Project 1

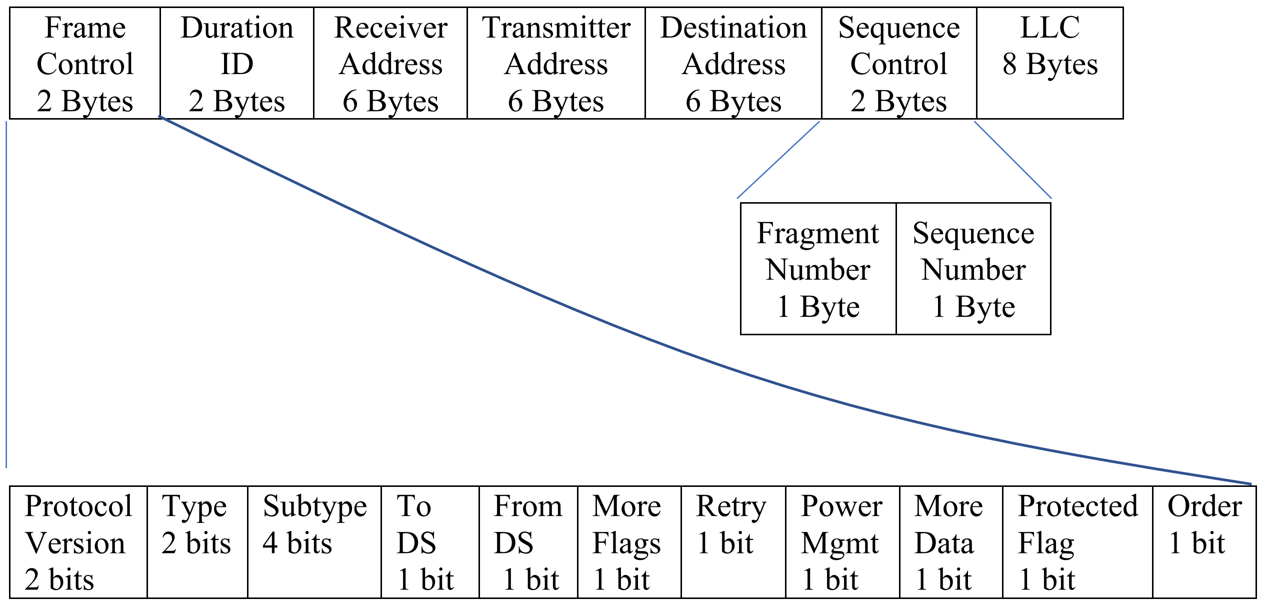
CS 4133/5133 – Data Networks – Due September 20, 2021, 11:59 PM

Background

The project will require you to parse binary data containing Ethernet frames that follow the IEEE 802.11 standard. By performing the parsing, you will learn how the frames and packets are injected into the network. Additionally, you will write a client and server programs that communicates using UDP sockets to send the Ethernet frames generated.

Project Description

A single IEEE 802.11 frame is shown below.



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                Example Internet Datagram Header
```

After you read (the client program does this) each frame from an input file (the input will contain several frames next to each other without any demarcation – you need to count the bytes and read the next frame) the client program sends the bytes to a server program a single frame using the UDP socket communication, the server upon receiving the bytes extracts the contents and prints as below:

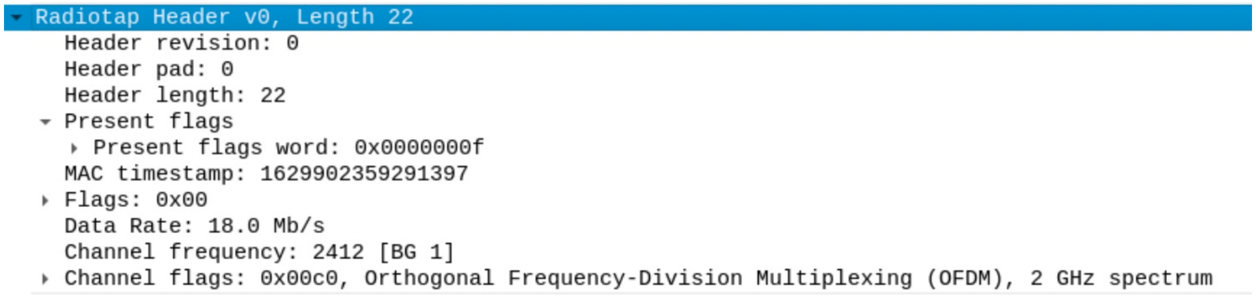
Use pcap_next_ex to read one frame at a time. Do not use pcap_dispatch or pcap_loop, as they internally execute all the frames at once, defeating the purpose of client server communication.

First, download Wireshark app onto your local desktop, the application is available for Windows, MacOS, and Linux.

Set the filter in the wireshark to "icmp", then load the pcap file provided for visualizing the frames.

For your C program:

The first 22 bytes of each frame contains Radiotop Header, which contains the following information

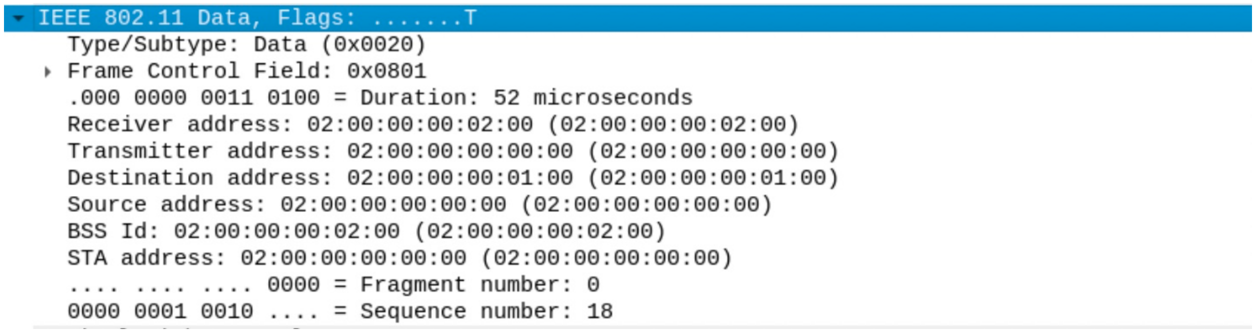


```
▾ Radiotap Header v0, Length 22
    Header revision: 0
    Header pad: 0
    Header length: 22
  ▾ Present flags
    ▸ Present flags word: 0x0000000f
    MAC timestamp: 1629902359291397
  ▸ Flags: 0x00
    Data Rate: 18.0 Mb/s
    Channel frequency: 2412 [BG 1]
  ▸ Channel flags: 0x00c0, Orthogonal Frequency-Division Multiplexing (OFDM), 2 GHz spectrum
```

Note: The contents of the field might change depending up on the packet.

```
Radiotop: -----Radiotop Header----
     Radiotop:
     Radiotop: Header revision = 0
     Radiotop: Header pad = 0
     Radiotop: Header length = 22
     Radiotop: Present flags word = f
     Radiotop: Mac Timestamp = 1629902359291397
     Radiotop: Flag = 0
     Radiotop: Data Rate = 18.0 Mb/s
     Radiotop: Channel frequency = 2412
     Radiotop: Channel flags = c0
```

Now read the next 24 bytes of 802.11 Header data:

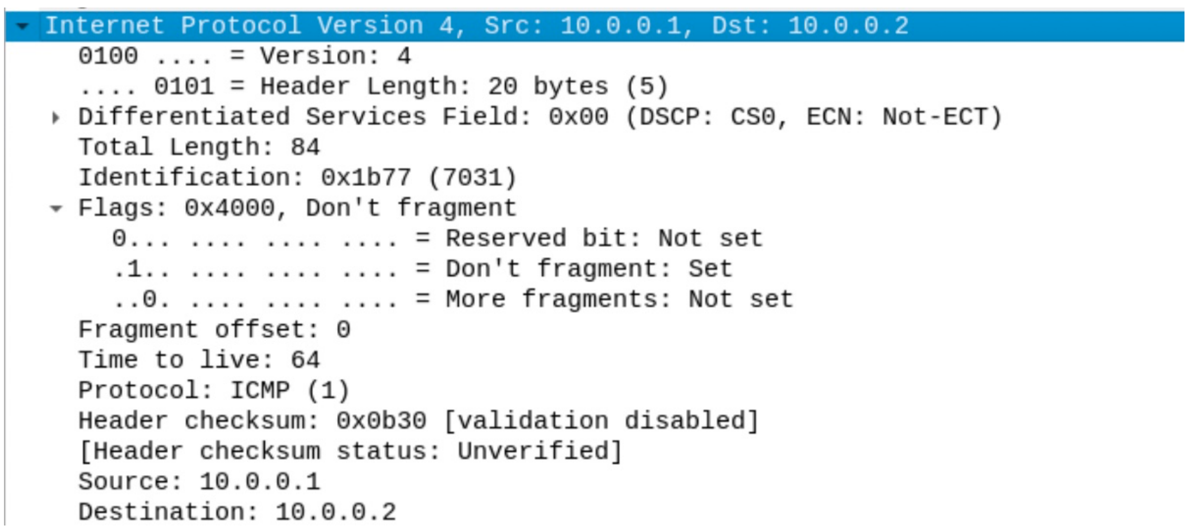


```
▾ IEEE 802.11 Data, Flags: .......T
    Type/Subtype: Data (0x0020)
  ▸ Frame Control Field: 0x0801
    .000 0000 0011 0100 = Duration: 52 microseconds
    Receiver address: 02:00:00:00:02:00 (02:00:00:00:02:00)
    Transmitter address: 02:00:00:00:00:00 (02:00:00:00:00:00)
    Destination address: 02:00:00:00:01:00 (02:00:00:00:01:00)
    Source address: 02:00:00:00:00:00 (02:00:00:00:00:00)
    BSS Id: 02:00:00:00:02:00 (02:00:00:00:02:00)
    STA address: 02:00:00:00:00:00 (02:00:00:00:00:00)
    .... .... .... 0000 = Fragment number: 0
    0000 0001 0010 .... = Sequence number: 18
```

```
Header: ----- 802.11 Header -----
     Header:
     Header: Frame Control Field = 0x0801
     Header: Duration = 52 microseconds
     Header: Receiver Address = 02:00:00:00:02:00
     Header: Transmitter Address = 02:00:00:00:00:00
     Header: Fragment number = 120
```

Next, 8 bytes of the pcap data is about Logical Link Control, which need not be processed.

Coming on to the IPv4 Header, you will have to read 20 bytes of data, and print the necessary information.



```
▾ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x1b77 (7031)
  ▾ Flags: 0x4000, Don't fragment
      0... .... .... .... = Reserved bit: Not set
      .1.. .... .... .... = Don't fragment: Set
      ..0. .... .... .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0x0b30 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.0.1
    Destination: 10.0.0.2
```

```
IP: ----- IP Header -----
     IP:
     IP: Version = 4
     IP: Header length = 20 bytes
     IP: Total length = 84
     IP: Flags = 0x4
     IP:          0... .... .... .... = Reserved bit: Not set
     IP:          .1.. .... .... .... = Don’t fragment: Set
     IP:          ..0. .... .... .... = More fragments: Not set
     IP: Fragment offset = 0 bytes
     IP: Time to live = 64 seconds/hop
     IP: Protocol = 1 (ICMP)
     IP: Header checksum = b30
     IP: Source address = 10.0.0.1
     IP: Destination address = 10.0.0.2
```

Rest of the bytes are the data. Print them as shown below.
ICMP: ----- Packet Data ----

```
0000 08 00 20 01 3d 94 08 00 69 01 5f 0e 08 00 45 00 .......% /!@...E.
0010 00 40 63 1c 40 00 64 00 06 05 0a 80 80 08 01 07 .@....d. ........
0020 ff ff 0b 2c 76 5f 00 2c 00 00 54 45 44 35 30 30 ...,v_., ..TED500
0030 30 20 20 20 20 20 20 20 20 7c 38 30 7c 30 30 2d 0....... |80|00-
0040 32 35 2d 32 46 2d 32 31 2d 34 30 2d 31 32 25 00 -2F-21.. -40-12..
```

Break Down of a Packet

```
0000    00 00 16 00 0f 00 00 00 05 e6 24 37 63 ca 05 00
0010    00 24 6c 09 c0 00 08 01 34 00 02 00 00 00 02 00
0020    02 00 00 00 00 00 02 00 00 00 01 00 20 01 aa aa
0030    03 00 00 00 08 00 45 00 00 54 1b 77 40 00 40 01
```

```
0040   0b 30 0a 00 00 01 0a 00 00 02 08 00 ba c9 10 39
0050   00 01 17 56 26 61 00 00 00 00 2c 72 04 00 00 00
0060   00 00 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d
0070   1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d
0080   2e 2f 30 31 32 33 34 35 36 37
```

Radiotop Header, 802.11 Header, Logical-link control (LLC), IP Header, Packet Data.

Each number is a byte, and all the bytes are in hexadecimal.

In the output above, the non-printable characters are printed with a "." (a single period).

As part of the project, while not necessary, I recommend that you use the headers below if you are writing this in C. It is noted that there are several programs on the web (c or c++) code that allows you to parse the pcap files.You are allowed to use them as long as you provide acknowledgement in the comment section at the beginning of the program.

```
#include <stdio.h>
#include <stdlib.h>
#include <pcap.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/ether.h> /* to parse Ethernet headers. */
#include <netinet/ip.h> /* to parse IP headers. */
#include <netinet/tcp.h> /* to parse TCP headers. */
```

You will have a single file containing several frames. After processing each frame, you will print the frame contents as described above.

Constraints:

1. The project is due on DATE by 11:59pm. Source files should be submitted to the canvas on the assignment page.
2. This project will be done individually. Consultation in any form with other members of the class is strictly prohibited.
3. Your program should be documented thoroughly and **20%** of the project grade will be for documentation.
4. You are required to demonstrate the functionalities of your programs for grading. The schedule will be announced later. The demonstration will take about 5 minutes.
5. Test file will be provided which contains 10 frames. During the demonstration, we will provide another test file, for which your program should produce appropriate results.