

C Language Programming: Homework #7
Assigned on 12/04/2018(Tuesday), Due on 12/18/2018(Tuesday)

Description:

This assignment allows you to practice processing packets stored in a file. You are required to do:

1. Use command line to input the filename
2. (70%) Read the packets in the input file, and according to the following pictures, you need to parse the packet fields. The fields include DST MAC, SRC MAC, protocol, SRC IP, DST IP, SRC Port, DST Port.(ICMP packet doesn't need Src Port, Dst Port) and count the Packet Length for each packets.
3. (10%) Count total number of packets, number of TCP packets, number of UDP packets, number of ICMP packets in the file
4. (20%) Report with right format should be both printed out and updated on server.

Figure 1 : packet layer

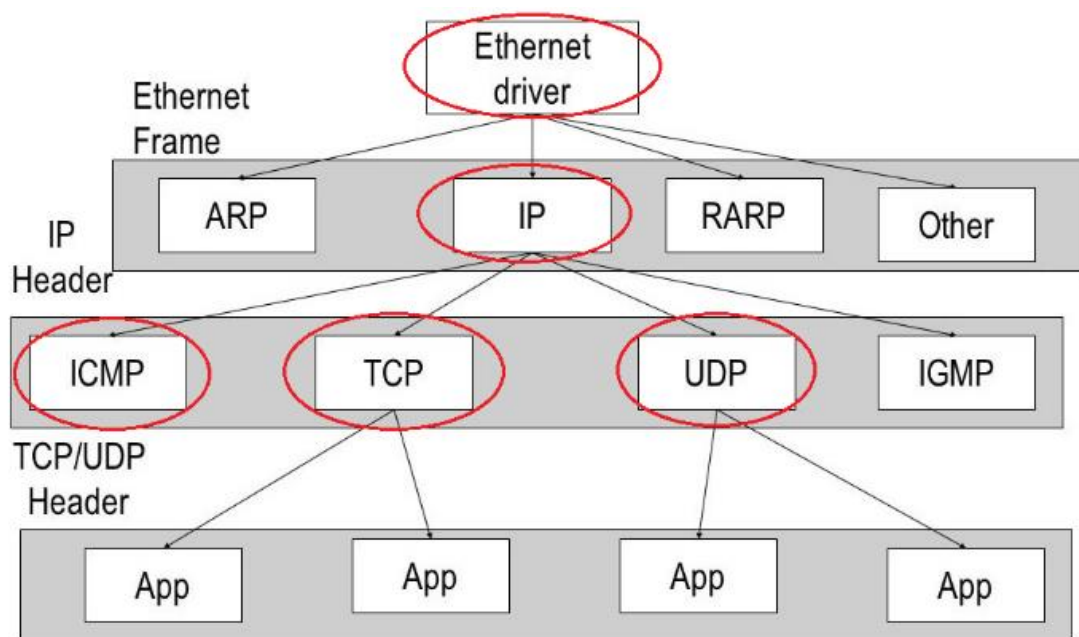
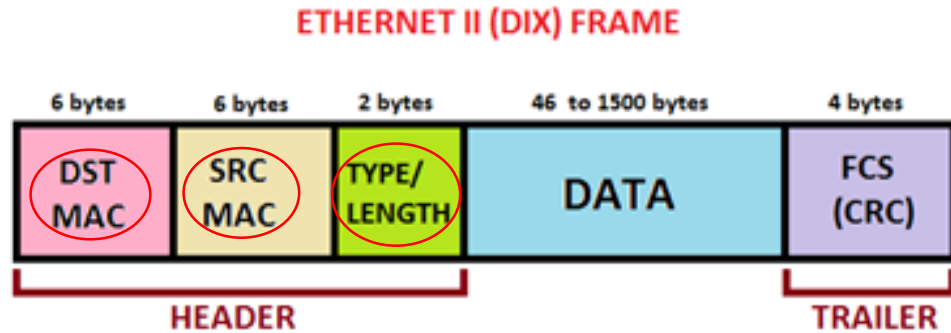
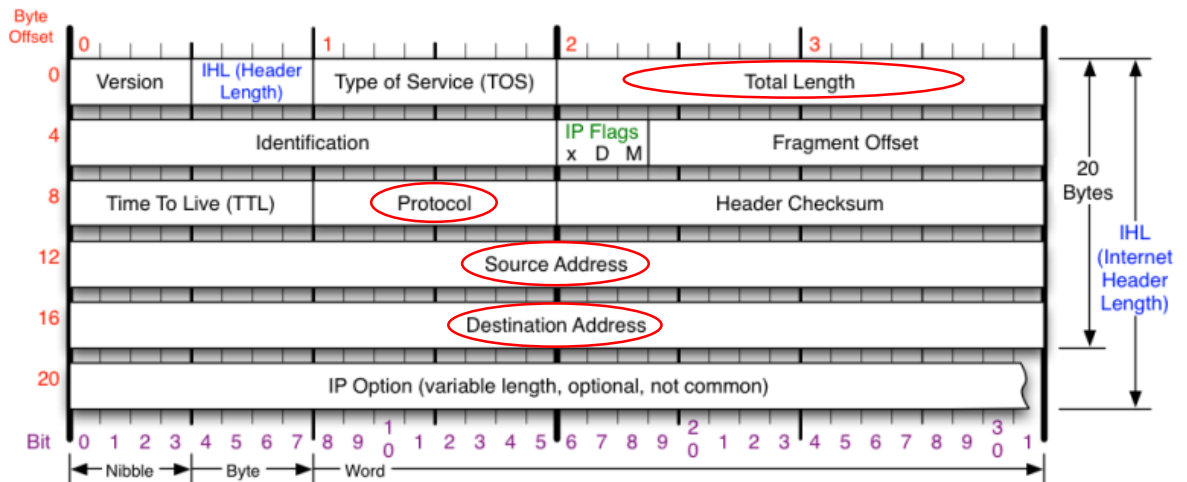


Figure 2 : Ethernet format



Filed TYPE: IPV4(0x0800)

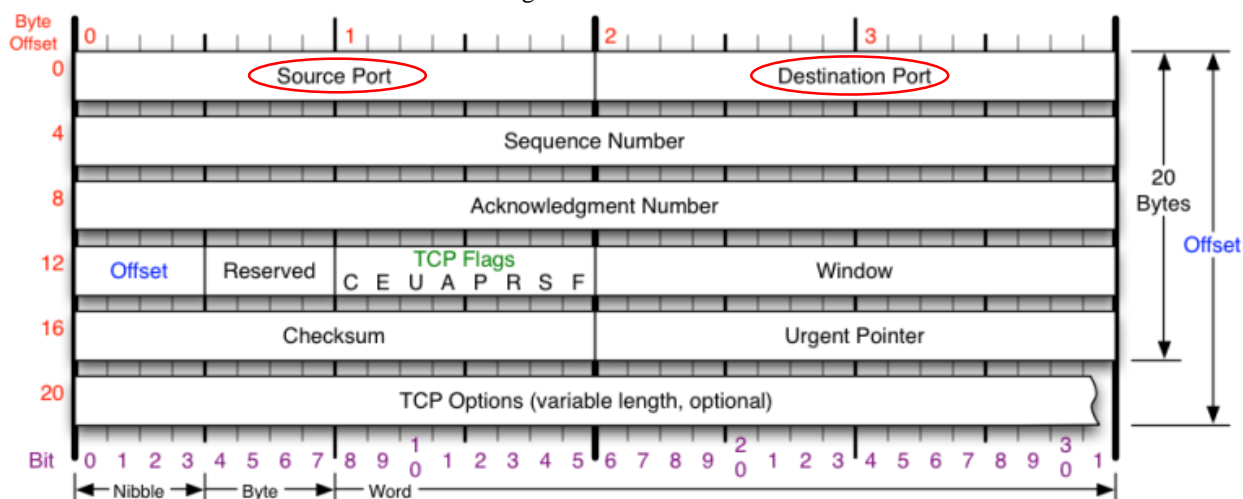
Figure 3 : IPV4 Format



Version Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.	Protocol IP Protocol ID. Including (but not limited to): 1 ICMP 17 UDP 57 SKIP 2 IGMP 47 GRE 88 EIGRP 6 TCP 50 ESP 89 OSPF 9 IGRP 51 AH 115 L2TP	Fragment Offset Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.	IP Flags x D M x 0x80 reserved (evil bit) D 0x40 Do Not Fragment M 0x20 More Fragments follow
Header Length Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.	Total Length Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.	Header Checksum Checksum of entire IP header	RFC 791 Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

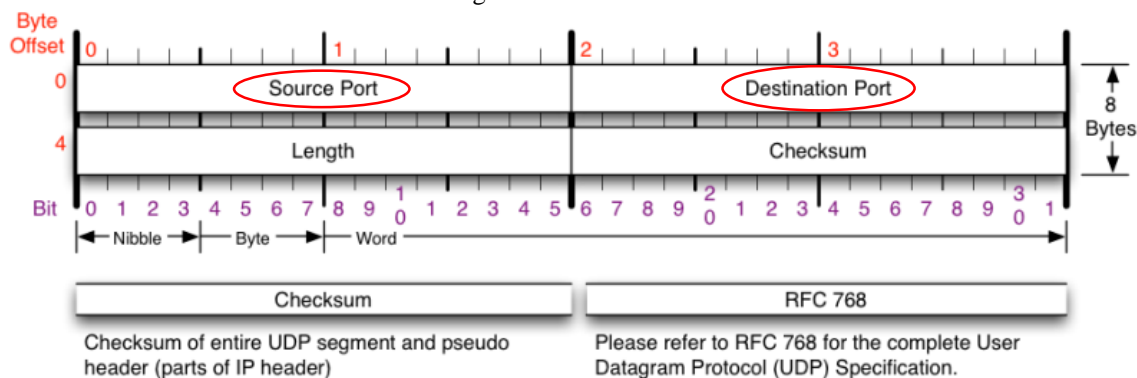
Field: Total Lengh+14 bytes=Packet Bytes and Protocol: TCP(0x06), UDP(0x11), ICMP(0x01)

Figure 4 TCP format



TCP Flags	Congestion Notification	TCP Options	Offset																											
C E U A P R S F	ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.	0 End of Options List 1 No Operation (NOP, Pad) 2 Maximum segment size 3 Window Scale 4 Selective ACK ok 8 Timestamp	Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.																											
Congestion Window C 0x80 Reduced (CWR) E 0x40 ECN Echo (ECE) U 0x20 Urgent A 0x10 Ack P 0x08 Push R 0x04 Reset S 0x02 Syn F 0x01 Fin	<table><tr><td>Packet State</td><td>DSB</td><td>ECN bits</td></tr><tr><td>Syn</td><td>0 0</td><td>1 1</td></tr><tr><td>Syn-Ack</td><td>0 0</td><td>0 1</td></tr><tr><td>Ack</td><td>0 1</td><td>0 0</td></tr><tr><td>No Congestion</td><td>0 1</td><td>0 0</td></tr><tr><td>No Congestion</td><td>1 0</td><td>0 0</td></tr><tr><td>Congestion</td><td>1 1</td><td>0 0</td></tr><tr><td>Receiver Response</td><td>1 1</td><td>0 1</td></tr><tr><td>Sender Response</td><td>1 1</td><td>1 1</td></tr></table>	Packet State	DSB	ECN bits	Syn	0 0	1 1	Syn-Ack	0 0	0 1	Ack	0 1	0 0	No Congestion	0 1	0 0	No Congestion	1 0	0 0	Congestion	1 1	0 0	Receiver Response	1 1	0 1	Sender Response	1 1	1 1	Checksum Checksum of entire TCP segment and pseudo header (parts of IP header)	RFC 793 Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.
Packet State	DSB	ECN bits																												
Syn	0 0	1 1																												
Syn-Ack	0 0	0 1																												
Ack	0 1	0 0																												
No Congestion	0 1	0 0																												
No Congestion	1 0	0 0																												
Congestion	1 1	0 0																												
Receiver Response	1 1	0 1																												
Sender Response	1 1	1 1																												

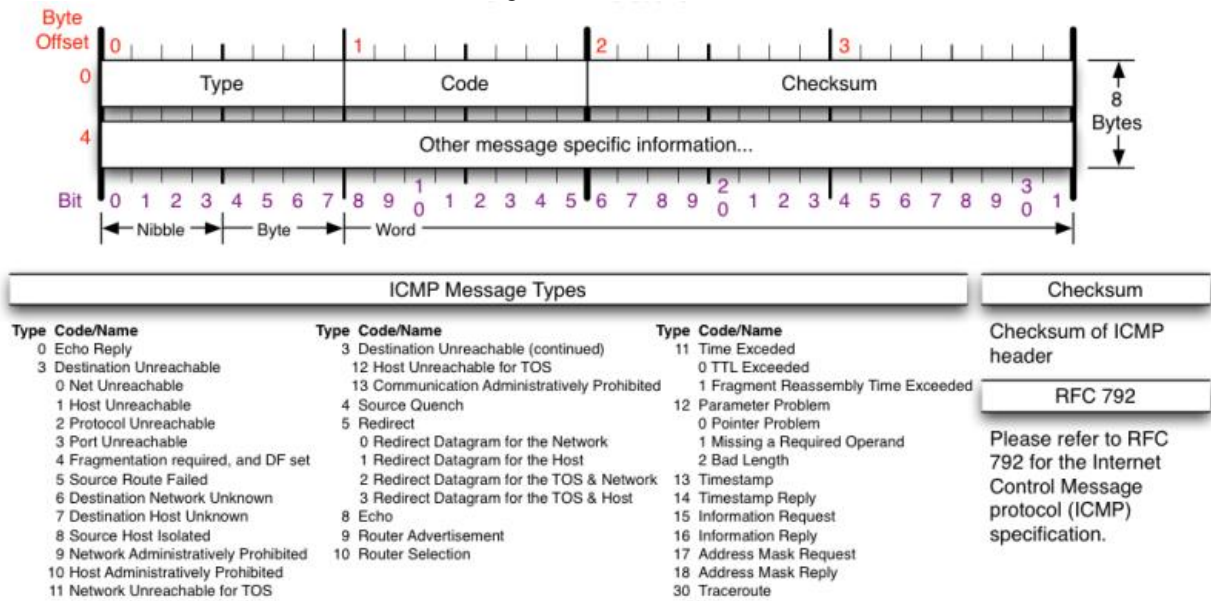
Figure 5 : UDP format



Checksum of entire UDP segment and pseudo header (parts of IP header)

Please refer to RFC 768 for the complete User Datagram Protocol (UDP) Specification.

Figure 6 : ICMP format



e.g. A TCP packet of 66 bytes shown in **hexadecimal form**.

	Ethernet	Protocol : TCP	TYPE : IPV4	
0000	00 08 9b c4 61 fc 08 62 66 50 b8 4e 08 00 45 00			
0010	00 34 2a 2b 40 00 80 06 00 00 8c 74 52 a5 8c 74			
0020	52 ab f8 89 00 50 b2 39 1d 13 00 00 00 00 80 02			
0030	fa f0 be 5f 00 00 02 04 05 b4 01 03 03 08 01 01			
0040	04 02			
	Total Length+14 bytes=66 bytes			

(NOTE: You are asked to make use of “**struct**” to save the fields of Ethernet, IPV4, TCP, UDP, ICMP.)

Remark

Attached file *hw7_sample.c* shows how to use **fread** to parse SRC MAC, SRC IP and SRC Port in *input.out*, which is a **binary file**.

```

xts26951haya@CIAL-PROG6:~$ ./hw7_sample input.out
SRC MAC: 1a:2b:3c:4d:5f:66
SRC IP: 192.168.0.1
SRC Port: 65502

```

Moreover, if you want to check the content of a binary file, you can use **xxd -b filename** shown as follows.

```

xts26951haya@CIAL-PROG6:~$ xxd -b input.out
00000000: 00011010 00101011 00111100 01001101 01011111 01100110  .+<M_f
00000006: 11000000 10101000 00000000 00000001 11111111 11011110  .....

```

(NOTE: The actual content is 0001101000101011.....11011110, spaces and other marks can be ignored.)

Similarly, you can check it in **hexadecimal form** with **xxd filename**.

Command Line:

```
./hw7 test.out
```

(Don't use other names except **test.out**, which is also a binary file.)

Example Output :

```
#1                                // the 1st packet
```

```
DST MAC: 00:08:9b:c4:61:fc
```

```
SRC MAC: 08:62:66:50:b8:4e
```

```
Protocol: TCP
```

```
SRC IP: 140.116.82.165
```

```
DST IP: 140.116.82.171
```

```
SRC Port: 63625
```

```
DST Port: 80
```

```
Packet Length: 66
```

```
#2
```

```
.....
```

```
#3
```

```
.....
```

```
#n
```

```
.....
```

```
Number of Packet: n
```

```
Number of TCP Packet: ?
```

```
Number of UDP Packet: ?
```

```
Number of ICMP Packet: ?
```

(You should follow above format as output **printed on screen**.)