Junhyeok Jeong

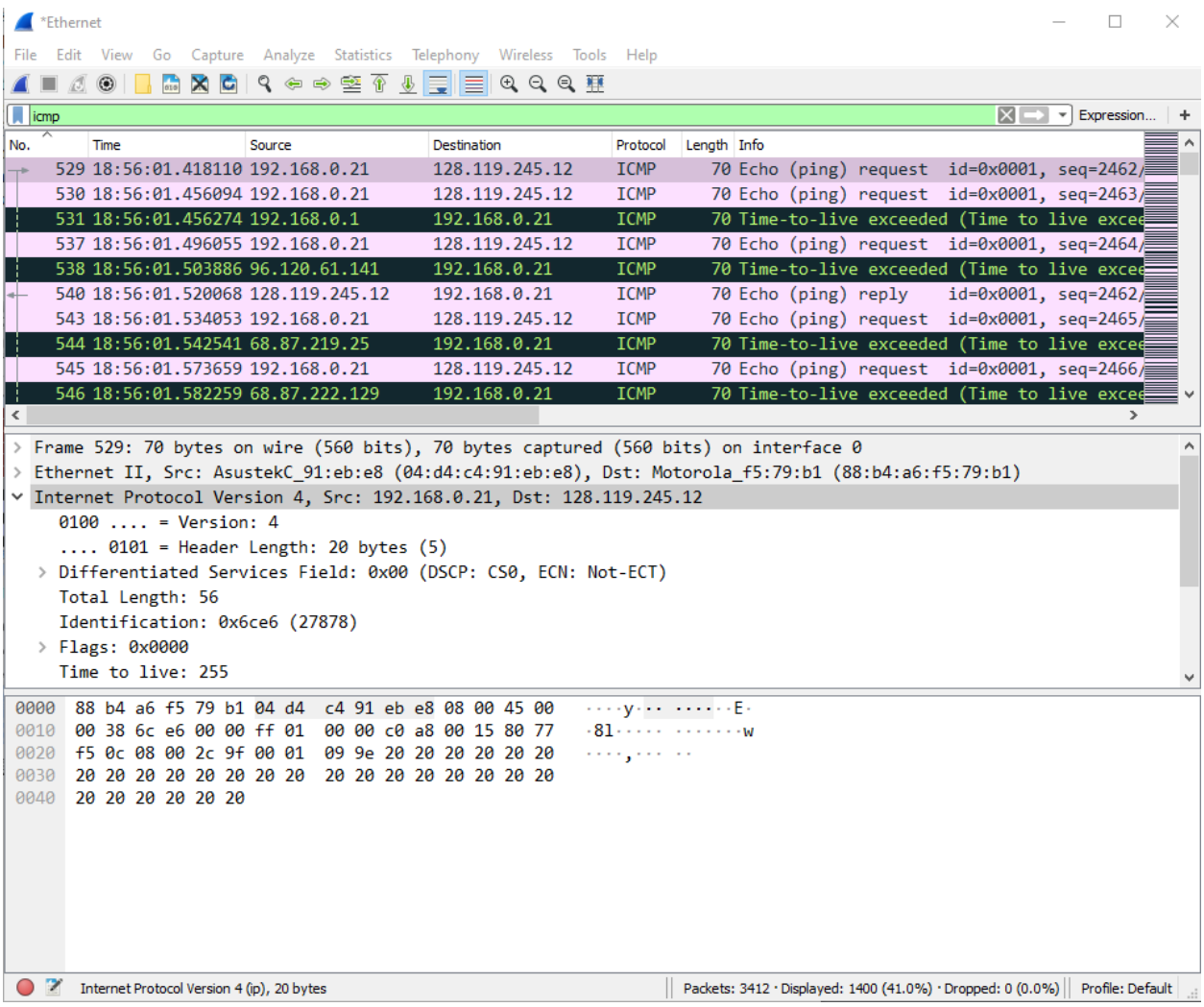
Bill Pfeil

CS 372-X001

11/24/2019

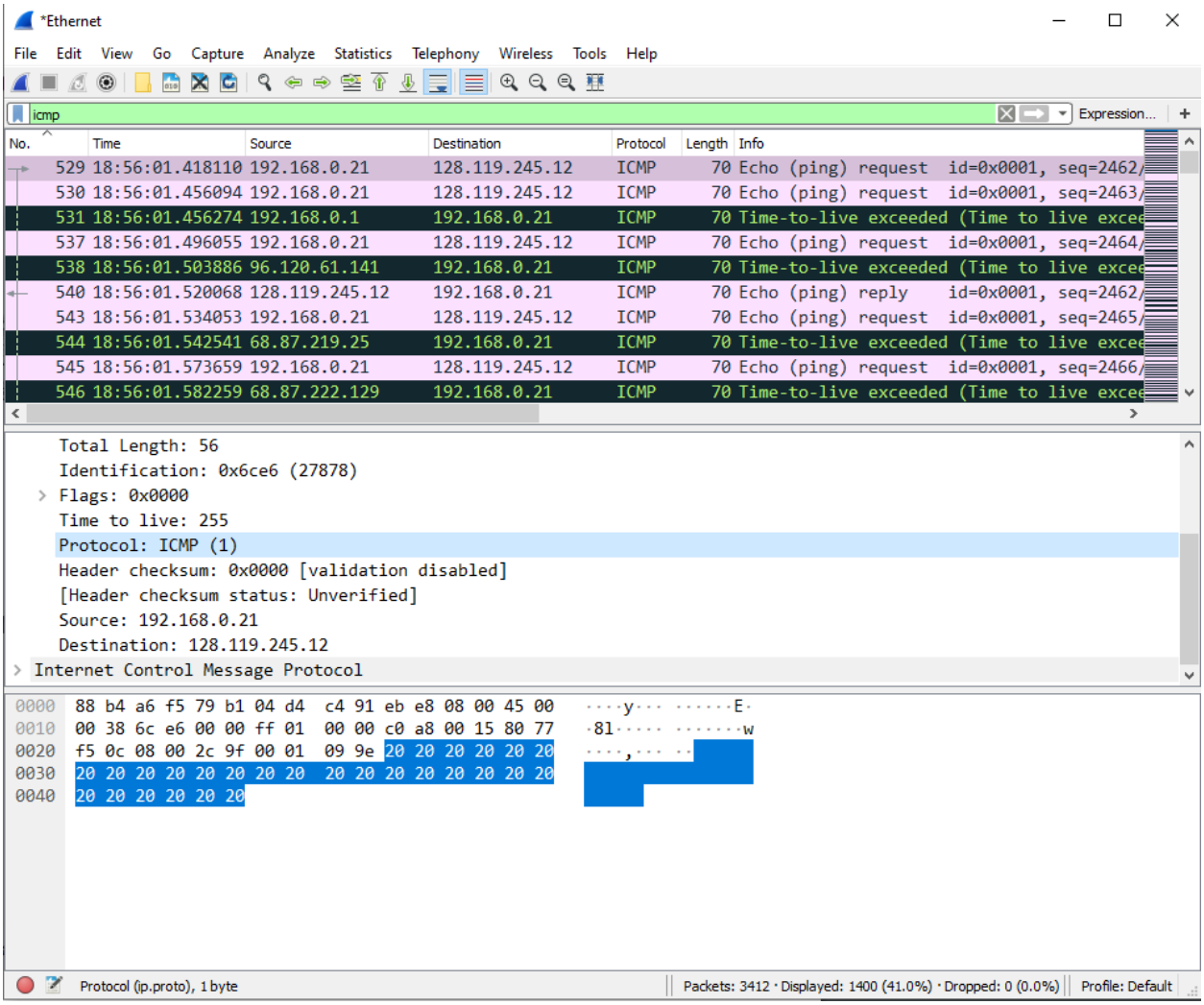
Lab 4

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?



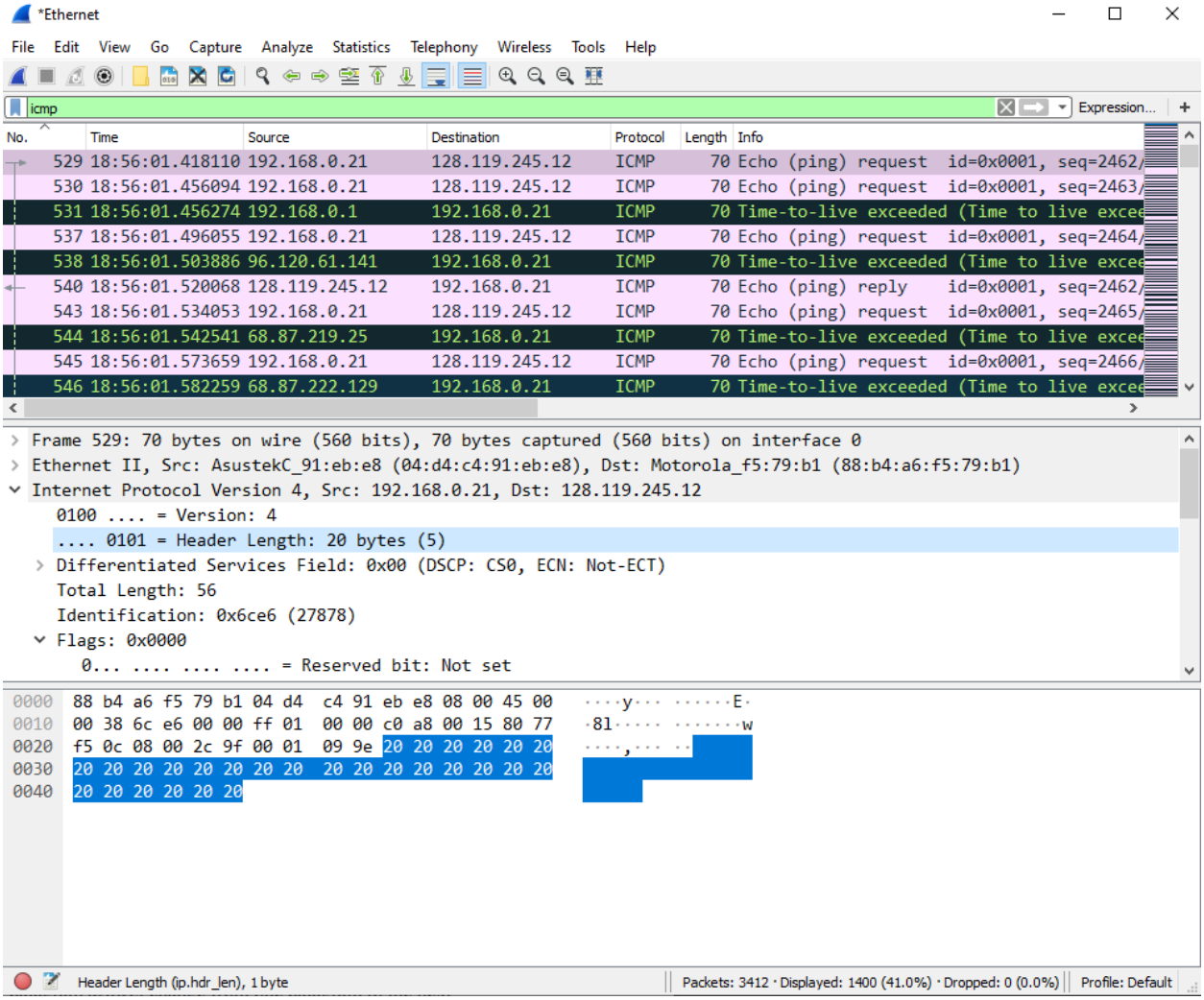
- My computer's IP address is 192.168.0.21

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



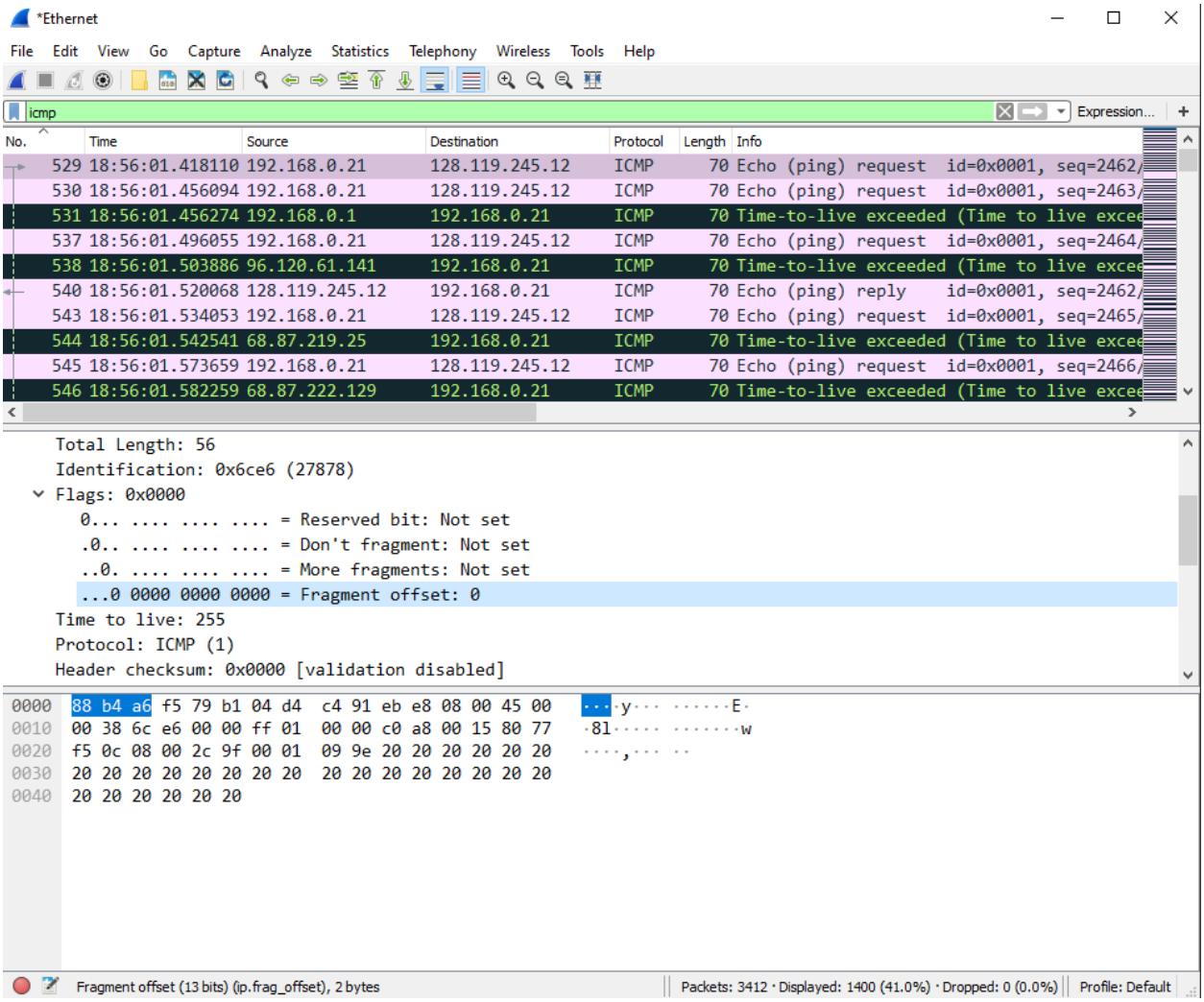
- Within the IP packet header, the value in the upper layer protocol field is ICMP (1)

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.



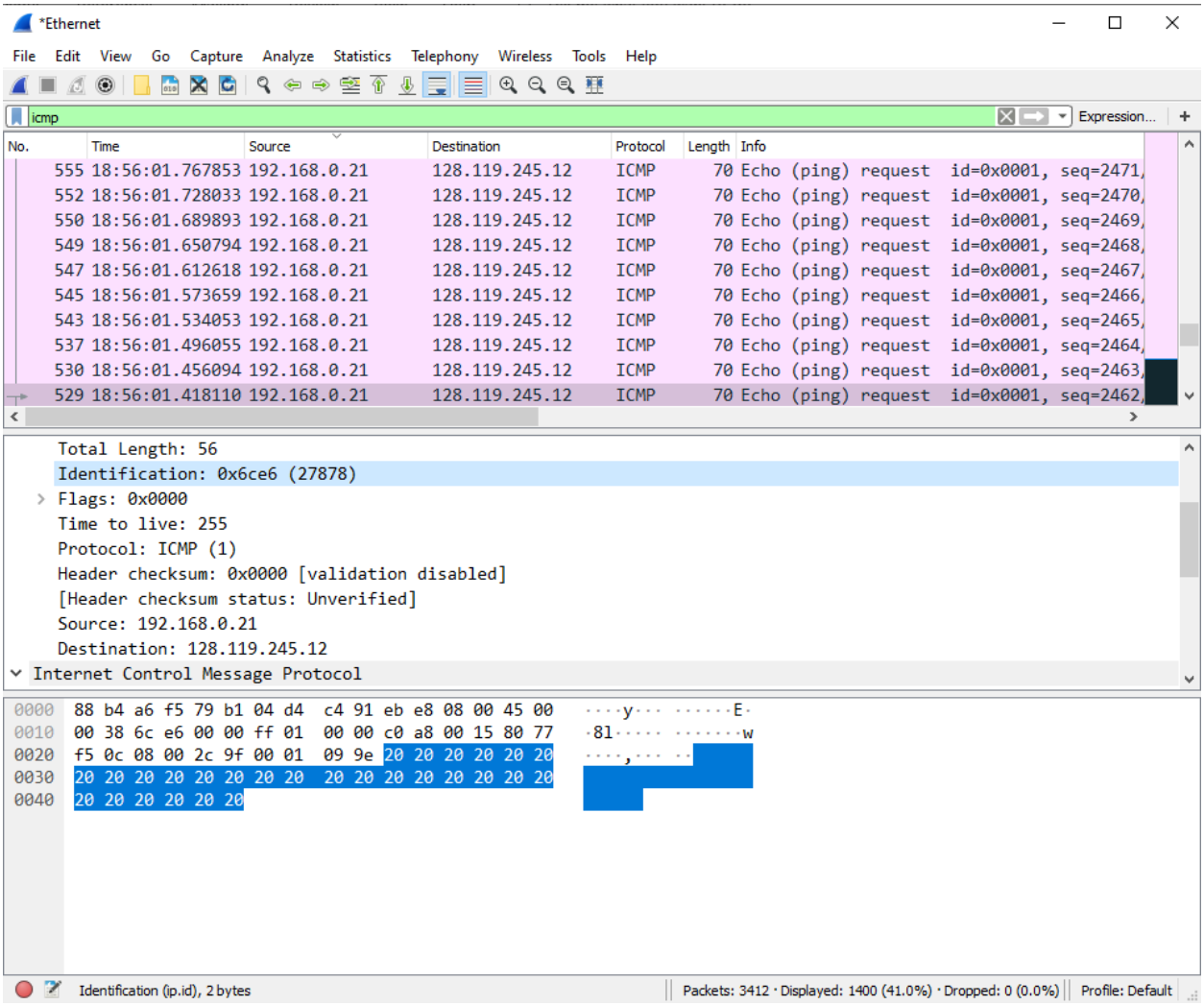
- The IP header length is 20 bytes and total length is 56 bytes because the payload length can be determined 56 bytes – 20 bytes = 36 bytes

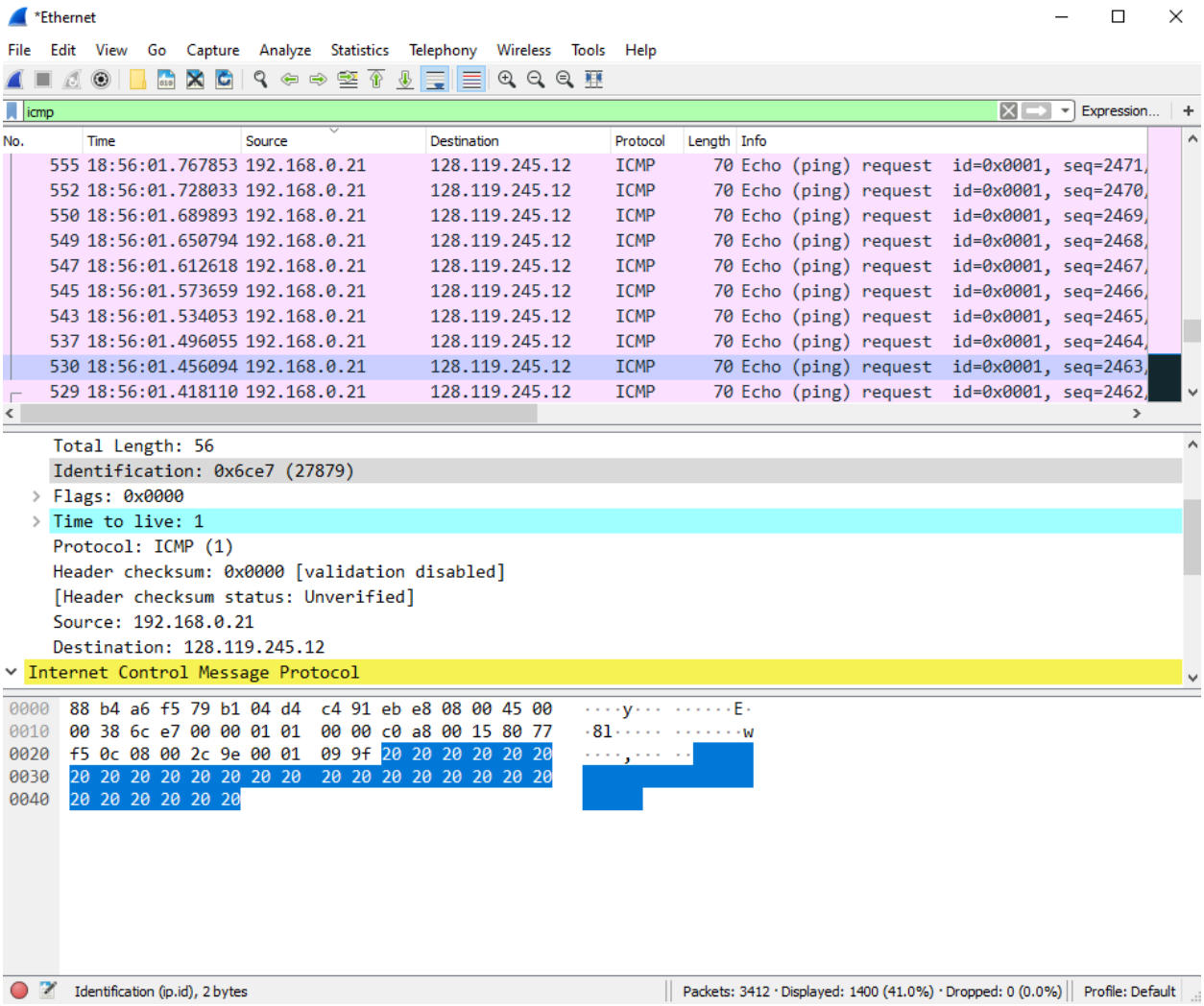
4. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.



- 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
Expression...
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 00
0010 00 38 6c e6 00 00 ff 01 00 00 c0 a8 00 15 80 77
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
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?

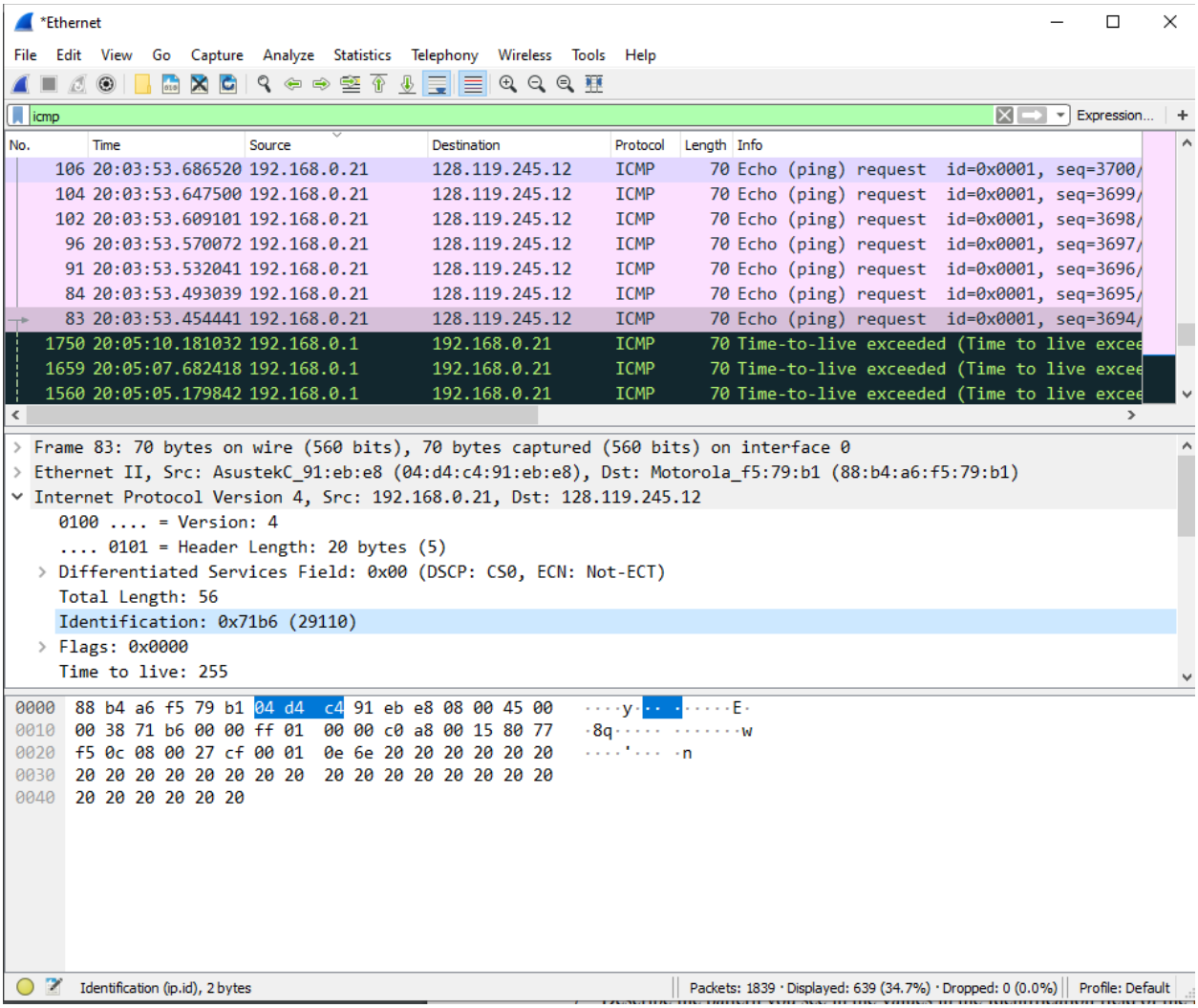
- The constantly stayed fields are:

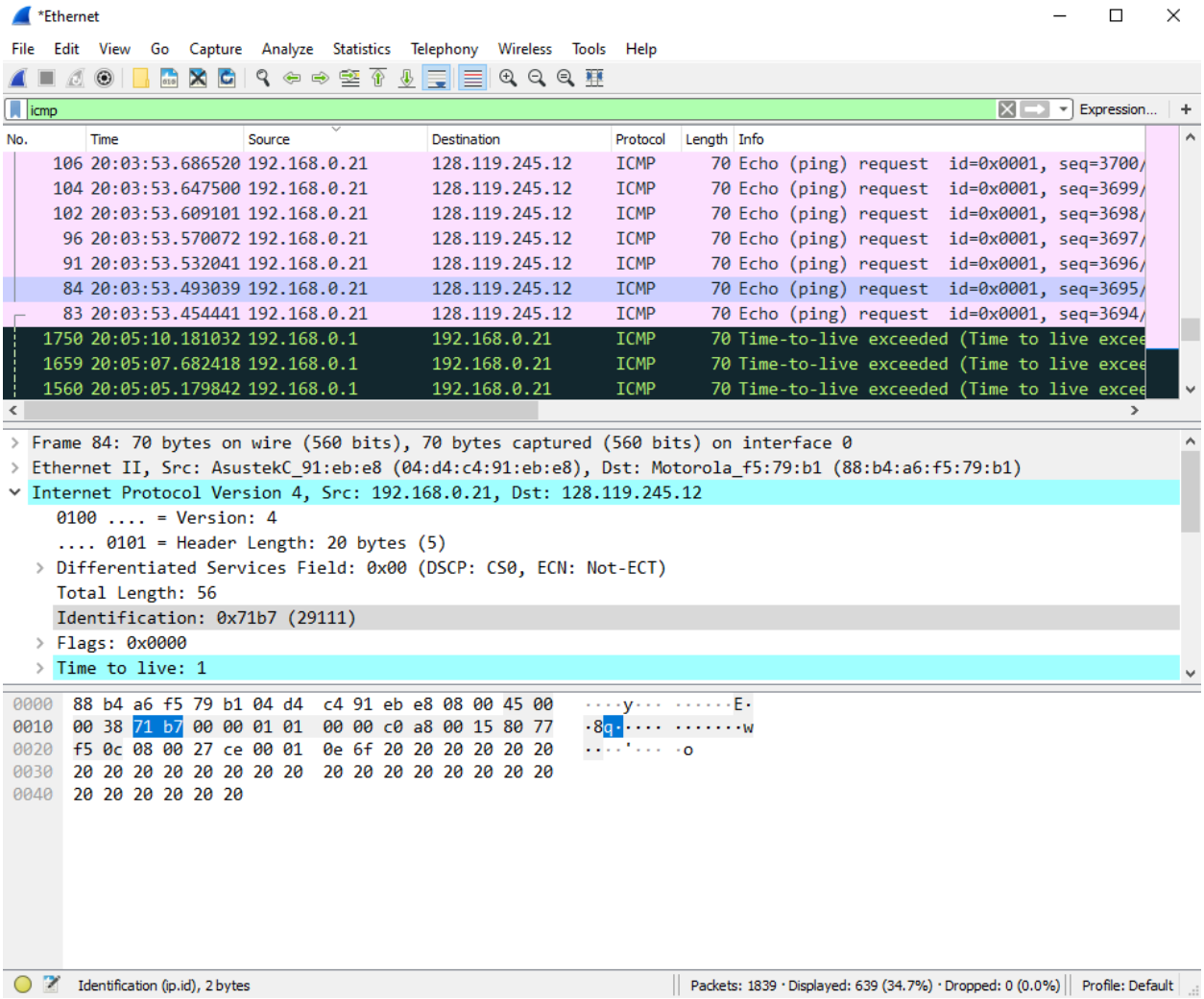
- IP version (using IPv4 for all packets)
- Header length (20bytes as ICMP protocol packets)
- Differentiated Services Field (0x00 (DSCP: CS0, ECN: Not-ECT), all packets are using same service class type)
- Upper layer protocol (ICMP (1), all packets are ICMP protocol packets)
- Source IP (192.168.0.21,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

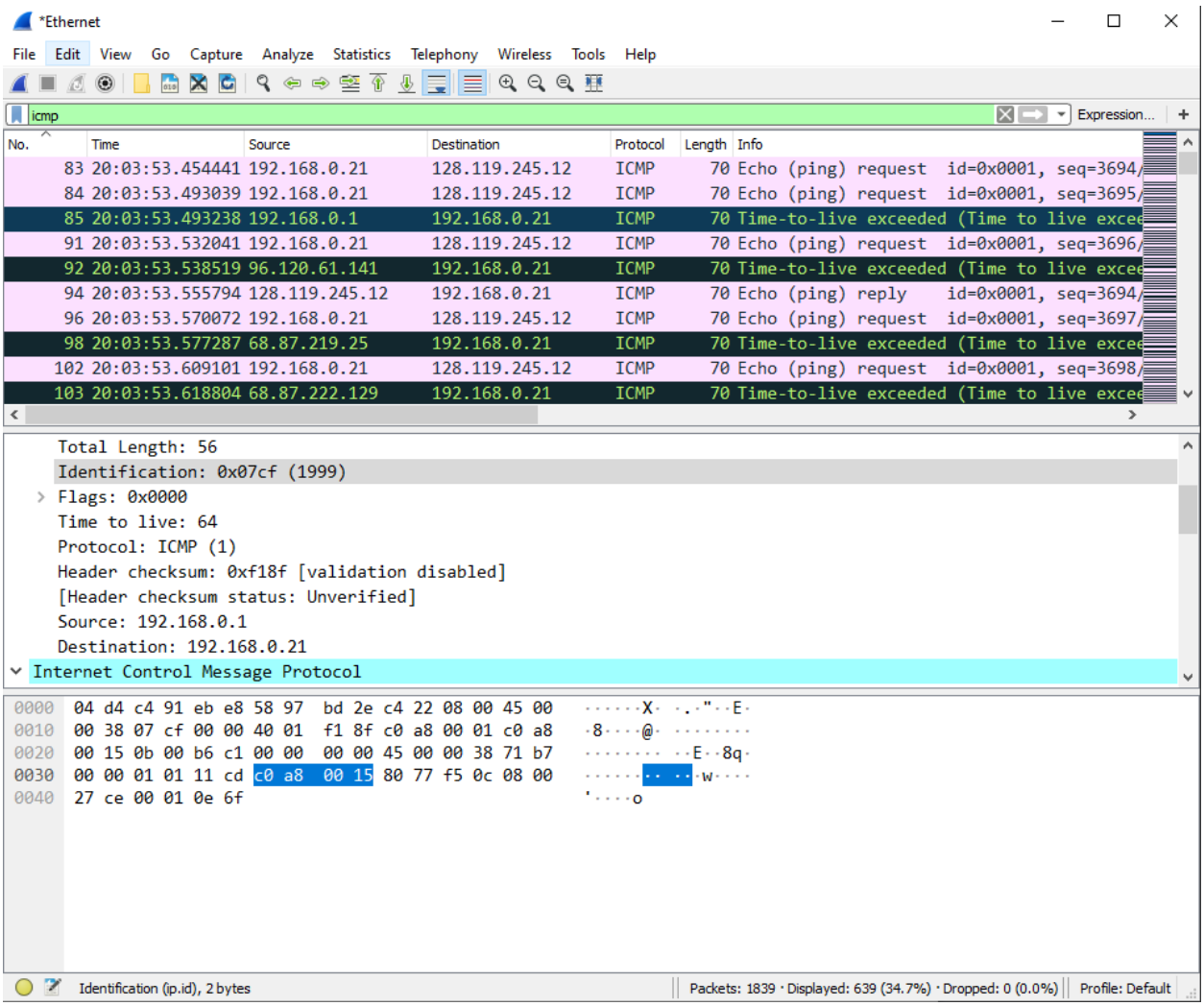
7. Describe the pattern you see in the values in the Identification field of the IP datagram





- The 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.

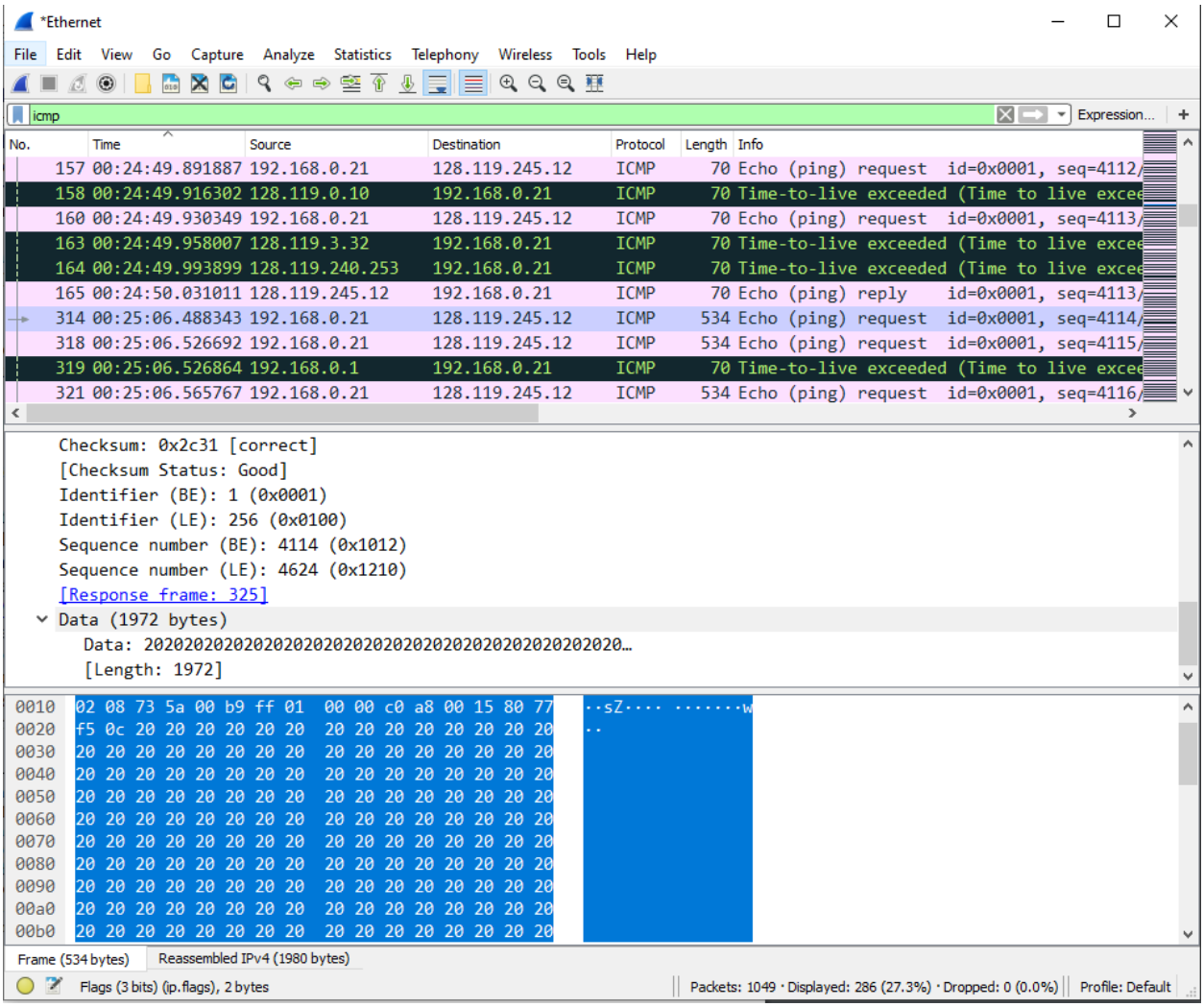
9. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

- 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.

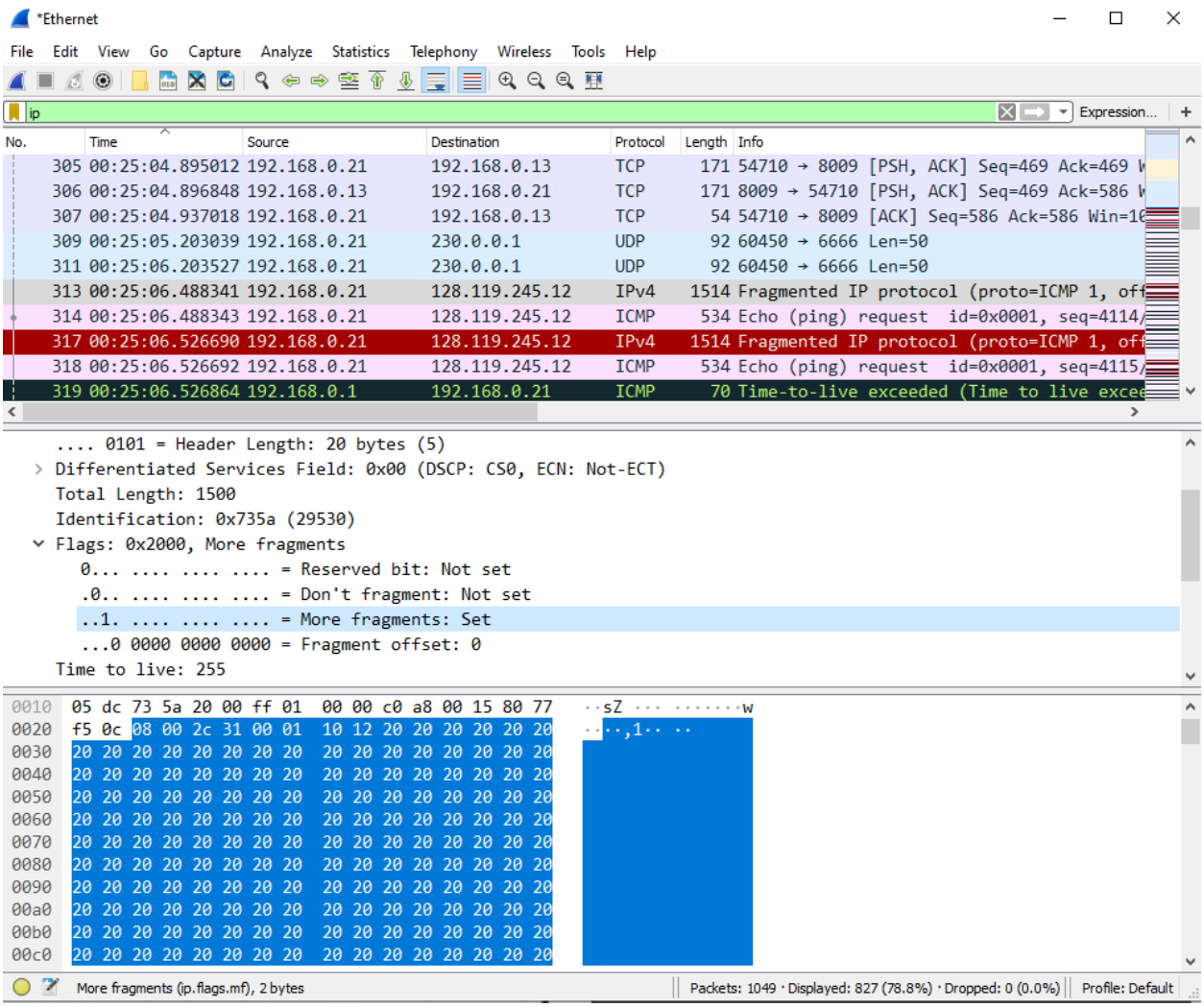
10. Find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 2000. Has that message been fragmented across more than one IP datagram? [Note: if you find your packet has not been fragmented, you should download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip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

ipethereal-trace-1packet trace. If your computer has an Ethernet interface, a packet size of 2000 should cause fragmentation.]



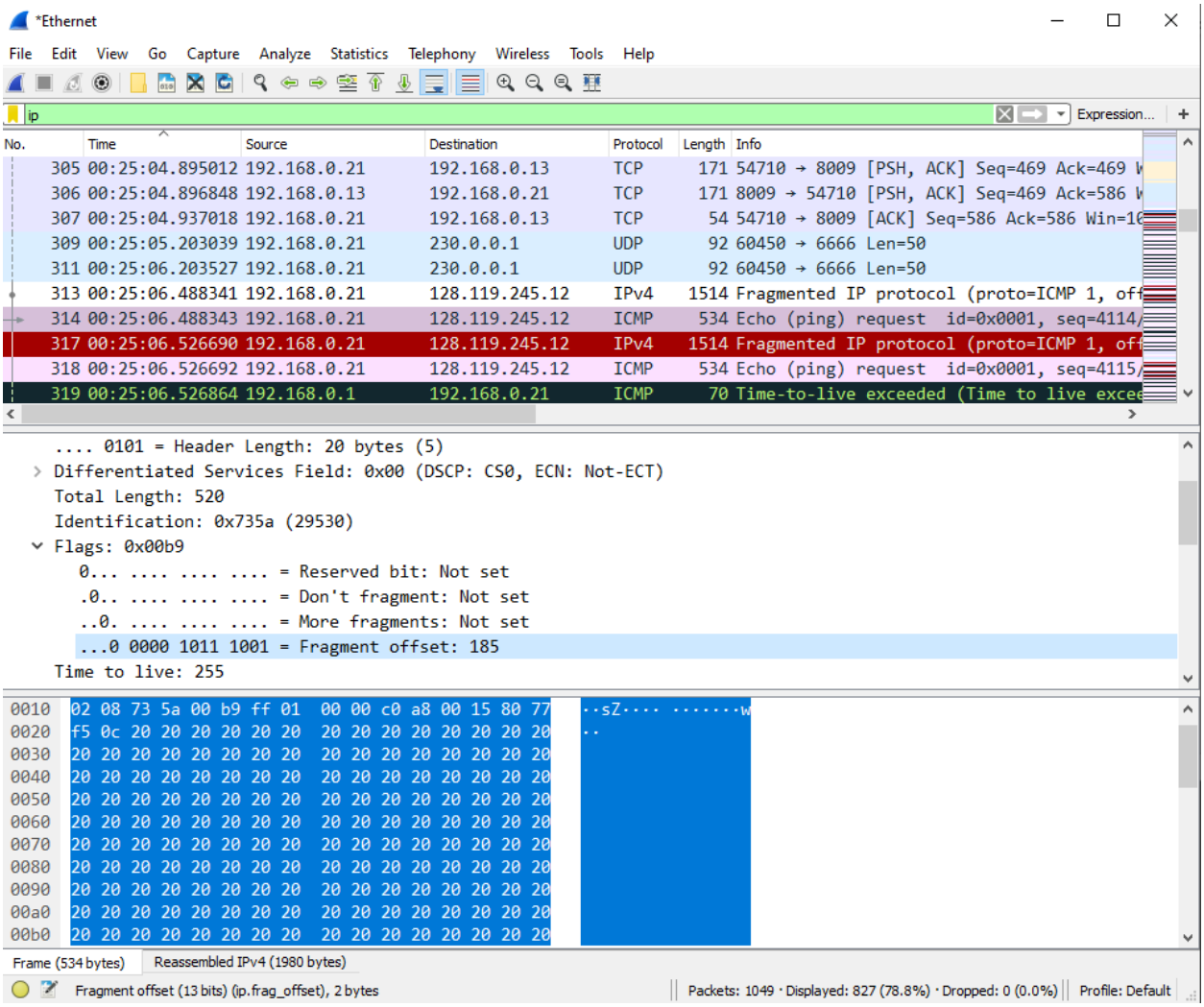
- Yes, the packet number 314 is the first ICMP Echo Request message that was sent by my computer after I changed the Packet Size in pingplotter to be 2000 and this packet has been fragmented across more than one IP datagram (above screenshots).

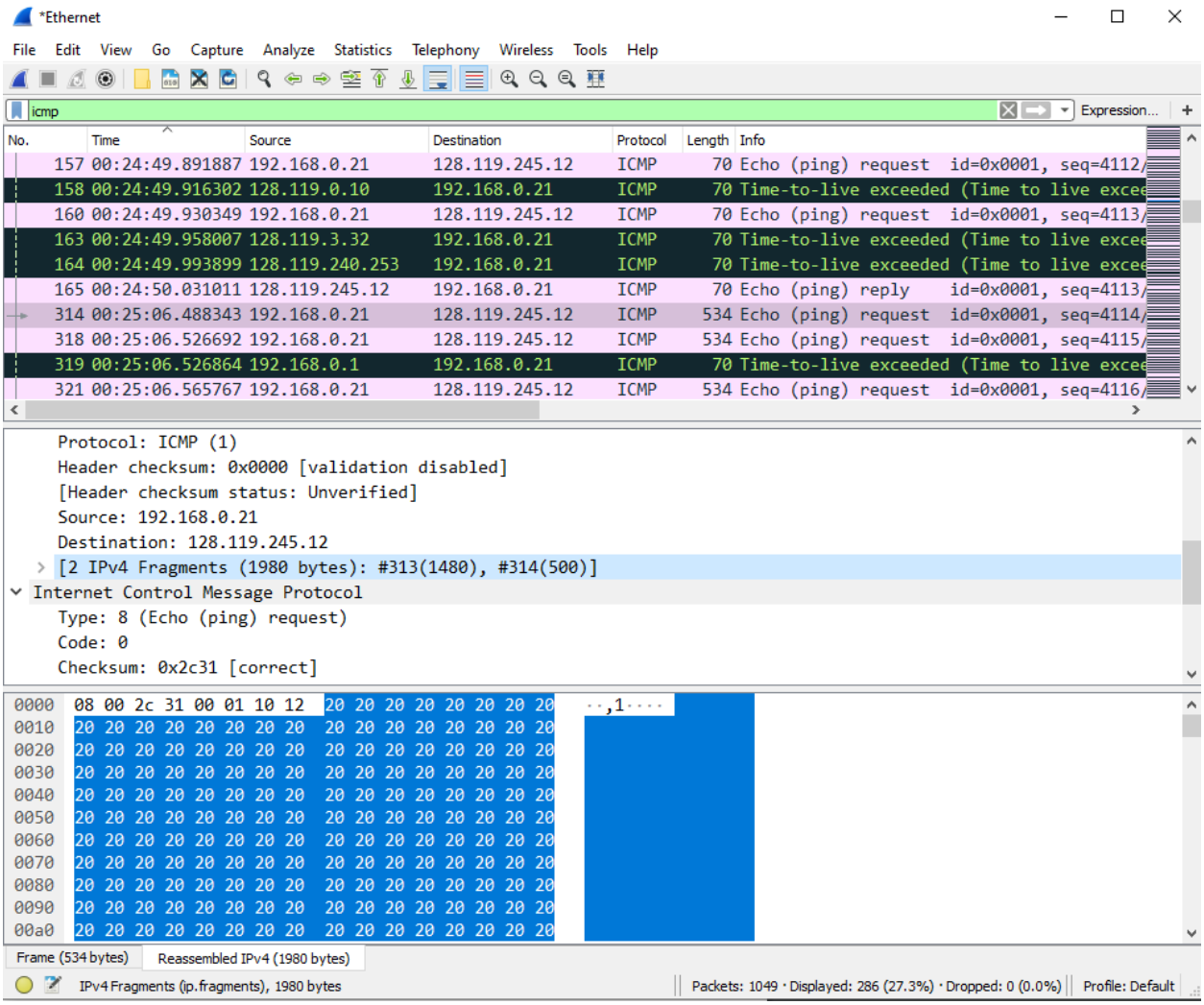
11. Screenshot the first fragment of the fragmented IP datagram (with sufficient details to answer these questions). What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment?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



- 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
Expression...
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
306 00:25:04.896848 192.168.0.13 192.168.0.21 TCP 171 8009 → 54710 [PSH, ACK] Seq=469 Ack=586
307 00:25:04.937018 192.168.0.21 192.168.0.13 TCP 54 54710 → 8009 [ACK] Seq=586 Ack=586 Win=1
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 exce
.... 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···· ·····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
0050 20 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 20
0070 20 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 20
0090 20 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 20
00b0 20 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?

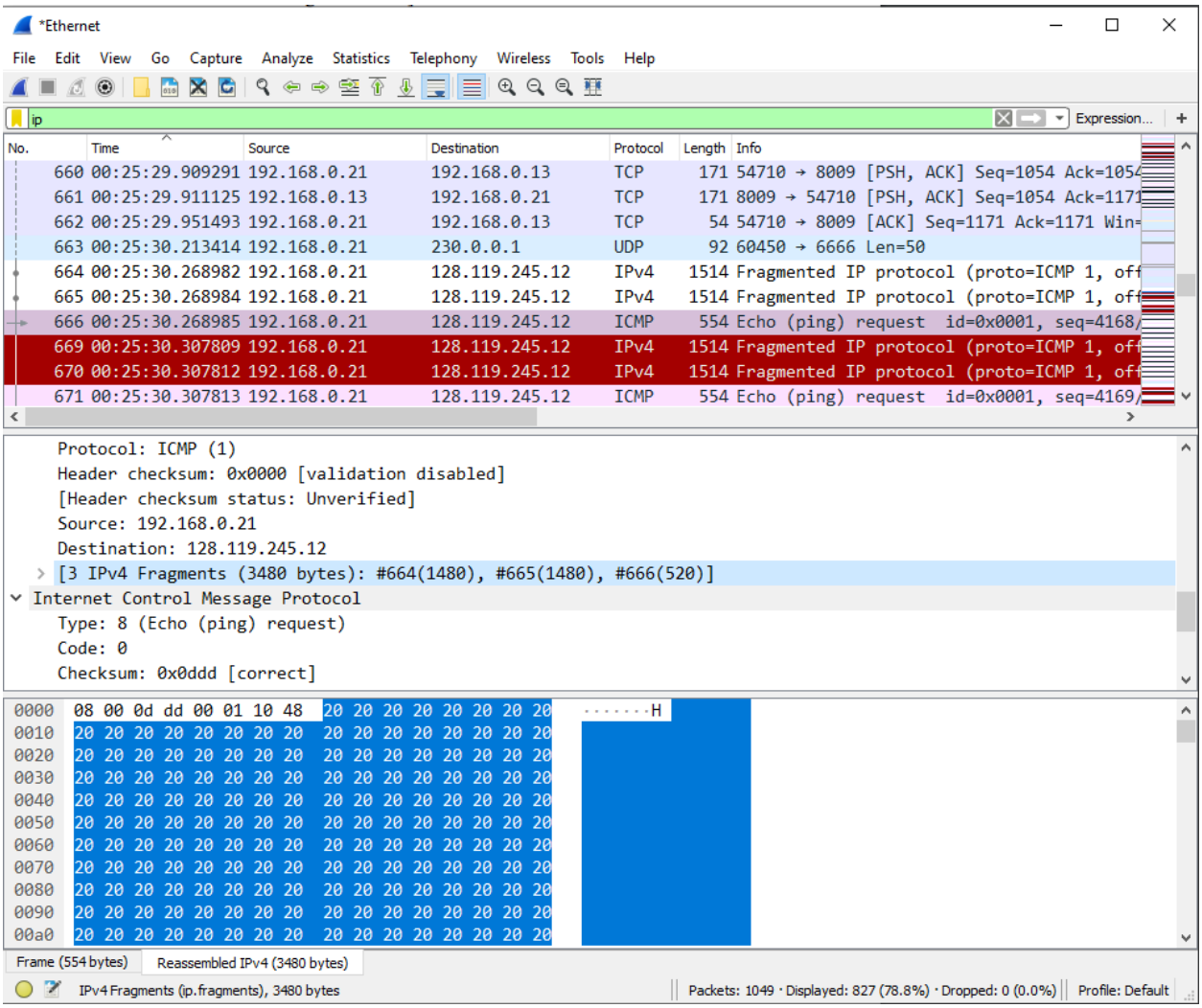
- When I compared the two fragments (#313 and #314),

- 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.

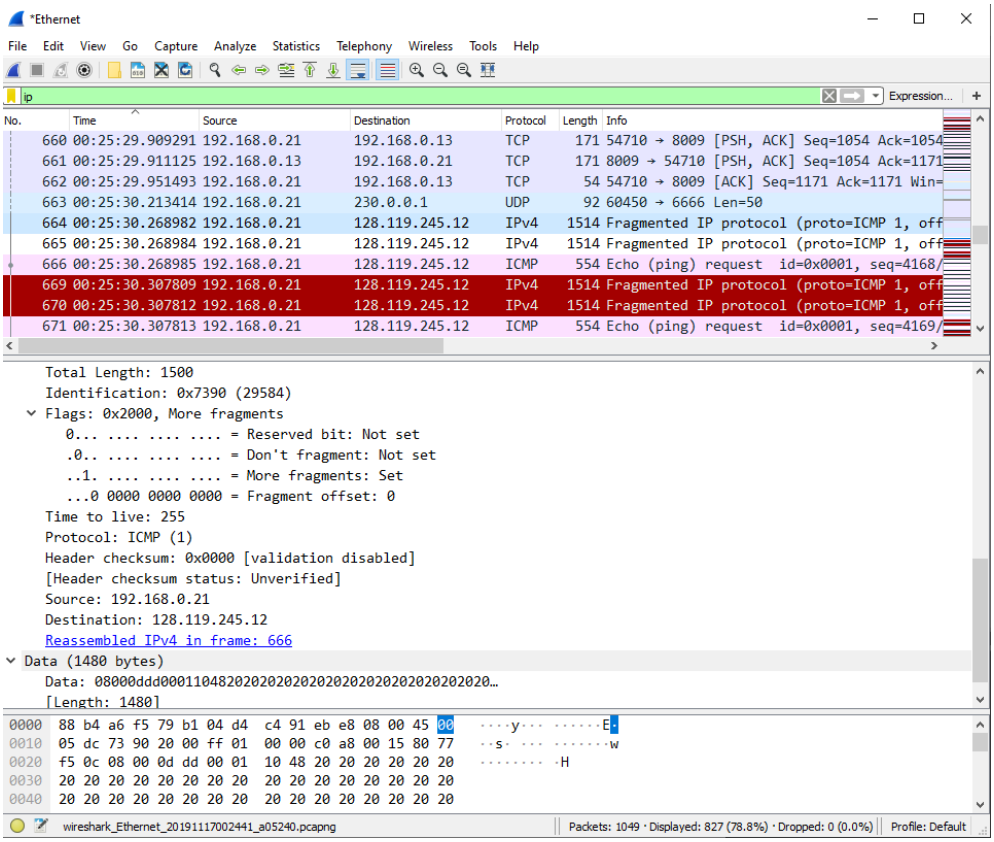
Now find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 3500.

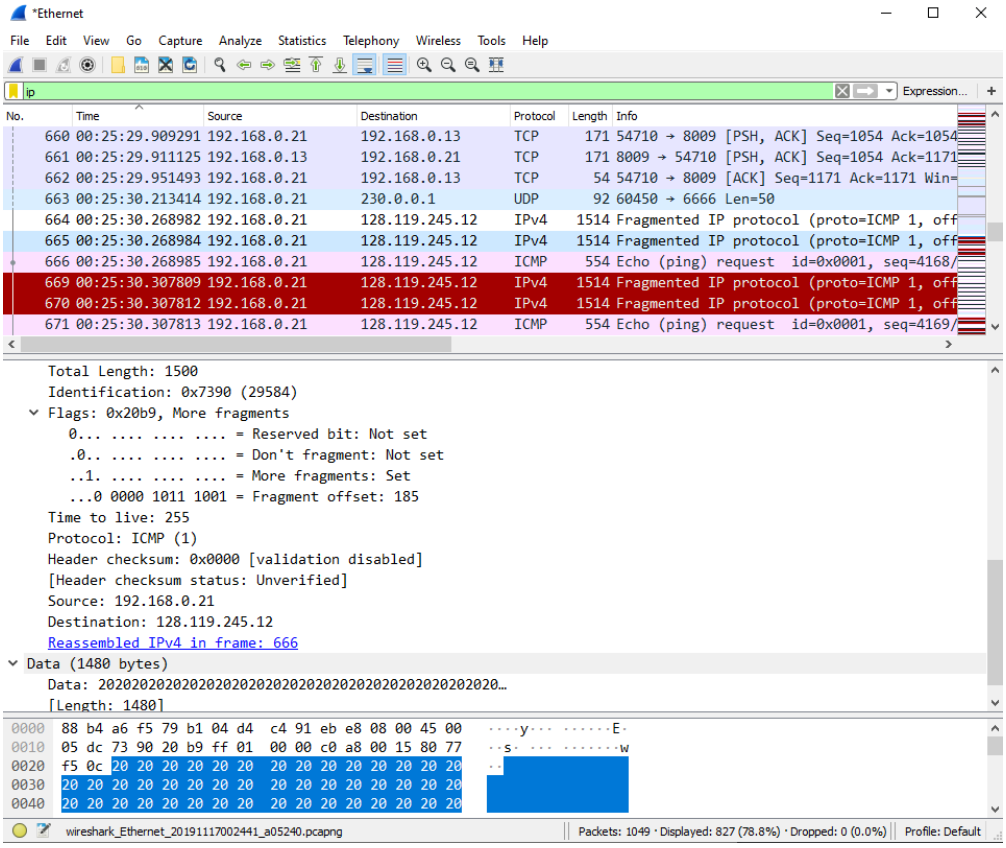
14. How many fragments were created from the original datagram?

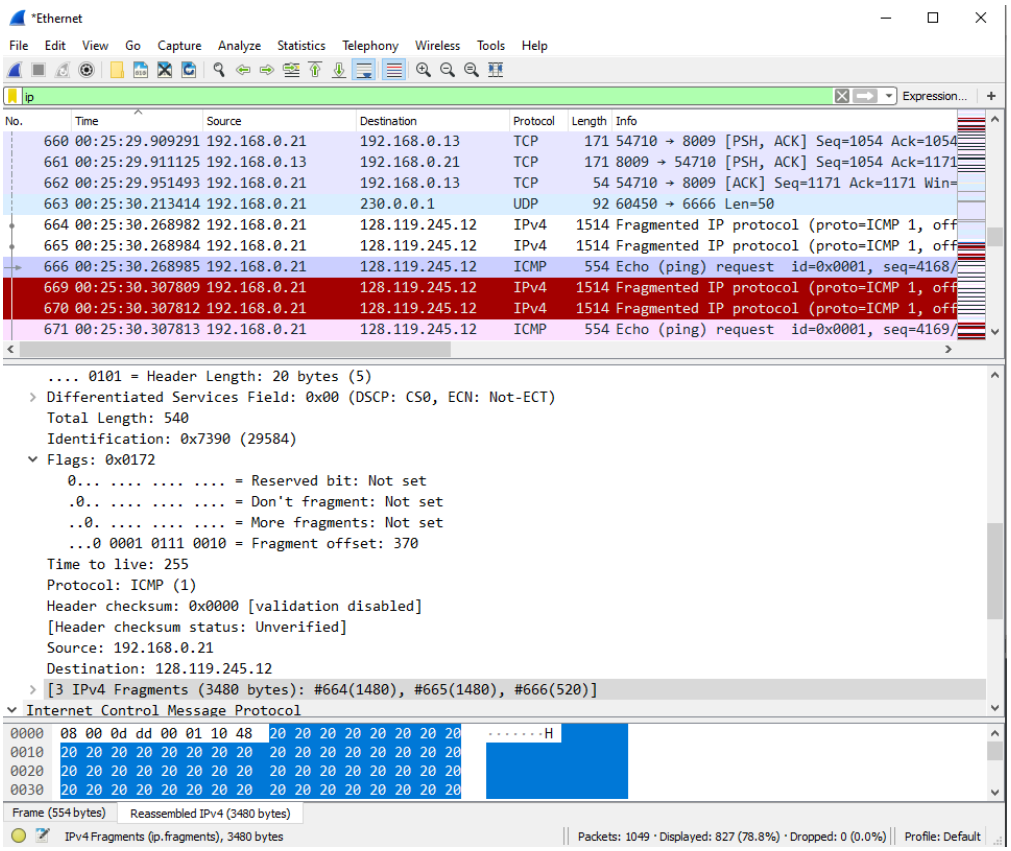


- There are 3 fragments (#664 packet 1480 bytes, #665 packet 1480bytes, #666 packet 520bytes) were created from original datagram after I changed the Packet Size in pingplotter to be 3500.

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


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip
Expression...
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/
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: 08000ddd00011048202020202020202020202020202020202020…
[Length: 1480]
0000 88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00
0010 05 dc 73 90 20 00 ff 01 00 00 c0 a8 00 15 80 77
0020 f5 0c 08 00 0d dd 00 01 10 48 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


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip
Expression...
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/
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: 2020202020202020202020202020202020202020202020202020…
[Length: 1480]
0000 88 b4 a6 f5 79 b1 04 d4 c4 91 eb e8 08 00 45 00
0010 05 dc 73 90 20 b9 ff 01 00 00 c0 a8 00 15 80 77
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



- 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

server.c

C: > Users > 15419 > AppData > Local > Temp > Mxt121 > RemoteFiles > 132778_7_6 > server.c

```c
//written by Junhyeok Jeong

// server code for UDP socket programming
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>

#define IP_PROTOCOL 0
#define PORT_NO 5000
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0
#define nofile "File Not Found!"

// function to clear buffer
void clearBuf(char* b)
{
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '\0';
}

// function to encrypt
char Cipher(char ch)
{
    return ch ^ cipherKey;
}

// function sending file
int sendFile(FILE* fp, char* buf, int s)
{
    int i, len;
    if (fp == NULL) {
        strcpy(buf, nofile);
        len = strlen(nofile);
        buf[len] = EOF;
        for (i = 0; i <= len; i++)
            buf[i] = Cipher(buf[i]);
        return 1;
    }

    char ch, ch2;
    for (i = 0; i < s; i++) {
        ch = fgetc(fp);
```

```c
    char ch, ch2;
    for (i = 0; i < s; i++) {
        ch = fgetc(fp);
        ch2 = Cipher(ch);
        buf[i] = ch2;
        if (ch == EOF)
            return 1;
    }
    return 0;
}

// driver code
int main()
{
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    int addrlen = sizeof(addr_con);
    addr_con.sin_family = AF_INET;
    addr_con.sin_port = htons(PORT_NO);
    addr_con.sin_addr.s_addr = INADDR_ANY;
    char net_buf[NET_BUF_SIZE];
    FILE* fp;

    // socket()
    sockfd = socket(AF_INET, SOCK_DGRAM, IP_PROTOCOL);

    if (sockfd < 0)
        printf("\nError: Check your socket!!\n");
    else
        printf("\nfile name will be received on socketfd %d \n", sockfd);

    // bind()
    if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(addr_con)) == 0)
        printf("\nSuccessfully binded!\n");
    else
        printf("\nBinding Failed!\n");

    while (1) {
        printf("\nlistening client connection and Waiting for file name...\n");

        // receive file name
        clearBuf(net_buf);

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

```c
        // 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

server.c | client.c

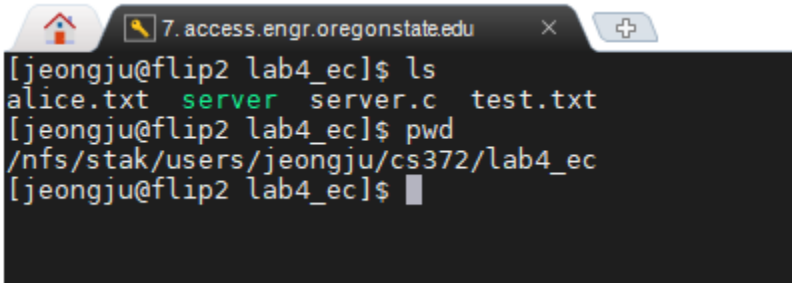
C: > Users > 15419 > AppData > Local > Temp > Mxt121 > RemoteFiles > 132778_7_7 > client.c > main()

```c
//written by Junhyeok Jeong

// client code for UDP socket programming
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>

#define IP_PROTOCOL 0
#define IP_ADDRESS "127.0.0.1" // localhost
#define PORT_NO 5000
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0

// function to clear buffer
void clearBuf(char* b)
{
    int i;
    for (i = 0; i < NET_BUF_SIZE; i++)
        b[i] = '\0';
}

// function for decryption
char Cipher(char ch)
{
    return ch ^ cipherKey;
}

// function to receive file
int recvFile(char* buf, int s, FILE *write_file)
{
    int i;
    char ch;

    for (i = 0; i < s; i++) {
        ch = buf[i];
        ch = Cipher(ch);
        if (ch == EOF)
            return 1;
        else
            fprintf(write_file, "%c", ch);
            printf("%c", ch);
```

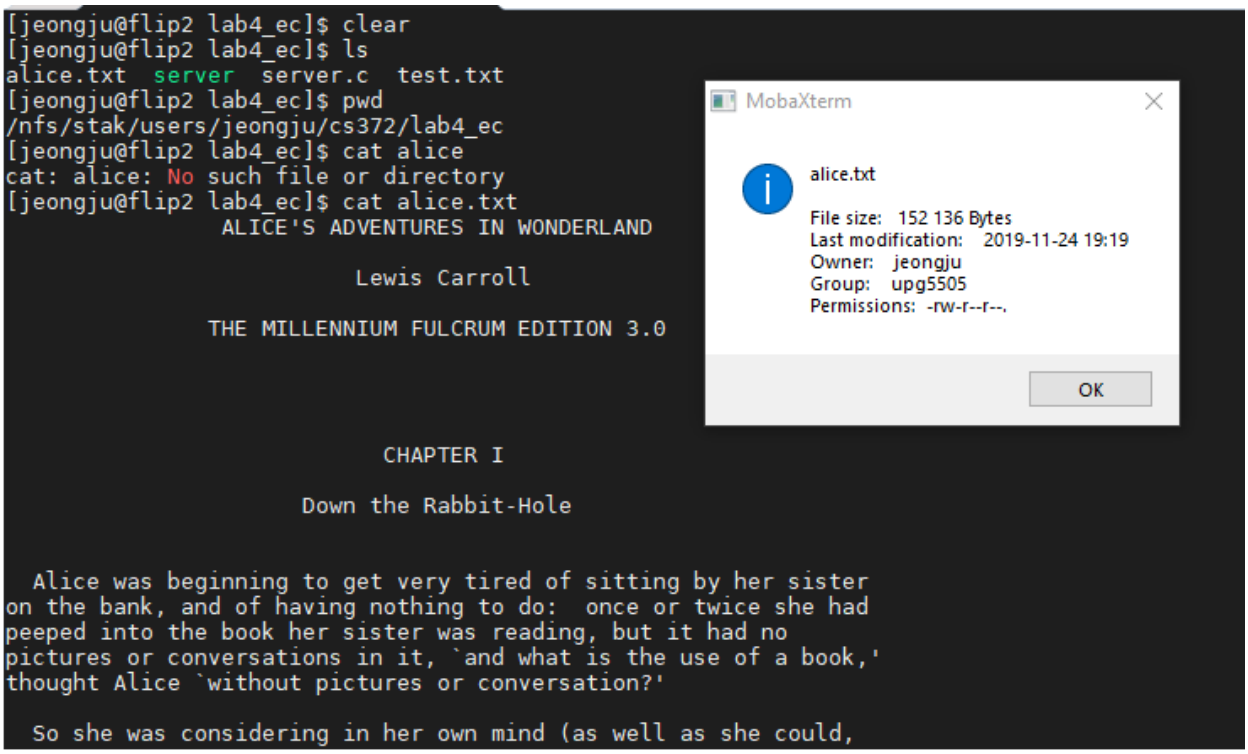
```c
// driver code
int main()
{
    int sockfd, nBytes;
    struct sockaddr_in addr_con;
    int addrlen = sizeof(addr_con);
    addr_con.sin_family = AF_INET;
    addr_con.sin_port = htons(PORT_NO);
    addr_con.sin_addr.s_addr = inet_addr(IP_ADDRESS);
    char net_buf[NET_BUF_SIZE];
    FILE *write_file;

    // socket()
    sockfd = socket(AF_INET, SOCK_DGRAM,
                    IP_PROTOCOL);

    if (sockfd < 0)
        printf("\nError: check your socket!!\n");
    else
        printf("\nfile will be received on socketfd %d\n", sockfd);

    while (1) {
        printf("\nPlease enter file name to receive:\n");
        scanf("%s", net_buf);
        write_file = fopen(net_buf, "w");

        sendto(sockfd, net_buf, NET_BUF_SIZE,
               sendrecvflag, (struct sockaddr*)&addr_con,
               addrlen);

        printf("\n--------Data Received--------\n");

        while (1) {
            // receive
            clearBuf(net_buf);
            nBytes = recvfrom(sockfd, net_buf, NET_BUF_SIZE,
                              sendrecvflag, (struct sockaddr*)&addr_con,
                              &addrlen);

            // process
            if (recvFile(net_buf, NET_BUF_SIZE, write_file)) {
                break;
            }
        }
        printf("\n------------------------------\n");
```

```c
        printf("\nthe requested file is received and stored successfully! And socket connection is end\n");
        break;
    }


    return 0;
}

```

3. Server.c is in different directory



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



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 socketfd 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 socketfd 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;
```