CS120 Project Report: Aethernet

Members:

- Name: Mokai Pan
  Affiliation: School of Information and Science and Technology, ShanghaiTech University
  Email: panmk@shanghaitech.edu.cn
- Name: Xihe Yu
  Affiliation: School of Information and Science and Technology, ShanghaiTech University
  Email: yuxh@shanghaitech.edu.cn

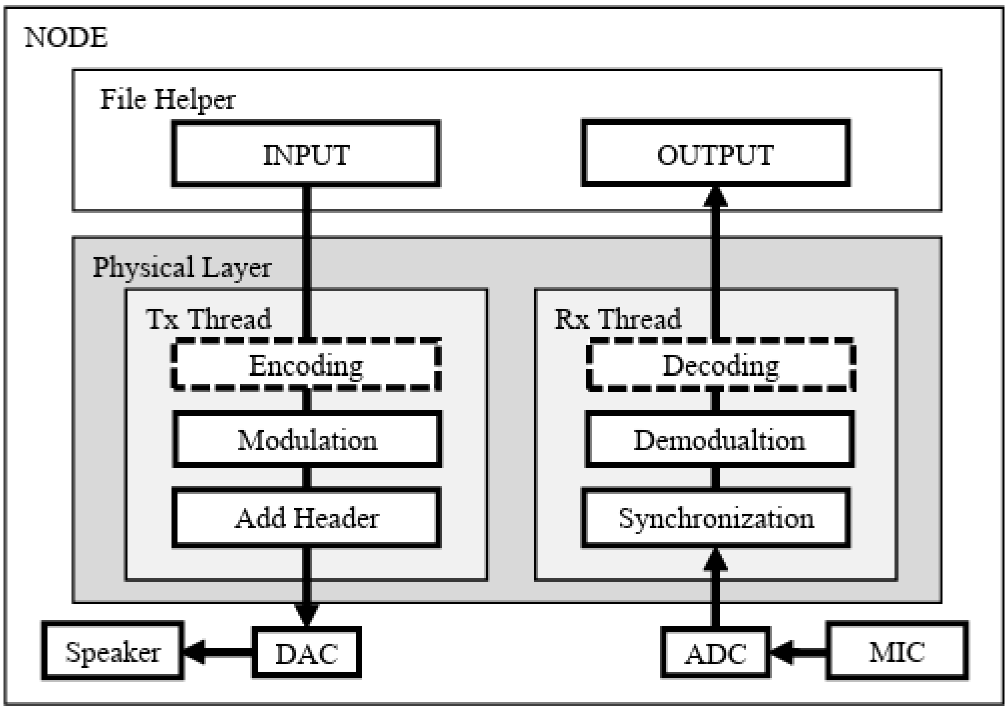
Introduction

For a whole term, we built a local area network named Aethernet using audio signals and connected our Aethernet to the real network. We learned to transmit data using audio and implement some protocols of the Internet in our Aethernet.

Project 1. Acoustic Link

Stories :

- As the old saying goes, it's hard to start everything: this class is no different! For me and my teammate, it was the first time that we came into contact with Java and dealing with the sound data, and we encountered a lot of challenges when doing the first project, not only on the code level but also on the cognitive level. The best way to learn a subject is to combine theory and practice.



Chanllenges :

- According to the recommendation of TAs and teacher, we chose Java as the language of the whole project. It was also the first time that we used Java to code a programme, so we met lots of difficulties.
- Understanding the concepts of preamble, bit rate and other related concepts well took us much time.
- The most challenging part was task 3, because, to most extent, the task 3 was based on hardware (e.g. sound card of the computer and the environment), which caused that it was very hard to debug for our algorithm. For example, we could not confirm which part of our implementation on earth went wrong.

Designs && Solutions :

- We searched for much information about how to design a preamble and related detecting method, but almost had gained nothing. Fortunately, there was teacher's reference code of detecting a preamble, we took the similar design: a 480-length preamble and Channelpower detection:

```
power = power*(1.0f - 1.0f / 64.0f) + (float)Math.pow(current_sample, 2.0f) /
64.0f;
```

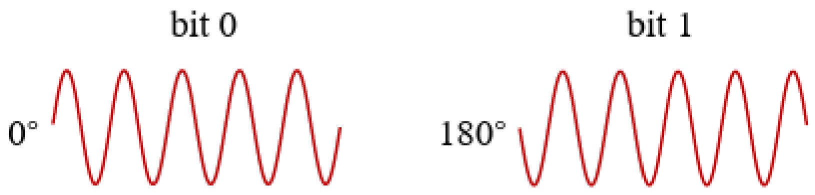
- Here, the power gave us a similar estimate of the power in the channel.
- We understood the relationship between the length of the wave signal corresponding to each bit and the bit rate, and we used PSK modulation to differentiate between 0s and 1s so that we could then demodulate to obtain the transmitted signal. Namely, we learned the mutual conversion of the digital signal and analog signal.
- For PSK modulation:

```
frame_wave[j*48+k] = cos_carrier_wave.get(j*48+k) * (data_every_frame.get(j)*2-1);
```

- For PSK demodulation:

```
sum = 0.0f;
for (int k = 10 + j * 48; k < 30 + j * 48; k++)
    sum += data_signal_remove_carrier_after_smooth[k];
if (sum > 0.0f)
    decoded_data.add(1);
else
    decoded_data.add(0);
```

- Simply,
    - $x_1(t) = cos(2\pi ft)$ and $x_0(t) = cos(2\pi ft + \pi)$
    - so : $\int x_1(t) * x_1(t) > 0$ and $\int x_1(t) * x_0(t) < 0$



- We designed a GUI interface to control operations about several different tasks, which seemed to make the result simple and beautiful.

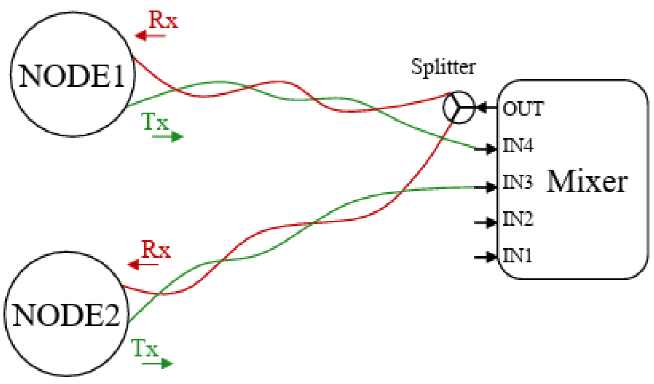
Suggestion :

- We used the simple method of PSK to modulate and demodulate the signal, and the result was not good. The task 3 of project 1 was very metaphysical and had a lot to do with the configuration of the computer as well as the environment, which was a very unfriendly point.
- In addtion, project 1 was the base of all projects, so if project 1 was designed unfriendly for the following 3 projects, here came much problems and obstructions. So we suggested that in the document of project 1, it was better to notify the students that project 1 should be carefully designed and implemented. We had this problem and it was hard to refactor the code because it needed lots of hard work.

Project 2. Manage Multiple Access

Stories :

- This project was arguably the most challenging one, after understanding the relationship between 0/1 bit and wave signals, the problem we faced changed from how to transmit to how to stabilize the transmission, which started with modifying our code to adapt it to our hardware: that is, the computer.
- With cable transmission, the stablility and accuracy of sound was promised. What we most focused on was to improve the efficiency of transmission.



Chanllenges :

- Different computers had different configurations and different sound card settings, so we spent some time on how to set up the right sound card, and the most important thing was that computers had different hardware facilities, they processed information at completely different speeds, the most difficult part was how to synchronize them, which bothered us for a long time, we needed to adjust our code to be able to adapt to both computers, and it was troublesome.

Designs && Solutions :

- We kept the code of our project 1 and designed a kind of Acknowledge (ack) in our acoustic network.
- In project 1, 48 signals represented for a bit. In project 2 task 1, we need to improve our efficiency from 1250 bytes in 15 seconds to 6250 bytes in 10 seconds. We considered to combine both PSK and ASK together and also subtract the number of signals represented for a bit.

```
// A simple ASK modulation method
if(frame.get(j) == 0)
    frame_wave[j*Util.data_size+k] = (float)(0.5) *
carrier.get(j*Util.data_size+k);
if(frame.get(j) == 1)
    frame_wave[j*Util.data_size+k] = carrier.get(j*Util.data_size+k);
```

- Take the first bit as PSK modulation, the last three bits as ASK modulation (For different combinations, multiply $\frac{i}{8}$ to control their amplitude, i = 1,2,3...,8), so we multiply the efficiency as four times as before. Finally, we got equivalently 6 signals represented for a bit, which is equally about 1000 bytes per second.
- We took actions such as delaying, waiting for inputs that would otherwise have been in a dead loop to deal with the synchronization problem. According to the ACK we received, we checked the packets that were not received and re-sent those packets.
- For retransmitting the frame, CRC checking is a good idea. Retransmission might occur in four situations below: the data frame might be lost, the CRC checking might not pass, the ACK might be lost or the timeout occurs. For CRC check, we considered simple CRC4 checking and compute the checksum.
- For CSMA, we didn't have enough time to solve this problem in certain time, but we had attempt to solve it after the deadline: every time before we sent a frame, we needed to listen to the channel for whether there was another one sending (collision):

```
while(true) {
    for (int i = 0; i < listen_recorded.size(); i++)
        avg_power += listen_recorded.get(i) / listen_recorded.size();
    if(avg_power > threshold){
        // collision
        // sleep for some time
    }
    else{
        // no collision
        break;
    }
}
```

Suggestion :

- The difficulty span between the task 2, 3 of the project 2 and the project 1 might be a little large, it needs lots of time to test and debug. And it was in the mid-term, we might lack time to spend on it.

Project 3. To the Internet

Stories :

- After learning to understand the knowledge of ICMP protocol, Router and NAT, we start to realize these problems in project 3, we need to simulate the Ping command to realize the forwarding function of the router and NAT, of course, these are all based on the sound signal. The implementation helps us realize a toy net, and this is a big gain for us!

Chanllenges :

- It cost us some time to understand the ICMP protocol and Pcap4j library. Although there was some examples given by the lib author, it was really hard to first understand the logic of the code.
- How to create a same and right packet as the real ICMP packet was difficult too. We had not experienced any problems about the network programming.
- Besides, when implementing the router, we encountered a difficulty in that the packets sent would often be lost, in fact, this problem was also encountered during check, and it was later found out that this was due to the unstable campus Wi-Fi, which caused us quite a few troubles.
- Understanding the functionality that was to be implemented in this project was a major difficulty. In this project we were mainly facing the problem of how to migrate the functions of the actual real network to the acoustic network, which seemed to be easy, but in fact, it was a big challenge.

Design && Solutions :

- According to teacher's recommendation, we used a Java library called Pcap4j to send or capture a real ICMP packet, which connected our acoustic network to the real network. The advantage of capturing ICMP packet by Pcap4j was that the programme would be blocked until it captured the target number of ICMP packets, which meaned that we would not have to worry about when or whether we had captured the packets, which bothered us a lot in our acoustic network.
- To capture the target packet from the Network Interface, we could set some filters for target IP address and source IP address:

```
// set filter
StringBuilder sb = new StringBuilder();
sb.append("src host ").append(SrcIpAddress).append(" and dst host
").append(DstIpAddress);
String filterString = new String(sb);
handle.setFilter(filterString, BpfProgram.BpfCompileMode.OPTIMIZE);
```

- For a router, we transferred the ICMP packet according to the IP address: If the destination IP address belongs to the directly connected networks, forward it to the corresponding right port. Otherwise, we would discard it.
- We could set listener, which helped us to deal with something after we captured the packet. For instance, in task 2, one function of the router is to transfer some packets. So we could set listener:

```
PacketListener listener = new PacketListener() {
    @Override
    public void gotPacket(PcapPacket packet) {
        // Transfer some packets
    }
};
```

- For realizing NAT, we considered to save all the mapping into a Hash table, and if we were to transferred the ICMP packet, first we would check the Hash table and then considered the transfer logic.
- How to calculate the RTT is also a thing worth considering, in the calculation of the RTT, we recorded the time of the system and solved in this way: the sender sends the time until the sender receives the ACK sent by the receiver, which is our RTT.

```
long start = System.currentTimeMillis();
// Start sending
// ... some code
// End receiving
long end = System.currentTimeMillis();
// store RTT
RTT = end - start;
```

- Through the Wireshark app, simply setting network interface and filters, we could check whether the packet was sent successfully to check the error of our code: whether the ICMP packet we designed ourselves was right or not.
- As for how to create a same and right packet as the real ICMP packet, We got an idea that when we captured an ICMP packet from the network interface, we could gain some information of the packet like DstIpV4Address, DstEthernetAddress, IcmpV4Identification and so on. So, we could save them and create a same packet (of course some information might need modifying) and send it.

Suggestion :

- The task 1 of project 3 is ICMP, which is a bit lofty to understand what we need to do, it might be better if the document could contain more detail information about what we need to do.

Project 4. Above IP

Stories :

- In this last project, the main thing we were going to implement was the domain name conversion, that is to say, we needed to change the domain name to IP address, which was learned in the course.

Chanllenges :

- For DNS, the main difficulty was how to convert a domain name into an IP address for access.
- For HTTP, we were very confused when reading the content of task 2 and could not have an idea to solve task 2 and also we had little time to try on task 2 because of the final-exam week.

Design && Solutions :

- We just realized a local DNS server. Our design for it was to send a DNS request to the "223.5.5.5" and got the IP address of the related domain name, then we saved the mapping between the domain name and IP address in a local Hash table, which was similar to what we did in the previous project 3 task 3 NAT. For Java, there is a maven repository called `org.xbill`, which helped us to visit the local DNS server like "8.8.8.8" or "223.5.5.5". We learned how to use this library to realize the transfer of a domain name to the IP address.

```
Resolver resolver = new SimpleResolver("223.5.5.5");
Lookup lookup = new Lookup(domain, Type.A);
lookup.setResolver(resolver);
Record[] records = lookup.run(); // get the IP address stored in records
```

- Of course, this operation would send a DNS request to the Internet "223.5.5.5" and get a DNS reply for IP address related to the domain name. The advantage was that the `org.xbill` library encapsulated the DNS packet and we didn't need to create and send a DNS request packet by ourselves.

Suggestion :

- The project 4 might be good enough.

Summary for all 4 Network projects

- Undoubtedly, the all four Network projects challenged us a lot, and urged us to learn more about how the real network works and network coding. Starting from the easy recording and sending audio, finally we implemented a toy network Aethernet, although the discovering process was much more complicated. At last, thanks for hard-working teacher and TA!