support and use that at the very beginning. So we used the WSAPI in our first project. The initial challenge we faced was significant latency. Although this could be resolved with a sliding window algorithm, it was a considerable concern initially. It seems fine at that time, but soon in the second project, we need really low-latency communication between devices. We go through deeply into the CMakefile configurations for compiling the whole project, and finally add ASIO support and use ASIO for our project.

Designing a Real-Time System Initially, we didn't pay too much attention to the "real-time" operation, we used several programs and ran a shell script to complete the task. We first create the packet to send, then we call the send program to send the packet. This method does not work anymore in the second project. Therefore, we pay real attention to the "real-time" network. We still made another mistake, though. To optimize the latency in our network and reduce the workload, we did not generate packets in the progress but before the progress. Such that there won't be delays caused by computing. That was a mistake, but we didn't have enough time to fix that, for the ping part, it functioned, our TA pointed out this issue and we fixed it then. After we promptly rectified this issue, we got a solid foundation for the subsequent projects.

3 Network Component

3.1 Exploring Wireshark

In our exploration beyond standard lectures, we extensively utilized Wireshark, a powerful network protocol analyzer, to deepen our understanding of network activities. This tool enabled us to capture and interactively browse the traffic running on a computer network. Our focus was primarily on monitoring fundamental network activities, such as ping requests and replies.

By closely examining various elements like the source (src) and destination (dst) addresses, along with different protocols, we gained a practical understanding of how network packets are structured and communicated. This hands-on experience was crucial in comprehending the intricate details of packet formation and network behavior.

Wireshark's ability to display detailed information about network packets in real-time provided us with an invaluable learning tool. We were able to observe the nuances of network communication, such as how packets are routed, how data encapsulation works, and how various protocols like TCP, and ICMP function in different scenarios.

One key aspect of our learning involved sending custom packets and then using Wireshark to verify if the responses received were as expected. This process not only solidified our theoretical knowledge but also sharpened our practical skills in network troubleshooting and analysis. For instance, when sending a ping request, we could dissect the ICMP packet, look at its header, payload, and understand how the echo request and echo reply mechanism works at a granular level.

Moreover, Wireshark's filtering capabilities allowed us to isolate specific communications and study them in isolation, which was particularly beneficial in crowded network environments. By applying filters, we could focus on specific IP addresses, protocols, or even individual conversations, enabling us to analyze network behaviors and troubleshoot issues with greater efficiency.

This exploration with Wireshark significantly supplemented our classroom learning, offering a hands-on approach to understanding the dynamics of computer networks. It bridged the gap between theoretical network concepts and their practical applications, enhancing our comprehension and proficiency in the field of network engineering, which is a good point for reducing the workload for us to find the bug in our network architecture.

3.2 Implementing ID-Specific Ping

In the ID-Specific Ping implementation, we tackled a unique challenge in network communication, particularly in the context of Network Address Translation (NAT) employed by routers. Our scenario involved two devices, Device A (an audio device) and Device B (a standard device), both operating under Router C. These devices were trying to communicate with an external Device D through ping requests.

A critical issue arose due to the nature of NAT: Router C, when forwarding packets from Device A and Device B, replaced their original IP addresses with its own IP address. This mechanism introduced a significant challenge: When Device D responded, how would Router C know whether the reply was meant for Device A or Device B? This differentiation was crucial for proper routing of the response packets to the correct device.

To solve this problem, we devised a strategy that involved embedding a specialized identifier within each packet sent from Devices A and B. This identifier was conceptually similar to an entry in a NAT table. Just like a NAT table maps the original IP and port numbers to new ones, our identifier served as a unique marker for each device under the router. This way, when a reply came back from Device D, Router C could inspect this identifier and determine whether to route the response to Device A or Device B.

We ensured that these identifiers were unique and consistently assigned to maintain a reliable mapping. The implementation required careful consideration of packet structure and the impact of additional data on the overall network performance.

This solution not only addressed the immediate challenge of distinguishing between devices behind a NAT but also provided a deeper understanding of NAT operation and its implications in a networked environment. It highlighted the complexities and intricacies of network communication, particularly in scenarios involving multiple devices and address translation.

4 Other Issues

In this section, we delve into the technical and logistical challenges encountered throughout our Computer Networks projects, particularly from Project 3 onwards. This part of our journey was marked by a significant shift in our approach to project development, primarily in terms of utilizing CMakeLists and managing library dependencies.

Initially, our projects heavily relied on modifying test codes from the RtAudio library. This process, while insightful, was fraught with complexities, especially in terms of compiling and linking libraries. We started with using the provided CMakeLists from RtAudio, but this approach had limitations, especially when it came to cross-device compatibility and modifications.

The introduction of the PcapPlusPlus library in Project 3 marked a turning point. Its detailed tutorials and provided CMakeLists eased the integration process. However, we encountered challenges in incorporating it with RtAudio, mainly due to the different versions of winsock.h used by RtAudio and winsock2.h used by PcapPlusPlus. This led to compatibility issues, which we initially tried to fix with various methods. Eventually, we resolved this by directly replacing the winsock content in the Visual Studio directories to align with winsock2.h, enabling us to compile the projects successfully.

Recognizing the limitations of our earlier approaches, we took the initiative to write our own CMakeLists. This not only facilitated the compilation of the two frameworks into library files but also streamlined the process of linking these libraries to our projects. This approach enhanced the reproducibility of our projects from Project 3 onwards.

Another significant change was in how we managed our library files. To improve convenience across different machines, we placed the library files in the root directory of the C drive. This allowed for easier setup in Visual Studio Code, particularly with the CMakeLists plugin, enabling one-click compilation and more efficient coding, debugging, and running processes.

These technical challenges and our approaches to overcoming them were critical in our learning journey. They provided us with a deeper understanding of software project management, library integration, and the importance of a well-structured development environment. This experience not only enhanced our technical skills in C++ and network programming but also offered invaluable lessons in problem-solving and adaptability in the face of complex software development challenges.

5 End-of-Course Reflections

Our project's primary regret was not being able to implement a virtual network interface card (virtual NIC) and the functionalities associated with it. We did explore some virtual network implementations using TUN, where we set up a virtual NIC along with its network segment. This setup was intended to direct corresponding IP packets to this virtual location. However, this virtual NIC acted like a void; it captured the data being sent to this segment but didn't process it further.

Our plan was to monitor network traffic on our devices and, when data was sent to this segment, capture it and then transmit it using our own audio transmission method to another device. The same process was to be replicated on the receiving device, effectively creating our own virtual NIC system.

Unfortunately, due to time constraints and the workload, we were unable to bring this aspect of our project to fruition. We spent significantly more time than anticipated on resolving issues related to audio transmission. This unexpected focus on audio left us with little time to address the networking challenges, particularly the implementation of the virtual NIC and its associated functionalities.

In retrospect, while our project achieved success in several areas, the inability to implement the virtual NIC remains a notable omission. This experience, however, has taught us valuable lessons about project management, the importance of time allocation, and the complexities involved in integrating different technological components into a cohesive system. The journey through this project was a profound learning experience, emphasizing the dynamic nature of technological problem-solving and the need for adaptability in the face of unforeseen challenges.

6 Conclusion

In reflecting upon our journey through the Computer Networks course, a significant aspect of our learning extended beyond the technical scope, delving into the realms of teamwork and collaboration. Throughout the course, our team faced challenges related to task division, leading to occasional conflicts among members. These experiences, while initially difficult, proved to be invaluable lessons in understanding and mastering the art of cooperative work and effective task distribution.

Through these interactions, we learned the importance of clear communication, mutual respect, and the need for a balanced approach to task allocation.

Moreover, this course highlighted the critical importance of adhering to software engineering standards and maintaining a clean and organized project structure. We recognized that well-structured code and organized project management are not just about technical proficiency, but also about ensuring readability, maintainability, and scalability of our work.

In conclusion, our experience in the Computer Networks course at ShanghaiTech University has been profoundly enriching, providing us with not only technical skills but also essential lessons in teamwork, project management, and the significance of software engineering principles. These lessons will undoubtedly be instrumental in our future academic and professional journeys.

7 Appendix

1. Our Github Repo: https://github.com/wzqvip/CS120-Computer-Networks
2. RtAudio Repo: https://github.com/thestk/rtaudio
3. PcapPlusPlus Repo: https://github.com/seladb/PcapPlusPlus