Computer Networks: Final Project Report

Due on Jan. 10^{th}, 2021 at 11:59am

Professor Zhice Yang

Abstract

This document will describe out implementation of 4 course projects, which contains 6 sections. The first 4 part contains the four projects and the last 2 part contains the feedback and the possible refinement.

Project 1

Physical Layer

Sound card and Target data line.

For the first project we use the amodem, which is a good sound API for python to finish the check. We utilize the java sound API to get access to the low level sound devices. The abstract up layer for programers is dataline, which is a byte array stream. Once we fill over the stream, the buffer stream will be piped through the sound devices with considerably short amount of latency time around 15 ms. The latency provides us with possibility to even more reduce the time.
We found that the OS scheduler may take low IO devices for the IO device has low priority and take long time to transfer. The lock acqure and release may take 500 more ms when transmitting 10000 bits. Another problem is the sound card will keep replaying the content in the buffer if no new data is offered.

Dataframe

We defined the data frame from the begin of data, we respectively add lens, crc8 and data. The project2 is similar to this design. Because we have the maximum packets of 256 bytes for the project1, so that we only need 8 bits for that. And intuively the CRC8 utilize 4 bits and if the length check is not passed and the CRC is not possible to pass.

Modulation and Demodulation

We apply the preamble from the matlab source code. First, we have to define wether the signal exists or not, this can be done by the real time power which will be discussed later on. Then let's discussed about the signature sine function:

$$\omega_k = \omega_{k-1} + \frac{f_k + f_{k-1}}{2 \times sr} \quad f_k = h - |k - \frac{1}{2}n| \frac{h-l}{n} \quad d_k = sin(2\pi\omega_k)$$

The preemble is suitable for PSK, for its dot product is small or it's perfectly aligned with the signal. PSK said the bit 0 and 1 corresponds to $cos(2\pi f)$ and $cos(2\pi f + \pi)$. The result of the advance of π make it pefect to let the data be modulated by multiplying by -1. The demodulation process is on opposite. The sampling rate is 480kHz, thus there's only 24 sample for the 2kbps bandwidth. This selection proves to be the best practice than QPSK or other parameter of sample rate or intervals.

OOP implementation

To utilize what we've learnt at CS100, we deeply apply the thoughts "High coupling, low cohesion". We devided the projects as soundAPI receiver transmitter and packets defininition. This refinement of code let us can take each other's code as black box to follow our own thoughts to make physical layer happen. Also, the parameter like the sample rate and volume of the computer and dataline buffer size can be easily treaked. We also found the inheritence which the packets can inherit to the macpackets and layer can inherit to the mac layer useful, but we failed to make it happen.

Project 2

Mac Layer

Mac Packets

The most important unit for the mac layer is the definition of the packets. which contains the ID for 8 bits, src and dest mac address for 2 bits each, and 4 bits type and data for the following up. We are delighted to find that the FSM learnt at CS131: Compilers are important. Because the state matchine is another important part. To convert the Packets to byte array in java is to utilizing the shift operation with deep coppy.

Mac Layer FSM

We should manifest the essense of Networks that is request and ack. The process of sending a packets in mac layer is to request the packets first and wait for frame detection and wait for the detection. All the state is hard coded in the type and handled by the macLayer class. For sending the packets. The ID will be

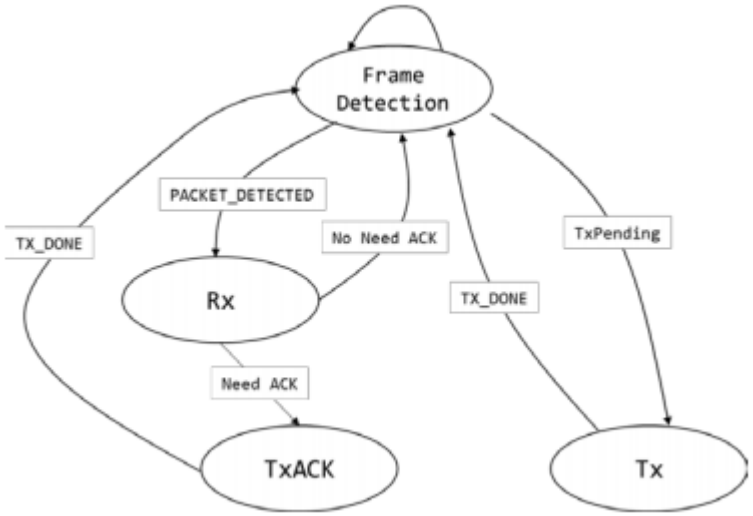

Figure 1: Mac Layer FSM

assigned, this is labeled one by one, which the ID is not possible to be same. The same ID in the ACK will automatically be lost when reply is lost. The fact that we reply ACK first may lead to normal data being chocked if we are receiving enormous packets and busy replying ACKs.
The timestamp and the counter will be stored in the member class. The iteration loop will be maintained when the user calls stop. Every iteration will be check the waiting loop and update the state every time.
The link error will be generated when we have resend it too much, we will report a link error.
The receive part is to receive from the dataqueue and reply with an ACK for normal circumstances. A patch we added for MACPING is to either reply that MACPING or calculate the time eclipsed.

Thread Management

We just start the maclayer by calling **MacLayer.startMacLayer()** to start them and use **MacLayer.stopMacLayer** to stop the threads.

The large file is transmitted by deviding them into different pieces and sequentially arrival data receive. We've maintained a datadeque utilizing the java nonblockconcurrent queue.

Signal Detection

We utilize the matlab code. We only send the packets with ACK if the power is greater than the following equation. The equaiton is calculated every sound transmission.

$$avg_k = \frac{win\_size - 1}{win\_size} avg_{k-1} + \frac{1}{win\_size} * power_k > thr \tag{1}$$

Project 3

Gateway

Scapy

We utilize a very good python packets sniff and resending tool for project3 and project4.
It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, store or read them using pcap files, match requests and replies, and much more. It is designed to allow fast packet prototyping by using default values that work.
It can easily handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks or network discovery (it can replace hping, 85% of nmap, arpspoof, arp-sk, arping, tcpdump, wireshark, p0f, etc.). It also performs very well at a lot of other specific tasks that most other tools can't handle, like sending invalid frames, injecting your own 802.11 frames, combining techniques (VLAN hopping+ARP cache poisoning, VoIP decoding on WEP protected channel, ...), etc.

Pcap

Another good tools for network universal packets transmission is pcap. For windows, we install winpcap. The pcap file is easy for mac layer transmission and good for modification like NAT operation.

ICMP

For ICMP, we first wrap the packets with additional informaiton like src port, destination port and informaiton in the payloads. in the node1 and then make the pcap. Then we transmit it to node2, node2 modify the ip header using Scapy.
For ICMP server, we have to disable the machine ICMP first and do literally the same thing as above.

Project 4

FTP Client

FTPlib

Python ftplib is a module that implements the client side of the FTP protocol. It contains an FTP client class and some helper functions.

Scapy Again

We translate all the FTP command to the ftplib command. Then we try to make all the TCP connection sniffed by the Scapy module and wrap them into the pcap file. The underlying OSI layer transmit the pcap file and the node 2 just modify the src address and send it out. For receiving the ACK, we let the node2 capture the pcap to the node1 and node1 make the response. All the operation are done using the middleware of our maclayer and physical layer.

Feedback

The lecture is unique to Tsinghua tech guy, with full of fun and hardcore. Personally, I don't favor the project a lot for most of the time I was debugging the physical layer. Maybe Tsinghua's Router Lab to hand write a OSPF algorithm using verilog is more practical During my intern at tech firm, most of the operation in terms of network is tcpdump, ebpf and pcap packets reordering, no other part like bit torrent or physical Signal Processing.
I think the project is best recommended with Csharp, C/C++ or Rust for the implementation of project 2 with full duply. The latency is too high even for java. We implemented the sliding window to hack over it. But not other people did it right.

Future Suggestion

I likes the lecture in the way it is. But I think we can learn more by having take-home quizes and have a 5 minutes quick discussion about that before each lecture. The quiz doesn't necessarily contribute to the grading, they are just some simple questions covering the lecture knowledge. In this way, we get to participate more and you can know if we understands something. Furthermore, you skipped a lot about applications, I understand it's because of limited time. I want to propose a new way for us to understand most of applications. That is to ask students to prepare presentations and talk about them as a bonus. We have to admit that making presentations is also an important skill. This presentation can be set at any time when everyone is available.