

Junhyeok Jeong

Bill Pfeil

CS 372-X001

11/24/2019

Lab 4

1. Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?

The screenshot shows a Wireshark packet capture of ICMP Echo (ping) messages. The packet list shows several requests and replies. The first request is selected, and the packet details window is expanded to show the Internet Protocol Version 4 section. The IP address 192.168.0.21 is highlighted in the packet details window.

No.	Time	Source	Destination	Protocol	Length	Info
529	18:56:01.418110	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2462
530	18:56:01.456094	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2463
531	18:56:01.456274	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exceeded)
537	18:56:01.496055	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2464
538	18:56:01.503886	96.120.61.141	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exceeded)
540	18:56:01.520068	128.119.245.12	192.168.0.21	ICMP	70	Echo (ping) reply id=0x0001, seq=2462
543	18:56:01.534053	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2465
544	18:56:01.542541	68.87.219.25	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exceeded)
545	18:56:01.573659	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2466
546	18:56:01.582259	68.87.222.129	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exceeded)

Frame 529: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: AsustekC_91:eb:e8 (04:d4:c4:91:eb:e8), Dst: Motorola_f5:79:b1 (88:b4:a6:f5:79:b1)
v Internet Protocol Version 4, Src: 192.168.0.21, Dst: 128.119.245.12
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 56
 Identification: 0x6ce6 (27878)
 > Flags: 0x0000
 Time to live: 255

0000 88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00 ...y... ..E.
0010 00 38 6c e6 00 00 ff 01 00 00 c0 a8 00 15 80 77 .81.....w
0020 f5 0c 08 00 2c 9f 00 01 09 9e 20 20 20 20 20 ...,...
0030 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040 20 20 20 20 20 20

Internet Protocol Version 4 (ip), 20 bytes Packets: 3412 · Displayed: 1400 (41.0%) · Dropped: 0 (0.0%) Profile: Default

- My computer's IP address is 192.168.0.21

2. Within the IP packet header, what is the value in the upper layer protocol field?

The image shows a Wireshark packet capture window titled "*Ethernet". The packet list on the left shows several ICMP packets. The selected packet is packet 531, which is an ICMP Echo (ping) request from 192.168.0.21 to 128.119.245.12. The packet details pane on the right shows the following information:

- Total Length: 56
- Identification: 0x6ce6 (27878)
- Flags: 0x0000
- Time to live: 255
- Protocol: ICMP (1)
- Header checksum: 0x0000 [validation disabled]
- [Header checksum status: Unverified]
- Source: 192.168.0.21
- Destination: 128.119.245.12

The packet bytes pane at the bottom shows the raw data of the packet. The first 20 bytes are highlighted in blue, corresponding to the IP header. The 21st byte is highlighted in blue, corresponding to the ICMP header. The 22nd byte is highlighted in blue, corresponding to the ICMP Echo request data.

- Within the IP packet header, the value in the upper layer protocol field is ICMP (1)

3. How many bytes are in the IP header? How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes.

Wireshark interface showing a packet capture on the 'icmp' filter. The packet list displays several ICMP Echo (ping) requests and replies. The selected packet (No. 544) is an ICMP Echo (ping) request from 192.168.0.21 to 128.119.245.12, with a length of 70 bytes. The packet details pane shows the following structure:

- Frame 529: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
- Ethernet II, Src: AsustekC_91:eb:e8 (04:d4:c4:91:eb:e8), Dst: Motorola_f5:79:b1 (88:b4:a6:f5:79:b1)
- Internet Protocol Version 4, Src: 192.168.0.21, Dst: 128.119.245.12
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 56
 - Identification: 0x6ce6 (27878)
 - Flags: 0x0000
 - 0... = Reserved bit: Not set

The packet bytes pane shows the raw data of the packet, with the IP header (20 bytes) and ICMP payload (36 bytes) highlighted in blue.

Header Length (ip_hdr_len), 1 byte | Packets: 3412 · Displayed: 1400 (41.0%) · Dropped: 0 (0.0%) | Profile: Default

- The IP header length is 20 bytes and total length is 56 bytes because the payload length can be determined 56 bytes – 20 bytes = 36 bytes

4. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

*Ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	Time	Source	Destination	Protocol	Length	Info
529	18:56:01.418110	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2462,
530	18:56:01.456094	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2463,
531	18:56:01.456274	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce
537	18:56:01.496055	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2464,
538	18:56:01.503886	96.120.61.141	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce
540	18:56:01.520068	128.119.245.12	192.168.0.21	ICMP	70	Echo (ping) reply id=0x0001, seq=2462,
543	18:56:01.534053	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2465,
544	18:56:01.542541	68.87.219.25	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce
545	18:56:01.573659	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2466,
546	18:56:01.582259	68.87.222.129	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce

Total Length: 56
Identification: 0x6ce6 (27878)
Flags: 0x0000
0... .. = Reserved bit: Not set
.0.. .. = Don't fragment: Not set
..0. = More fragments: Not set
...0 0000 0000 0000 = Fragment offset: 0
Time to live: 255
Protocol: ICMP (1)
Header checksum: 0x0000 [validation disabled]

```

0000 88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00 ...y... ..E-
0010 00 38 6c e6 00 00 ff 01 00 00 c0 a8 00 15 80 77 .81..... ..w
0020 f5 0c 08 00 2c 9f 00 01 09 9e 20 20 20 20 20 ....,....
0030 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040 20 20 20 20 20 20

```

Fragment offset (13 bits) (ip.frag_offset), 2 bytes | Packets: 3412 · Displayed: 1400 (41.0%) · Dropped: 0 (0.0%) | Profile: Default

- In the flags, more fragments flag has set to 0 bit and fragment offset is 0. Therefore, the data is not fragmented.

5. Which fields in the IP datagram always change from one datagram to the next within this series of ICMP messages sent by your computer?

***Ethernet**

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	Time	Source	Destination	Protocol	Length	Info
555	18:56:01.767853	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2471,
552	18:56:01.728033	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2470,
550	18:56:01.689893	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2469,
549	18:56:01.650794	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2468,
547	18:56:01.612618	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2467,
545	18:56:01.573659	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2466,
543	18:56:01.534053	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2465,
537	18:56:01.496055	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2464,
530	18:56:01.456094	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2463,
529	18:56:01.418110	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2462,

Total Length: 56
Identification: 0x6ce6 (27878)
> Flags: 0x0000
Time to live: 255
Protocol: ICMP (1)
Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.0.21
Destination: 128.119.245.12
Internet Control Message Protocol

0000 88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00y... ..E-
0010 00 38 6c e6 00 00 ff 01 00 00 c0 a8 00 15 80 77 ..81..... ..w
0020 f5 0c 08 00 2c 9f 00 01 09 9e 20 20 20 20 20 20,.....
0030 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
Identification (p.id), 2 bytes

Packets: 3412 · Displayed: 1400 (41.0%) · Dropped: 0 (0.0%) Profile: Default

Wireshark capture of ICMP Echo (ping) requests. The packet list shows multiple requests from 192.168.0.21 to 128.119.245.12 with varying sequence numbers and identification values. The packet details pane shows the structure of an ICMP Echo request, including the identification field (0x6ce7) and time to live (1). The packet bytes pane shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
555	18:56:01.767853	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2471
552	18:56:01.728033	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2470
550	18:56:01.689893	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2469
549	18:56:01.650794	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2468
547	18:56:01.612618	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2467
545	18:56:01.573659	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2466
543	18:56:01.534053	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2465
537	18:56:01.496055	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2464
530	18:56:01.456094	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2463
529	18:56:01.418110	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=2462

Total Length: 56
 Identification: 0x6ce7 (27879)
 > Flags: 0x0000
 > Time to live: 1
 Protocol: ICMP (1)
 Header checksum: 0x0000 [validation disabled]
 [Header checksum status: Unverified]
 Source: 192.168.0.21
 Destination: 128.119.245.12
 Internet Control Message Protocol

0000 88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00 ...y... ..E-
 0010 00 38 6c e7 00 00 01 01 00 00 c0 a8 00 15 80 77 ..81.....w
 0020 f5 0c 08 00 2c 9e 00 01 09 9f 20 20 20 20 20 20,...
 0030 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
 0040 20 20 20 20 20 20

Identification (ip.id), 2 bytes | Packets: 3412 · Displayed: 1400 (41.0%) · Dropped: 0 (0.0%) | Profile: Default

- The identification (0x6ce6 -> 0x6ce7) and Time to live (TTL 255 -> 1) are always changed. However, the header checksum should be always changed too since the TTL are decremented. In my wireshark captures, it seemed never changing.

6. Which fields stay constant? Which of the fields must stay constant? Which fields must change? Why?

- The constantly stayed fields are:

- IP version (using IPv4 for all packets)
- Header length (20bytes as ICMP protocol packets)
- Differentiated Services Field (0x00 (DSCP: CS0, ECN: Not-ECT), all packets are using same service class type)
- Upper layer protocol (ICMP (1), all packets are ICMP protocol packets)
- Source IP (192.168.0.21, sent from my computer)
- Destination IP (128.119.245.12, received server)

The version, header length, differentiated services field, upper layer protocol, source and destination IP must be stayed constant because all packets are from same source IP address and same type of protocol (ICMP).

- The fields must be changed are Identification, Time to live (TTL), and Header checksum because IP packets must have different ids and there is increment for each subsequent packet by traceroute. Furthermore, all packet's headers keep changing.

7. Describe the pattern you see in the values in the Identification field of the IP datagram

The screenshot shows a Wireshark packet capture on an Ethernet interface. The packet list displays several ICMP Echo (ping) requests from 192.168.0.21 to 128.119.245.12, followed by three 'Time-to-live exceeded' messages. The packet details pane for packet 83 shows the IP header fields: Version 4, Header Length 20 bytes, Differentiated Services Field 0x00, Total Length 56, Identification 0x71b6 (29110), Flags 0x0000, and Time to live 255. The packet bytes pane shows the raw data of the packet, with the Identification field (0x71b6) highlighted in blue.

No.	Time	Source	Destination	Protocol	Length	Info
106	20:03:53.686520	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3700/
104	20:03:53.647500	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3699/
102	20:03:53.609101	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3698/
96	20:03:53.570072	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3697/
91	20:03:53.532041	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3696/
84	20:03:53.493039	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3695/
83	20:03:53.454441	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3694/
1750	20:05:10.181032	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce
1659	20:05:07.682418	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce
1560	20:05:05.179842	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce

Frame 83: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
 Ethernet II, Src: AsustekC_91:eb:e8 (04:d4:c4:91:eb:e8), Dst: Motorola_f5:79:b1 (88:b4:a6:f5:79:b1)
 Internet Protocol Version 4, Src: 192.168.0.21, Dst: 128.119.245.12
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 56
 Identification: 0x71b6 (29110)
 > Flags: 0x0000
 Time to live: 255

0000 88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00 ...y...E.
 0010 00 38 71 b6 00 00 ff 01 00 00 c0 a8 00 15 80 77 ...8q.....w
 0020 f5 0c 08 00 27 cf 00 01 0e 6e 20 20 20 20 20 ...'....n
 0030 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
 0040 20 20 20 20 20 20

Identification (p.id), 2 bytes

Packets: 1839 · Displayed: 639 (34.7%) · Dropped: 0 (0.0%) | Profile: Default

*Ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	Time	Source	Destination	Protocol	Length	Info
106	20:03:53.686520	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3700/
104	20:03:53.647500	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3699/
102	20:03:53.609101	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3698/
96	20:03:53.570072	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3697/
91	20:03:53.532041	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3696/
84	20:03:53.493039	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3695/
83	20:03:53.454441	192.168.0.21	128.119.245.12	ICMP	70	Echo (ping) request id=0x0001, seq=3694/
1750	20:05:10.181032	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce
1659	20:05:07.682418	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce
1560	20:05:05.179842	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exce

> Frame 84: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0

> Ethernet II, Src: AsustekC_91:eb:e8 (04:d4:c4:91:eb:e8), Dst: Motorola_f5:79:b1 (88:b4:a6:f5:79:b1)

> Internet Protocol Version 4, Src: 192.168.0.21, Dst: 128.119.245.12

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 56

Identification: 0x71b7 (29111)

> Flags: 0x0000

> Time to live: 1

```

0000  88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00  ....y... ..E.
0010  00 38 71 b7 00 00 01 01 00 00 c0 a8 00 15 80 77  -8q.....w
0020  f5 0c 08 00 27 ce 00 01 0e 6f 20 20 20 20 20 20  .....o
0030  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040  20 20 20 20 20 20

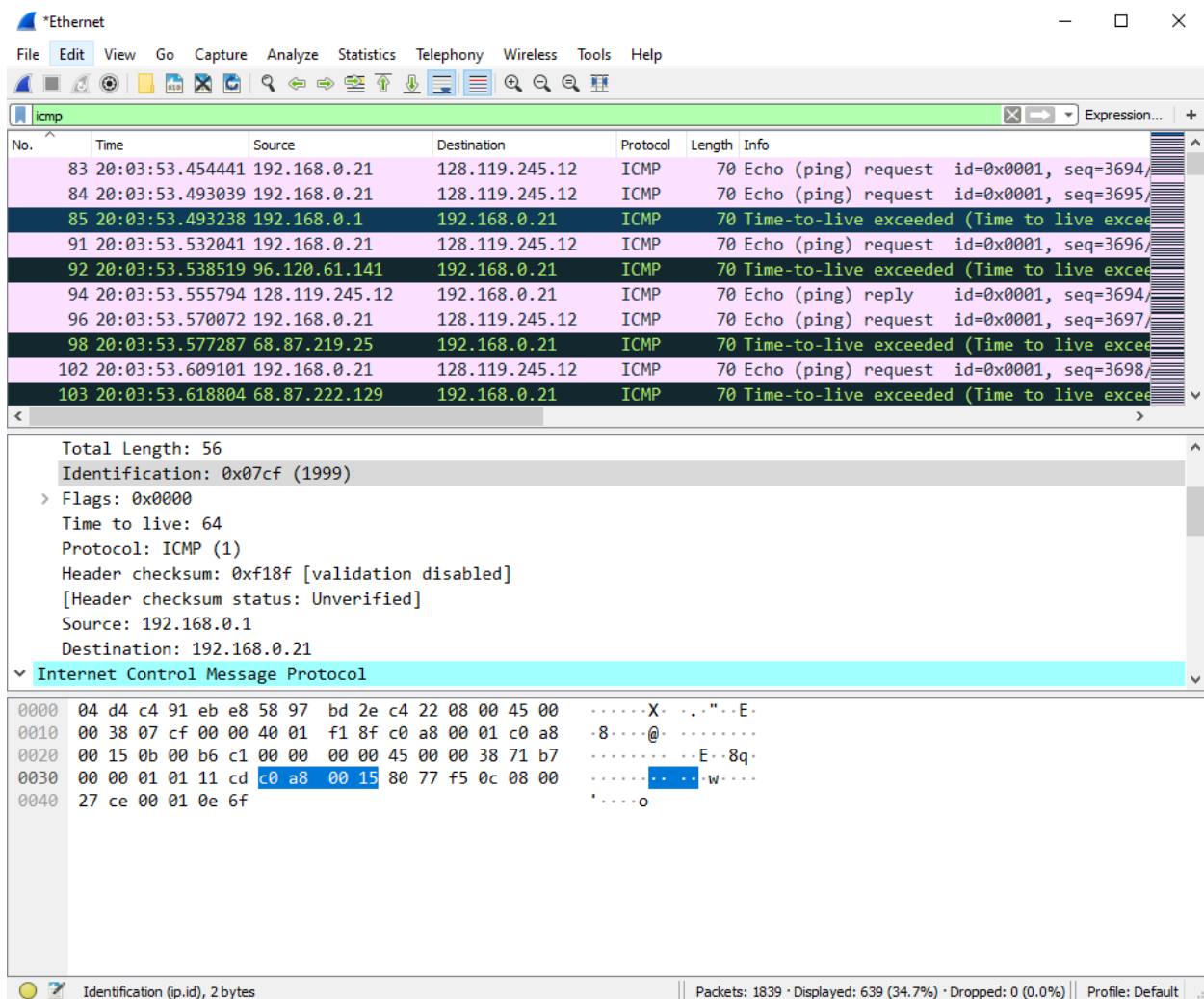
```

Identification (p.id), 2 bytes

Packets: 1839 · Displayed: 639 (34.7%) · Dropped: 0 (0.0%) | Profile: Default

- The pattern seems that the IP header identification fields keep increasing by 1 per one ICMP echo ping request from my computer ethernet.

8. What is the value in the Identification field and the TTL field?



- The identification is 0x07cf (1999) and the TTL is 64.

9. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

- No, it is not remained. The identification keeps changing for all of the ICMP TTL-exceeded replies because it is unique value by source. Some identification values are remained the same because it caused by fragments from single IP datagram.

- The TTL remains the same value in the first hop to the router. However, if another hop to another router is happened, then TTL value changed.

10. Find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 2000. Has that message been fragmented across more than one IP datagram? [Note: if you find your packet has not been fragmented, you should download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip> and extract the

[illegible]

11. Screenshot the first fragment of the fragmented IP datagram (with sufficient details to answer these questions). What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?

***Ethernet**

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip

No.	Time	Source	Destination	Protocol	Length	Info
305	00:25:04.895012	192.168.0.21	192.168.0.13	TCP	171	54710 → 8009 [PSH, ACK] Seq=469 Ack=469 Win=0 Len=0
306	00:25:04.896848	192.168.0.13	192.168.0.21	TCP	171	8009 → 54710 [PSH, ACK] Seq=469 Ack=586 Win=0 Len=0
307	00:25:04.937018	192.168.0.21	192.168.0.13	TCP	54	54710 → 8009 [ACK] Seq=586 Ack=586 Win=1088 Len=0
309	00:25:05.203039	192.168.0.21	230.0.0.1	UDP	92	60450 → 6666 Len=50
311	00:25:06.203527	192.168.0.21	230.0.0.1	UDP	92	60450 → 6666 Len=50
313	00:25:06.488341	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, offset=0)
314	00:25:06.488343	192.168.0.21	128.119.245.12	ICMP	534	Echo (ping) request id=0x0001, seq=4114, len=534
317	00:25:06.526690	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, offset=0)
318	00:25:06.526692	192.168.0.21	128.119.245.12	ICMP	534	Echo (ping) request id=0x0001, seq=4115, len=534
319	00:25:06.526864	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live exceeded)

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 1500

Identification: 0x735a (29530)

Flags: 0x2000, More fragments

0... .. = Reserved bit: Not set

.0.. .. = Don't fragment: Not set

..1. = More fragments: Set

...0 0000 0000 0000 = Fragment offset: 0

Time to live: 255

0010 05 dc 73 5a 20 00 ff 01 00 00 c0 a8 00 15 80 77 ...sZw

0020 f5 0c 08 00 2c 31 00 01 10 12 20 20 20 20 20,1.. ..

0030 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

0040 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

0050 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

0060 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

0070 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

0080 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

0090 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

00a0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

00b0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

00c0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

More fragments (ip.flags.mf), 2 bytes

Packets: 1049 · Displayed: 827 (78.8%) · Dropped: 0 (0.0%) Profile: Default

- IP protocol datagram packet (number 313) shows that the more fragments flag bit is set. This means that the datagram is fragmented. In addition, the fragment offset is 0 from above screenshot, so this is first fragment which has total length of 1500(1480 + 20(header)) bytes

12. Screenshot the second fragment of the fragmented IP datagram (with sufficient details to answer these questions). What information in the IP header indicates that this is not the first datagram fragment? Are the more fragments? How can you tell?

***Ethernet**

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip

No.	Time	Source	Destination	Protocol	Length	Info
305	00:25:04.895012	192.168.0.21	192.168.0.13	TCP	171	54710 → 8009 [PSH, ACK] Seq=469 Ack=469 Win=...
306	00:25:04.896848	192.168.0.13	192.168.0.21	TCP	171	8009 → 54710 [PSH, ACK] Seq=469 Ack=586 Win=...
307	00:25:04.937018	192.168.0.21	192.168.0.13	TCP	54	54710 → 8009 [ACK] Seq=586 Ack=586 Win=16...
309	00:25:05.203039	192.168.0.21	230.0.0.1	UDP	92	60450 → 6666 Len=50
311	00:25:06.203527	192.168.0.21	230.0.0.1	UDP	92	60450 → 6666 Len=50
313	00:25:06.488341	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off...
314	00:25:06.488343	192.168.0.21	128.119.245.12	ICMP	534	Echo (ping) request id=0x0001, seq=4114/...
317	00:25:06.526690	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off...
318	00:25:06.526692	192.168.0.21	128.119.245.12	ICMP	534	Echo (ping) request id=0x0001, seq=4115/...
319	00:25:06.526864	192.168.0.1	192.168.0.21	ICMP	70	Time-to-live exceeded (Time to live excee...

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 520

Identification: 0x735a (29530)

Flags: 0x00b9

0... .. = Reserved bit: Not set

.0.. .. = Don't fragment: Not set

..0. = More fragments: Not set

...0 0000 1011 1001 = Fragment offset: 185

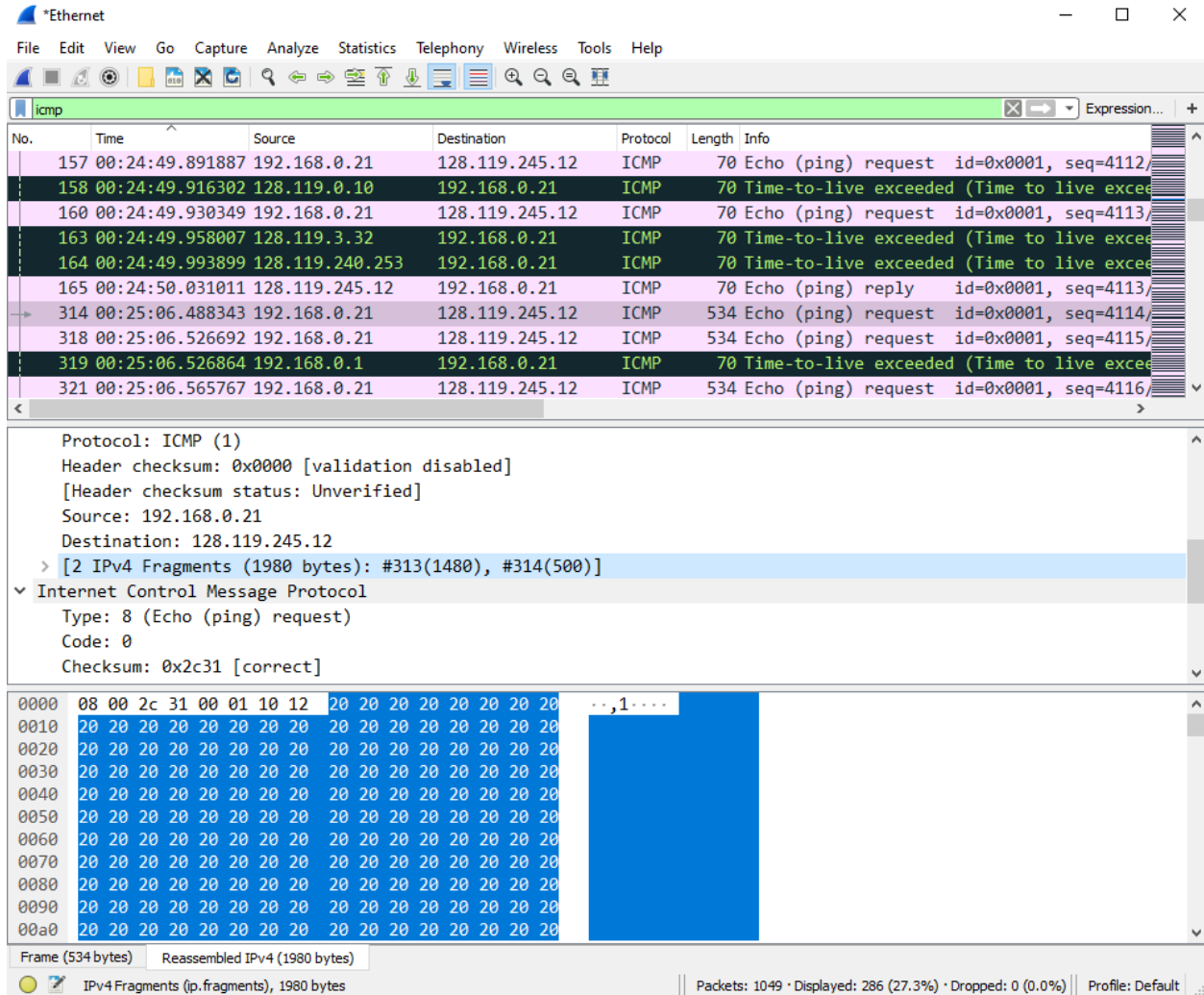
Time to live: 255

0010	02 08 73 5a 00 b9 ff 01 00 00 c0 a8 00 15 80 77	...sZ....
0020	f5 0c 20 20 20 20 20 20 20 20 20 20 20 20 20	...
0030	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
0040	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
0050	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
0060	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
0070	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
0080	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
0090	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
00a0	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
00b0	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	

Frame (534 bytes) Reassembled IPv4 (1980 bytes)

Fragment offset (13 bits) (ip.frag_offset), 2 bytes

Packets: 1049 · Displayed: 827 (78.8%) · Dropped: 0 (0.0%) Profile: Default



- From the above first screenshot, the fragment offset is 185. This means this is not first fragment. Furthermore, the second screenshot shows that there are two IPv4 fragments (1980 bytes): #313(1480), #314(500). Therefore, there is no more fragment after packet number 314.

13. What fields change in the IP header between the first and second fragment?

- When I compared the two fragments (#313 and #314),
 - the total length is changed from 1500 to 520
 - flags are changed from 0x2000, more fragments to 0x00b9
 - fragment offset from 0 to 185

However, the checksum should be changed because all headers keep changing, but my wireshark program is not indicated.

Now find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 3500.

14. How many fragments were created from the original datagram?

The image shows a Wireshark packet capture interface. The top pane displays a list of packets. Packet 666 is selected, showing it is an ICMP Echo (ping) request. The bottom pane shows the details of this packet, indicating it is fragmented into 3 IPv4 fragments (3480 bytes total). The packet list shows packets 664, 665, and 666 as fragments of the ICMP Echo request. The packet details pane for packet 666 shows the ICMP header and the reassembled IPv4 packet. The packet bytes pane shows the raw data of the fragments, with a blue box indicating the reassembled data.

No.	Time	Source	Destination	Protocol	Length	Info
660	00:25:29.909291	192.168.0.21	192.168.0.13	TCP	171	54710 → 8009 [PSH, ACK] Seq=1054 Ack=1054
661	00:25:29.911125	192.168.0.13	192.168.0.21	TCP	171	8009 → 54710 [PSH, ACK] Seq=1054 Ack=1171
662	00:25:29.951493	192.168.0.21	192.168.0.13	TCP	54	54710 → 8009 [ACK] Seq=1171 Ack=1171 Win=
663	00:25:30.213414	192.168.0.21	230.0.0.1	UDP	92	60450 → 6666 Len=50
664	00:25:30.268982	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
665	00:25:30.268984	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
666	00:25:30.268985	192.168.0.21	128.119.245.12	ICMP	554	Echo (ping) request id=0x0001, seq=4168
669	00:25:30.307809	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
670	00:25:30.307812	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
671	00:25:30.307813	192.168.0.21	128.119.245.12	ICMP	554	Echo (ping) request id=0x0001, seq=4169

Protocol: ICMP (1)
Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.0.21
Destination: 128.119.245.12
> [3 IPv4 Fragments (3480 bytes): #664(1480), #665(1480), #666(520)]

Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x0ddd [correct]

Frame (554 bytes) Reassembled IPv4 (3480 bytes)

IPv4 Fragments (p.fragments), 3480 bytes Packets: 1049 · Displayed: 827 (78.8%) · Dropped: 0 (0.0%) Profile: Default

- There are 3 fragments (#664 packet 1480 bytes, #665 packet 1480bytes, #666 packet 520bytes) were created from original datagram after I changed the Packet Size in pingplotter to be 3500.

15. What fields change in the IP header among the fragments?

Wireshark capture of Ethernet traffic. The packet list shows a series of TCP and UDP packets, followed by ICMP Echo (ping) requests. The selected packet (No. 666) is an ICMP Echo request from 192.168.0.21 to 128.119.245.12. The packet details show the ICMP header and the data payload (1480 bytes).

No.	Time	Source	Destination	Protocol	Length	Info
660	00:25:29.909291	192.168.0.21	192.168.0.13	TCP	171	54710 → 8009 [PSH, ACK] Seq=1054 Ack=1054
661	00:25:29.911125	192.168.0.21	192.168.0.13	TCP	171	8009 → 54710 [PSH, ACK] Seq=1054 Ack=1171
662	00:25:29.951493	192.168.0.21	192.168.0.13	TCP	54	54710 → 8009 [ACK] Seq=1171 Ack=1171 Win=
663	00:25:30.213414	192.168.0.21	230.0.0.1	UDP	92	60450 → 6666 Len=50
664	00:25:30.268982	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
665	00:25:30.268984	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
666	00:25:30.268985	192.168.0.21	128.119.245.12	ICMP	554	Echo (ping) request id=0x0001, seq=4168/
669	00:25:30.307809	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
670	00:25:30.307812	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
671	00:25:30.307813	192.168.0.21	128.119.245.12	ICMP	554	Echo (ping) request id=0x0001, seq=4169/

Total Length: 1500
Identification: 0x7390 (29584)
Flags: 0x2000, More fragments
0... .. = Reserved bit: Not set
.0.. .. = Don't fragment: Not set
..1. = More fragments: Set
...0 0000 0000 0000 = Fragment offset: 0
Time to live: 255
Protocol: ICMP (1)
Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.0.21
Destination: 128.119.245.12
[Reassembled IPv4 in frame: 666](#)
Data (1480 bytes)
Data: 08000ddd0001104820202020202020202020202020202020...
[Length: 1480]

0000 88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00 ...y... ..E
0010 05 dc 73 90 20 00 ff 01 00 00 c0 a8 00 15 80 77 ..s... ..w
0020 f5 0c 08 00 0d dd 00 01 10 48 20 20 20 20 20 20H
0030 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

Wireshark_Ethernet_20191117002441_a05240.pcapng | Packets: 1049 · Displayed: 827 (78.8%) · Dropped: 0 (0.0%) | Profile: Default

Wireshark capture of Ethernet traffic. The packet list shows a series of TCP and UDP packets, followed by ICMP Echo (ping) requests. The selected packet (No. 666) is an ICMP Echo request from 192.168.0.21 to 128.119.245.12. The packet details show the ICMP header and the data payload (1480 bytes).

No.	Time	Source	Destination	Protocol	Length	Info
660	00:25:29.909291	192.168.0.21	192.168.0.13	TCP	171	54710 → 8009 [PSH, ACK] Seq=1054 Ack=1054
661	00:25:29.911125	192.168.0.21	192.168.0.13	TCP	171	8009 → 54710 [PSH, ACK] Seq=1054 Ack=1171
662	00:25:29.951493	192.168.0.21	192.168.0.13	TCP	54	54710 → 8009 [ACK] Seq=1171 Ack=1171 Win=
663	00:25:30.213414	192.168.0.21	230.0.0.1	UDP	92	60450 → 6666 Len=50
664	00:25:30.268982	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
665	00:25:30.268984	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
666	00:25:30.268985	192.168.0.21	128.119.245.12	ICMP	554	Echo (ping) request id=0x0001, seq=4168/
669	00:25:30.307809	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
670	00:25:30.307812	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
671	00:25:30.307813	192.168.0.21	128.119.245.12	ICMP	554	Echo (ping) request id=0x0001, seq=4169/

Total Length: 1500
Identification: 0x7390 (29584)
Flags: 0x20b9, More fragments
0... .. = Reserved bit: Not set
.0.. .. = Don't fragment: Not set
..1. = More fragments: Set
...0 0000 1011 1001 = Fragment offset: 185
Time to live: 255
Protocol: ICMP (1)
Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.0.21
Destination: 128.119.245.12
[Reassembled IPv4 in frame: 666](#)
Data (1480 bytes)
Data: 20...
[Length: 1480]

0000 88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00 ...y... ..E
0010 05 dc 73 90 20 b9 ff 01 00 00 c0 a8 00 15 80 77 ..s... ..w
0020 f5 0c 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0030 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

Wireshark_Ethernet_20191117002441_a05240.pcapng | Packets: 1049 · Displayed: 827 (78.8%) · Dropped: 0 (0.0%) | Profile: Default

Wireshark packet capture showing ICMP Echo (ping) request fragments. The packet list shows packets 660-671. Packet 666 is selected, showing details for the ICMP Echo request. The packet bytes pane shows the raw data of the fragments.

No.	Time	Source	Destination	Protocol	Length	Info
660	00:25:29.909291	192.168.0.21	192.168.0.13	TCP	171	54710 → 8009 [PSH, ACK] Seq=1054 Ack=1054
661	00:25:29.911125	192.168.0.21	192.168.0.13	TCP	171	8009 → 54710 [PSH, ACK] Seq=1054 Ack=1171
662	00:25:29.951493	192.168.0.21	192.168.0.13	TCP	54	54710 → 8009 [ACK] Seq=1171 Ack=1171 Win=
663	00:25:30.213414	192.168.0.21	230.0.0.1	UDP	92	60450 → 6666 Len=50
664	00:25:30.268982	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
665	00:25:30.268984	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
666	00:25:30.268985	192.168.0.21	128.119.245.12	ICMP	554	Echo (ping) request id=0x0001, seq=4168/
669	00:25:30.307809	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
670	00:25:30.307812	192.168.0.21	128.119.245.12	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off
671	00:25:30.307813	192.168.0.21	128.119.245.12	ICMP	554	Echo (ping) request id=0x0001, seq=4169/

Details for packet 666 (ICMP Echo request):

- Header Length: 20 bytes (5)
- Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
- Total Length: 540
- Identification: 0x7390 (29584)
- Flags: 0x0172
 - 0... .. = Reserved bit: Not set
 - .0.. .. = Don't fragment: Not set
 - ..0. = More fragments: Not set
 - ...0 0001 0111 0010 = Fragment offset: 370
- Time to live: 255
- Protocol: ICMP (1)
- Header checksum: 0x0000 [validation disabled]
- [Header checksum status: Unverified]
- Source: 192.168.0.21
- Destination: 128.119.245.12
- [3 IPv4 Fragments (3480 bytes): #664(1480), #665(1480), #666(520)]

Internet Control Message Protocol

Frame (554 bytes) Reassembled IPv4 (3480 bytes)

IPv4 Fragments (ip.fragments), 3480 bytes

Packets: 1049 · Displayed: 827 (78.8%) · Dropped: 0 (0.0%) Profile: Default

- When I compared the three fragments (#664, #665, and #666),

- the total length is changed from 1500 (#664 and #665) to 540 (#666) bytes
- flags are changed from 0x2000 (#664) to 0x20b9 (#665) to 0x0172 (#666).
- More fragments flag of #664 and #665 was set, but it changed to not set on #666
- fragment offset from 0 (#664) to 185 (#665) and then from 185 (#665) to 370 (#666)

However, the checksum should be changed because all headers keep changing, but my wireshark program is not indicated.

Lab 4 – extra credit #2

Programming language: C

Explanation: I used Mobaxterm terminal by creating another process for listening on local host.

1. server.c

```
C server.c x
C: > Users > 15419 > AppData > Local > Temp > Mxt121 > RemoteFiles > 132778_7_6 > C server.c
1 //written by Junhyeok Jeong
2
3 // server code for UDP socket programming
4 #include <arpa/inet.h>
5 #include <netinet/in.h>
6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <string.h>
9 #include <sys/socket.h>
10 #include <sys/types.h>
11 #include <unistd.h>
12
13 #define IP_PROTOCOL 0
14 #define PORT_NO 5000
15 #define NET_BUF_SIZE 32
16 #define cipherKey 'S'
17 #define sendrecvflag 0
18 #define nofile "File Not Found!"
19
20 // function to clear buffer
21 void clearBuf(char* b)
22 {
23     int i;
24     for (i = 0; i < NET_BUF_SIZE; i++)
25         b[i] = '\0';
26 }
27
28 // function to encrypt
29 char Cipher(char ch)
30 {
31     return ch ^ cipherKey;
32 }
33
34 // function sending file
35 int sendFile(FILE* fp, char* buf, int s)
36 {
37     int i, len;
38     if (fp == NULL) {
39         strcpy(buf, nofile);
40         len = strlen(nofile);
41         buf[len] = EOF;
42         for (i = 0; i <= len; i++)
43             buf[i] = Cipher(buf[i]);
44         return 1;
45     }
46
47     char ch, ch2;
48     for (i = 0; i < s; i++) {
49         ch = fgetc(fp);
```

```

47     char ch, ch2;
48     for (i = 0; i < s; i++) {
49         ch = fgetc(fp);
50         ch2 = Cipher(ch);
51         buf[i] = ch2;
52         if (ch == EOF)
53             return 1;
54     }
55     return 0;
56 }
57
58 // driver code
59 int main()
60 {
61     int sockfd, nBytes;
62     struct sockaddr_in addr_con;
63     int addrlen = sizeof(addr_con);
64     addr_con.sin_family = AF_INET;
65     addr_con.sin_port = htons(PORT_NO);
66     addr_con.sin_addr.s_addr = INADDR_ANY;
67     char net_buf[NET_BUF_SIZE];
68     FILE* fp;
69
70     // socket()
71     sockfd = socket(AF_INET, SOCK_DGRAM, IP_PROTOCOL);
72
73     if (sockfd < 0)
74         printf("\nError: Check your socket!!\n");
75     else
76         printf("\nfile name will be received on sockfd %d \n", sockfd);
77
78     // bind()
79     if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(addr_con)) == 0)
80         printf("\nSuccessfully binded!\n");
81     else
82         printf("\nBinding Failed!\n");
83
84     while (1) {
85         printf("\nlistening client connection and Waiting for file name...\n");
86
87         // receive file name
88         clearBuf(net_buf);
89
90         nBytes = recvfrom(sockfd, net_buf,
91                         NET_BUF_SIZE, 0,
92                         (struct sockaddr*)&addr_con, &addrlen);
93     }

```

```

// receive file name
clearBuf(net_buf);

nBytes = recvfrom(sockfd, net_buf,
                  NET_BUF_SIZE, sendrecvflag,
                  (struct sockaddr*)&addr_con, &addrlen);

fp = fopen(net_buf, "r");
printf("\nFile Name Received: %s\n", net_buf);
if (fp == NULL)
    printf("\nFile open failed!\n");
else
    printf("\nFile Successfully opened!\n");

while (1) {
    // process
    if (sendFile(fp, net_buf, NET_BUF_SIZE)) {
        sendto(sockfd, net_buf, NET_BUF_SIZE,
                sendrecvflag,
                (struct sockaddr*)&addr_con, addrlen);
        break;
    }

    // send
    sendto(sockfd, net_buf, NET_BUF_SIZE,
            sendrecvflag,
            (struct sockaddr*)&addr_con, addrlen);
    clearBuf(net_buf);
}
if (fp != NULL)
    fclose(fp);
}
return 0;
}

```

2. client.c

C server.c C client.c X
C: > Users > 15419 > AppData > Local > Temp > Mxt121 > RemoteFiles > 132778_7_7 > C client.c > main()

```
1 //written by Junhyeok Jeong
2
3 // client code for UDP socket programming
4 #include <arpa/inet.h>
5 #include <netinet/in.h>
6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <string.h>
9 #include <sys/socket.h>
10 #include <sys/types.h>
11 #include <unistd.h>
12
13 #define IP_PROTOCOL 0
14 #define IP_ADDRESS "127.0.0.1" // localhost
15 #define PORT_NO 5000
16 #define NET_BUF_SIZE 32
17 #define cipherKey 'S'
18 #define sendrecvflag 0
19
20 // function to clear buffer
21 void clearBuf(char* b)
22 {
23     int i;
24     for (i = 0; i < NET_BUF_SIZE; i++)
25         b[i] = '\0';
26 }
27
28 // function for decryption
29 char Cipher(char ch)
30 {
31     return ch ^ cipherKey;
32 }
33
34 // function to receive file
35 int rcvFile(char* buf, int s, FILE *write_file)
36 {
37     int i;
38     char ch;
39
40     for (i = 0; i < s; i++) {
41         ch = buf[i];
42         ch = Cipher(ch);
43         if (ch == EOF)
44             return 1;
45         else
46             fprintf(write_file, "%c", ch);
47         printf("%c", ch);
```

```

52 // driver code
53 int main()
54 {
55     int sockfd, nBytes;
56     struct sockaddr_in addr_con;
57     int addrlen = sizeof(addr_con);
58     addr_con.sin_family = AF_INET;
59     addr_con.sin_port = htons(PORT_NO);
60     addr_con.sin_addr.s_addr = inet_addr(IP_ADDRESS);
61     char net_buf[NET_BUF_SIZE];
62     FILE *write_file;
63
64     // socket()
65     sockfd = socket(AF_INET, SOCK_DGRAM,
66                     IP_PROTOCOL);
67
68     if (sockfd < 0)
69         printf("\nError: check your socket!!\n");
70     else
71         printf("\nfile will be received on sockfd %d\n", sockfd);
72
73     while (1) {
74         printf("\nPlease enter file name to receive:\n");
75         scanf("%s", net_buf);
76         write_file = fopen(net_buf, "w");
77
78         sendto(sockfd, net_buf, NET_BUF_SIZE,
79                sendrecvflag, (struct sockaddr*)&addr_con,
80                addrlen);
81
82         printf("\n-----Data Received-----\n");
83
84         while (1) {
85             // receive
86             clearBuf(net_buf);
87             nBytes = recvfrom(sockfd, net_buf, NET_BUF_SIZE,
88                               sendrecvflag, (struct sockaddr*)&addr_con,
89                               &addrlen);
90
91             // process
92             if (recvFile(net_buf, NET_BUF_SIZE, write_file)) {
93                 break;
94             }
95         }
96         printf("\n-----\n");

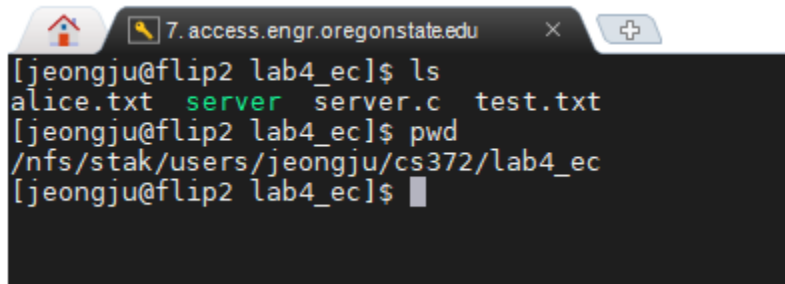
```

```

98         printf("\nthe requested file is received and stored successfully! And socket connection is end\n");
99         break;
100     }
101
102     return 0;
103 }
104
105

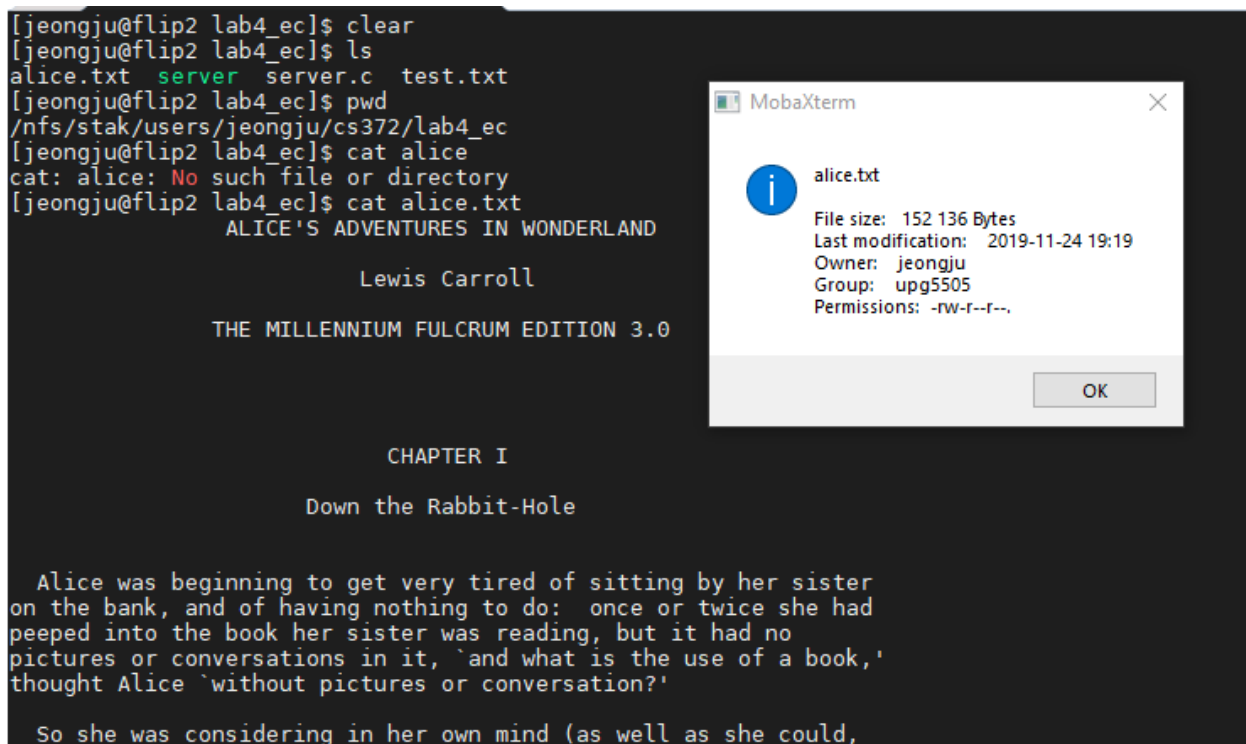
```

3. Server.c is in different directory



```
[jeongju@flip2 lab4_ec]$ ls
alice.txt  server  server.c  test.txt
[jeongju@flip2 lab4_ec]$ pwd
/nfs/stak/users/jeongju/cs372/lab4_ec
[jeongju@flip2 lab4_ec]$
```

4. Test file: alice.txt (from lab3)



```
[jeongju@flip2 lab4_ec]$ clear
[jeongju@flip2 lab4_ec]$ ls
alice.txt  server  server.c  test.txt
[jeongju@flip2 lab4_ec]$ pwd
/nfs/stak/users/jeongju/cs372/lab4_ec
[jeongju@flip2 lab4_ec]$ cat alice
cat: alice: No such file or directory
[jeongju@flip2 lab4_ec]$ cat alice.txt
ALICE'S ADVENTURES IN WONDERLAND

Lewis Carroll

THE MILLENNIUM FULCRUM EDITION 3.0

CHAPTER I

Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do:  once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, 'and what is the use of a book,'
thought Alice 'without pictures or conversation?'

So she was considering in her own mind (as well as she could,
```

MobaXterm

i alice.txt

File size: 152 136 Bytes
Last modification: 2019-11-24 19:19
Owner: jeongju
Group: upg5505
Permissions: -rw-r--r--

OK

5. Compile server.c on child process for listening

```

[jeongju@flip2 lab4_ec]$ ps aux | grep jeongju
root      23445  0.0  0.0 183720  6136 ?        Ss   19:23   0:00 sshd: jeongju [priv]
jeongju   29186  0.0  0.0 165628  3072 pts/111  R+   19:53   0:00 ps aux
jeongju   29187  0.0  0.0 112716   964 pts/111  S+   19:53   0:00 grep --color=auto jeongju
jeongju   32672  0.0  0.0 184036  3096 ?        S    19:23   0:00 sshd: jeongju@pts/111
jeongju   32723  0.0  0.0 136644  2596 pts/111  Ss   19:23   0:00 -tcsh
[jeongju@flip2 lab4_ec]$ ./server &
[1] 487
[jeongju@flip2 lab4_ec]$ file name is received on sockfd 3

Successfully binded!

Waiting for file name...
jobs
[1]  + Running                  ./server
[jeongju@flip2 lab4_ec]$ █

```

6. Compile client.c on other directory

```

[jeongju@flip2 lab4_ec]$ ps aux | grep jeongju
root      23445  0.0  0.0 183720  6136 ?        Ss   19:23   0:00 sshd: jeongju [priv]
jeongju   29186  0.0  0.0 165628  3072 pts/111  R+   19:53   0:00 ps aux
jeongju   29187  0.0  0.0 112716   964 pts/111  S+   19:53   0:00 grep --color=auto jeongju
jeongju   32672  0.0  0.0 184036  3096 ?        S    19:23   0:00 sshd: jeongju@pts/111
jeongju   32723  0.0  0.0 136644  2596 pts/111  Ss   19:23   0:00 -tcsh
[jeongju@flip2 lab4_ec]$ ./server &
[1] 487
[jeongju@flip2 lab4_ec]$ file name is received on sockfd 3

Successfully binded!

Waiting for file name...
jobs
[1]  + Running                  ./server
[jeongju@flip2 lab4_ec]$ ls
alice.txt  server  server.c  test.txt
[jeongju@flip2 lab4_ec]$ cd ..
[jeongju@flip2 ~/cs372]$ pwd
/nfs/stak/users/jeongju/cs372
[jeongju@flip2 ~/cs372]$ ls
client  client.c  ec_lab1.py  ec_lab2.c  lab2  lab4_ec
[jeongju@flip2 ~/cs372]$ ./client
file is received

Please enter file name to receive:
█

```

7. Error handling for wrong file name

```
Please enter file name to receive:
idontknow

-----Data Received-----

File Name Received: idontknow

File open failed!

Waiting for file name...
File Not Found!
-----

Please enter file name to receive:
█
```

8. Test through alice.txt file and check the directory


```

[jeongju@flip2 ~/cs372]$ jobs
[1]  + Running                  ./server
[jeongju@flip2 ~/cs372]$ ls
client  client.c  ec_lab1.py  ec_lab2.c  lab2  lab4_ec
[jeongju@flip2 ~/cs372]$ clear
[jeongju@flip2 ~/cs372]$ gcc -o client client.c
[jeongju@flip2 ~/cs372]$ ./client

file will be received on socketfd 3

Please enter file name to receive:
alice.txt

-----Data Received-----

File Name Received: alice.txt

File Successfully opened!
      ALICE'S ADVENTURES IN WONDERLAND

      Lewis Carroll

      THE MILLENNIUM FULCRUM EDITION 3.0

      CHAPTER I

      Down the Rabbit-Hole

      Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do: once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, 'and what is the use of a book,'
thought Alice 'without pictures or conversation?'

      So she was considering in her own mind (as well as she could,
for the hot day made her feel very sleepy and stupid), whether
the pleasure of making a daisy-chain would be worth the trouble
of getting up and picking the daisies, when suddenly a White
Rabbit with pink eyes ran close by her.

      There was nothing so VERY remarkable in that; nor did Alice
think it so VERY much out of the way to hear the Rabbit say to
itself, 'Oh dear! Oh dear! I shall be late!' (when she thought
it over afterwards, it occurred to her that she ought to have
wondered at this, but at the time it all seemed quite natural);
but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-
POCKET, and looked at it, and then hurried on, Alice started to
her feet, for it flashed across her mind that she had never
before seen a rabbit with either a waistcoat-pocket, or a watch to
take out of it, and burning with curiosity, she ran across the
field after it, and fortunately was just in time to see it pop
down a large rabbit-hole under the hedge.

      In another moment down went Alice after it, never once
considering how in the world she was to get out again.

```

...

So she sat on, with closed eyes, and half believed herself in Wonderland, though she knew she had but to open them again, and all would change to dull reality--the grass would be only rustling in the wind, and the pool rippling to the waving of the reeds--the rattling teacups would change to tinkling sheep-bells, and the Queen's shrill cries to the voice of the shepherd boy--and the sneeze of the baby, the shriek of the Gryphon, and all the other queer noises, would change (she knew) to the confused clamour of the busy farm-yard--while the lowing of the cattle in the distance would take the place of the Mock Turtle's heavy sobs.

Lastly, she pictured to herself how this same little sister of hers would, in the after-time, be herself a grown woman; and how she would keep, through all her riper years, the simple and loving heart of her childhood: and how she would gather about her other little children, and make THEIR eyes bright and eager with many a strange tale, perhaps even with the dream of Wonderland of long ago: and how she would feel with all their simple sorrows, and find a pleasure in all their simple joys, remembering her own child-life, and the happy summer days.

THE END

Waiting for file name...

the requested file is received and stored successfully! And socket connection is end

[jeongju@flip2 ~/cs372]\$ ls

alice.txt client client.c ec_lab1.py ec_lab2.c lab2 lab4_ec

[jeongju@flip2 ~/cs372]\$ █

```
[jeongju@flip2 ~/cs372]$ ls
alice.txt  client  client.c  ec_lab1.py  ec_lab2.c  lab2  lab4_ec
[jeongju@flip2 ~/cs372]$ vim alice.txt
[jeongju@flip2 ~/cs372]$ cat alice.txt
    ALICE'S ADVENTURES IN WONDERLAND
```

Lewis Carroll

THE MILLENNIUM FULCRUM EDITION 3.0

CHAPTER I

Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, 'Oh dear! Oh dear! I shall be late!' (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and noticed that they were filled with cupboards and book-shelves;