Final Report

Group 4
Liu Chengqi and Ren Yijing

Project 1

1.1 Technical Details

1.1.1 Modulation Method

Modulation method is very important, because different methods will result in different implementation, which will completely affect the subsequent projects and is difficult to re implement in subsequent projects. Four basic modulation methods were taught in class: OOK, ASK, FSK and PSK. As indicated in the project requirements, FSK is easy to implement and PSK can reach high throughput. ASK is not recommended. So, we only tested FSK and PSK. Our tests showed that the performance of PSK is better than FSK, so we used PSK for all projects except for OFDM.

1.1.2 Programming Language

Another important technical detail is which language should be chosen to implement the project. The only programming languages we have learned are C++, Python and MATLAB. We have to learn other languages from scratch if we need them. We finally chose Python, because it is more friendly to beginners and there is more information available for Pyaudio on the network. Although in subsequent projects, we found that Python is not a good choice for its poor speed, which makes many parts with time limit quite hard to complete. Fortunately, we persisted. We used many methods to improve the speed of our codes and persisted to the last project with Python.

1.1.3 Number of Sampling Points

Because our sampling rate is 48000 and we need to transmit 10000 bits in 15s, each signal should have no more than 72 sampling points. The header's length should also be considered. In our final implementation, each signal uses 48 sampling points, and each header uses 480. The header can still be shortened, but this is enough for the requirement.

1.1.4 OFDM

OFDM is the most widely used multi carrier transmission scheme with the lowest complexity. That's why we took a long time to implement it. Unfortunately, some folk Chinese materials on the Internet are misleading, which led to our initial implementation completely wrong. OFDM should use IFFT for modulation and FFT for demodulation. Its transmission is based on FSK (not our PSK before), and a signal guard should be inserted between signals. We finally implemented an OFDM system with five carriers (2000Hz, 3000Hz, …, 6000Hz) on the basis of hardware limitation. High frequency carrier will have great distortion, and we also need to ensure that the period of each signal is complete.

1.2 Challenges

1.2.1 Equipment Problems

We have encountered many difficulties in the equipment: different equipment has different performance; sound card needs to be preheated; the same program has different performance in different environments; the same program has different delay in different environments; high

frequency signal has great distortion; various parameters need to be adjusted, etc. Many other groups reflected similar problems, and some even use their own special equipment. We also bought our own devices, but found they didn't perform as well as the original devices with the computer. We carried out a lot of debugging in advance, to ensure the frequency, signal length, header length and the packet length can be quickly adjusted on site. Our method is to use the microphone of one computer's original headphone for recording and put it on the sound of another computer.

1.3 Course improvement

1.3.1 OFDM

We strongly suggest that the investigation of OFDM should be put into the wired transmission part of project 2. Our OFDM program is difficult to achieve the target due to the problems of the equipment itself, but it is easy to perform much better with the support of wired transmission. We think this project should focus on OFDM technology, not debugging skills, so we suggest moving it to project 2.

Project 2

2.1 Technical Details

2.1.1 Parameters for Wired Transmission

The required settings of various parameters for wired transmission are different from those for wireless transmission. Transmission should be as fast and accurate as possible. In our implementation, the frequency is 6000Hz, each signal uses 4 sample points and each header uses 60. However, due to the jamming signal in part 3, this configuration is prone to have error. So, we used frequency of 10000Hz and header length of 120 sampling points for part3. We also used Hamming Code to deal with a small number of errors.

2.1.2 Faster Programming Language

Python runs very slowly, which is related to the library of Pyaudio and the characteristics of the language itself. Other groups using Java also encountered similar problems, although it was more related to the library of audio. We used many methods to accelerate our program, such as using FFT, reducing the cost of function calls, and reducing signal errors. We managed to meet the requirements finally. Many teams that used more basic languages (such as C #, Rust) did not encounter such problems. Therefore, we are interested in how much acceleration the basic languages can achieve.

2.1.3 Collision Detection

Although it is a feature that we haven't implemented yet, we're quite interested in collision detection, because our codes for part 3 is quite slow and collision detection has been taught in class. Our code for part 3 needs about 50 seconds to transmit 6250B, so transmitting 2MB takes about 4 hours and 40 minutes, which is too long to accept. We look forward to a significant acceleration after the implementation of collision detection.

2.2 Challenges

2.2.1 Equipment Problems

Unfortunately, the biggest challenge still comes from the equipment. The sound card needs to be preheated. The volume of one our laptop will automatically gradually decrease in playing, but the other laptop does not have this problem. The recording ends earlier than playing will cause a

serious impact on the playing signal, and lead to some strange errors. Some of these problems may be related to the mismatching of the sound card driver, but the exact reason is still unknown until we finished the project. We just tried hard to avoid these problems.

In addition, after a lot of tests, we found that the performance is better when the power cord is plugged in on wireless transmission, but it is better without power cord on wired transmission. Unfortunately, our one computer's battery is aging, causing the battery to run out quickly and fall into a state that the laptop can't be turned on without the power cord, which made our check hard to complete.

2.2.2 Synchronization of Multithreading

Because our program uses multithreading, many errors caused by synchronization problems are difficult to reproduce, which brings great difficulties to our debugging. In addition, the cache state of the computer can also affect the speed of the program, resulting in the same program running at different speeds. To ensure that the program's effect is consistent every time, we press the Enter key of two computers at the same time, and insert many time.sleep() and print() commands to ensure synchronization.

2.3 Course improvement

2.3.1 Degree of Difficulty

Among the four projects, project 2 is the most difficult one, which even hit the confidence of many students. Considering that all the 5 parts of project 2 are very important, we suggest that some parts of project 2 be released in advance to project 1, or delayed to project 3, in order to average the difficulty of the four projects. The sound card can be provided in project 1 in advance for students to adapt and debug their codes.

2.3.2 More Document Tips

Some parts' requirements of the document should be clearer and more tips should be added in order to be more friendly to the novices on programming or computer network. Although challenging project can train the students to master the characteristics of their chosen programming language, but too many difficulties will also lead to many students give up prematurely.

2.3.3 Grading Rules

The grading rules can be more detailed, so that some students who can only achieve simple functions can get some scores instead of zero.

Project 3

3.1 Technical Details

3.1.1 NAT

Nat implements the function of address translation, mapping multiple internal addresses to a legal public address. This technology is very important for utilizing limited IP address resources. Our NAT is not perfect. Some functions on node 2 can be moved to node 1. Although we have meet the requirement, we are still very interested in building a better NAT for our toy network.

3.1.2 Ping from External Network

The implementation of this technical will help us to realize the toy net, which is a very interesting function. Unfortunately, we didn't have enough time for this optional part.

3.2 Challenges

3.2.1 Poor Quality of Campus Network

The poor quality of campus network causes the same code to run well in some places, but runs terrible in other places, especially some places where network is poor. At first, we thought that the reason came from the error of our codes. However, after a long time of debugging, we found that turning on the hotspot of the mobile phone would eliminate the problems.

3.2.2 Firewall

At first, we tried to add whitelist in the system firewall and firewall applications but it didn't work. Firewall settings in the Windows system are complex, and many details are difficult to find. In the end, we disabled the firewall directly. This measure is simple and crude, but effective.

3.3 Course improvement

We think project 3 is good enough.

Project 4

4.1 Technical Details

4.1.1 FTP

We found some problems with Python's original ftplib, so we made some modifications and added some new features to it. We are very interested in the details of FTP implementation in Python. Unfortunately, there are still many technical details in the official library that we haven’t figure out. Some details in FTP RFC file seems to be not clear enough for us. However, the requirements of project 4 is relatively simple. We do not need to use all the functions of FTP, which brings convenience to our implementation.

4.2 Challenges

4.2.1 Unstable Network

FTP response time is often unstable, sometimes even takes many seconds, and some unexpected network errors can cause a variety of error messages. We need to make sure that our program can cope with all kinds of bad network environment, or it may fail on TA’s check. We try to ensure the synchronization by extending the recording time and adjusting the interval between related commands. But this method still can't cope with all the situations. Some large file downloads, or some complex commands written by us, may run many times as long as other commands. In view of these special circumstances, we decided to let the program send the signal for completing the task, and wait for the next command.

4.2.2 FTP RFC File

The FTP RFC file is so long that it is difficult for us to quickly find the technical information we need. We looked up a lot of information as a supplement, and even found an official version of the Chinese translation edition. These greatly help us to understand FTP and complete the Part 1.

4.2.3 Poor Underlying Code

We found that the underlying code we wrote before was not good enough, which affect the implementation of project 4. Therefore, we first modified the underlying code to make its encapsulation clearer and easier to use. We also achieved a better implementation of variable length transmission. However, because the part 1 was implemented too hastily, we didn't have enough time to implement a pure NAT for part 2. It is such a pity that we didn't finish it until the end of the course.

4.3 Course improvement

We think project 4 is good enough.