CS120 Project Report

You Wu and Liwei Pang
School of Information Science and Technology, ShanghaiTech University

Introduction

In the four projects, we built a local area network called Aethernet and connected it to the Internet. We got familiar with Java language and deepened our understanding of the implementation of lower-layer network protocols and the usage of upper-layer network protocols.

Project 1

In project 1, we implemented the physical layer of Aethernet from scratch. The most important issue is how to modulate the bits we want to transmit and demodulate the received bits. We applied Phase Shift Keying, which we believed to be the simplest. We used 44 samples to represent a bit. If we use more bits to encode a bit, the accuracy would increase while the speed of transmission would decrease. We found the value to satisfy both accuracy and speed requirements. Both 0s and 1s are conveyed by a carrier wave of frequency 2000Hz while their phases are opposite. When we receive 44 samples and want to determine whether they represent a 0 or 1, we compute the dot product between the received samples and the predefined template samples of 0. We demodulate the received samples as 0 if the result is positive and 1 if negative. The modulation and demodulation method performed well and we achieved an accuracy of 99.97% even though we did not implement FEC codes.
We also encountered some problems during the process. The accuracy was lower than 50% for a long period, which upset us a lot. We tuned many parameters in the config file and finally found that we did not modify the sample rate of the system when we modified the sample rate in the config file. The overall experience of this project was good, and we think it could have been better if we had not made such a stupid mistake and wasted much time tuning parameters.

Project 2

In project 2, we implemented the MAC layer of Aethernet based on project 1. We were provided a toolkit to reimplement the physical layer. We got familiar with the implementation of the physical layer in project 1, the cable connection also provided a more stable transmission medium, so we finished task 1 with ease. The following tasks require us to allow multiple nodes to share the common medium, which is the main challenge of project 2. We noticed that in the first four tasks, there are no more than two nodes transmitting simultaneously. We came up with a solution with a single thread and there would be no conflicts theoretically. There are four states in the state machine of each node, which are Tx Frame, Rx ACK, Rx Frame, and Tx ACK. When there is only one node transmitting, the transmission node would start with Tx Frame and cycle between Tx Frame and Rx ACK. The reception node would start with Rx Frame and cycle between Rx Frame and Tx ACK. When there are two nodes transmitting simultaneously, one node would start with Tx Frame and the other would start with Rx Frame. They both cycle between all four states and there is always one node transmitting.
Our implementation did not consider the realistic conditions of the medium and was definitely not the ideal one for the project, but it was enough for all four projects except for some optional tasks. We finished this project smoothly and thought it was the best-designed one among the four projects. The toolkit simulated the environment of early local area networks and enabled us to understand the implementation of the MAC layer thoroughly.

Project 3

In project 3, we connected Aethernet to the Internet. In task 1, we were required to enable ICMP Echo between the Aethernet host and the Aethernet router. The Aethernet host started with Tx Frame and cycled between Tx Frame and Rx ACK. The Aethernet router started with Rx Frame and cycled between Rx Frame and Tx ACK. The frame had a simple structure and was not consistent with the IP protocol. The payload only contained the source and destination IP addresses. The router checked the destination IP address of the received frame. It was the IP address of itself in this task. So we did not need to forward it to the Internet and only needed to transmit ACK to the host. In task 2, we were required to enable ICMP Echo between the Aethernet host and a generic host. This task required us to transmit and receive ICMP packets, so we added two states to the state machine, which were Tx ICMP Packet and Rx ICMP Packet. The Aethernet host started with Rx Frame and cycled between Rx Frame and Tx ACK. The Aethernet router started with Rx ICMP (Echo) Packet (from the generic host) and

cycled between Rx ICMP (Echo) Packet (from the generic host), Tx Frame Tx ACK, and Tx ICMP (Echo Reply) Packet (to the generic host). In task 3, we were required to enable ICMP Echo between the Aethernet host and a public server. The Aethernet host started with Tx Frame and cycled between Tx Frame and Rx ACK. The Aethernet router started with Rx Frame and cycled between Rx Frame, Tx ICMP (Echo) Packet (to the public server), Rx ICMP (Echo Reply) Packet (from the public server), and Tx ACK.
Our implementation was not elegant because the Aethernet frame structure was not consistent with the IP protocol and the router reached the protocols above IP. It was enough for simple protocols like ICMP while implementing other protocols like FTP would need extra effort. We mastered the usage of the Wireshark packet capturing tool and Pcap4j library, and the flow of general network protocols.

Project 4

In project 4, we added some features to the Aethernet. Task 1 is a simple extension of the last project. The other tasks are related to the HTTP protocol, which is much more complicated. So we did not finish them. In task 1, the input of ping could be the domain name instead of the IP address of a public server. When the router received a frame, it first checked whether it contained a domain name or an IP address. For a domain name, it transmitted a DNS packet to a DNS server and got the IP address corresponding to the domain name in the received DNS packet. The rest of the process was the same as project 3 task 3.
In our implementation, DNS was done by the router instead of the host. We chose this implementation for project 4 because it was the most convenient way based on what we have implemented in project 3. The protocols we support were relatively simple so the performance was not bad. However, under the circumstance where multiple hosts work simultaneously and the protocols are more complex, our implementation may have poor performance because we put too much workload on the router.

Conclusion

The four projects helped us better understand the computer network protocol stack and provided a good exercise in writing robust and extendable code. The projects are challenging without too much unnecessary workload. We think the first two projects are the highlight. They require us to build the physical and MAC layers of Aethernet from scratch and cover almost

everything about these two layers taught in the class. The last two projects are also exciting but there are still many interesting topics at the network layer and above that are not covered, such as routing algorithm and congestion control. Maybe more techniques like simulation can be used to enrich their content. Finally, we sincerely thank the efforts of Professor Yang and TAs to continuously improve the projects and wish them to be better and better.