Junhyeok Jeong

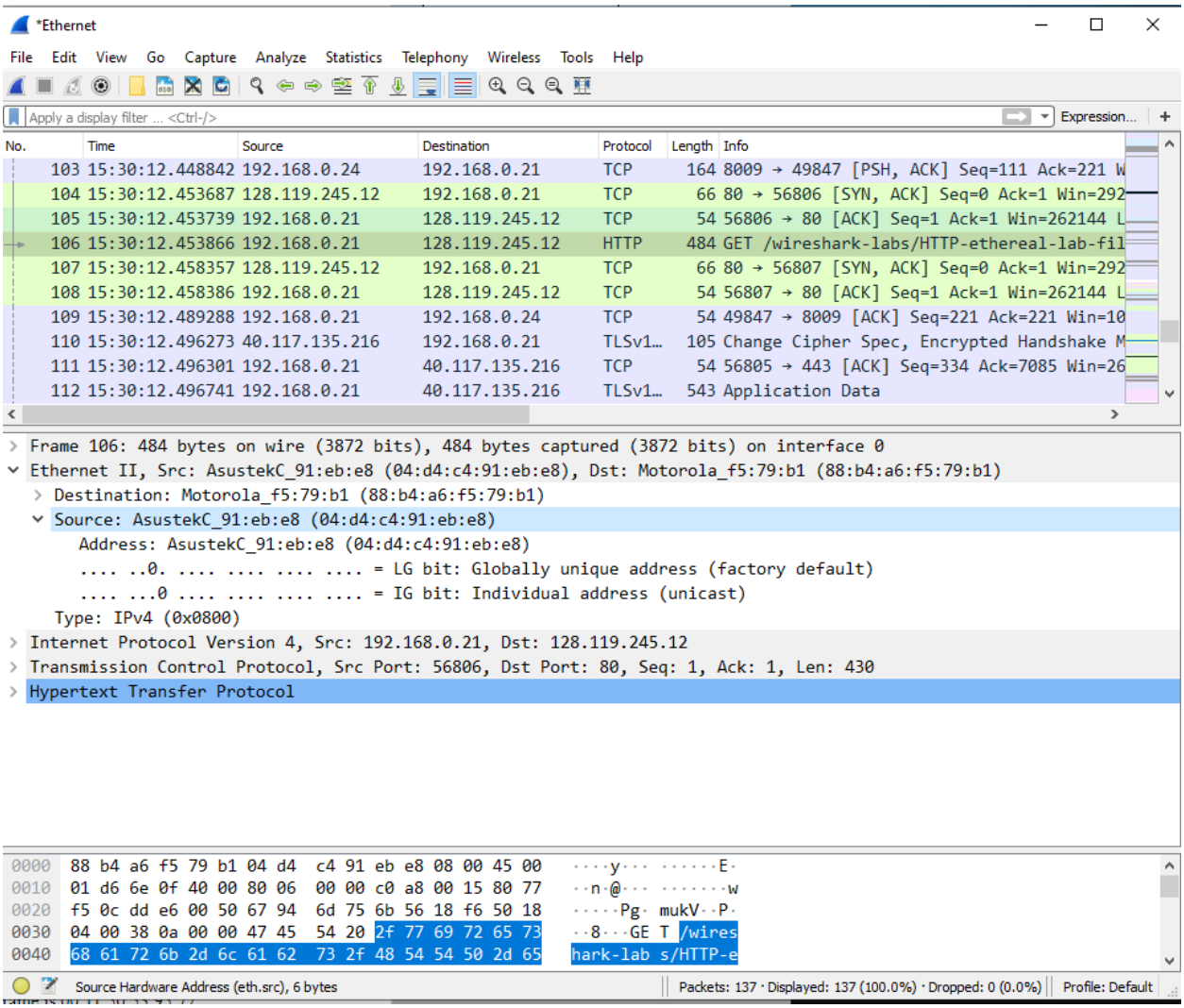
Bill Pfeil

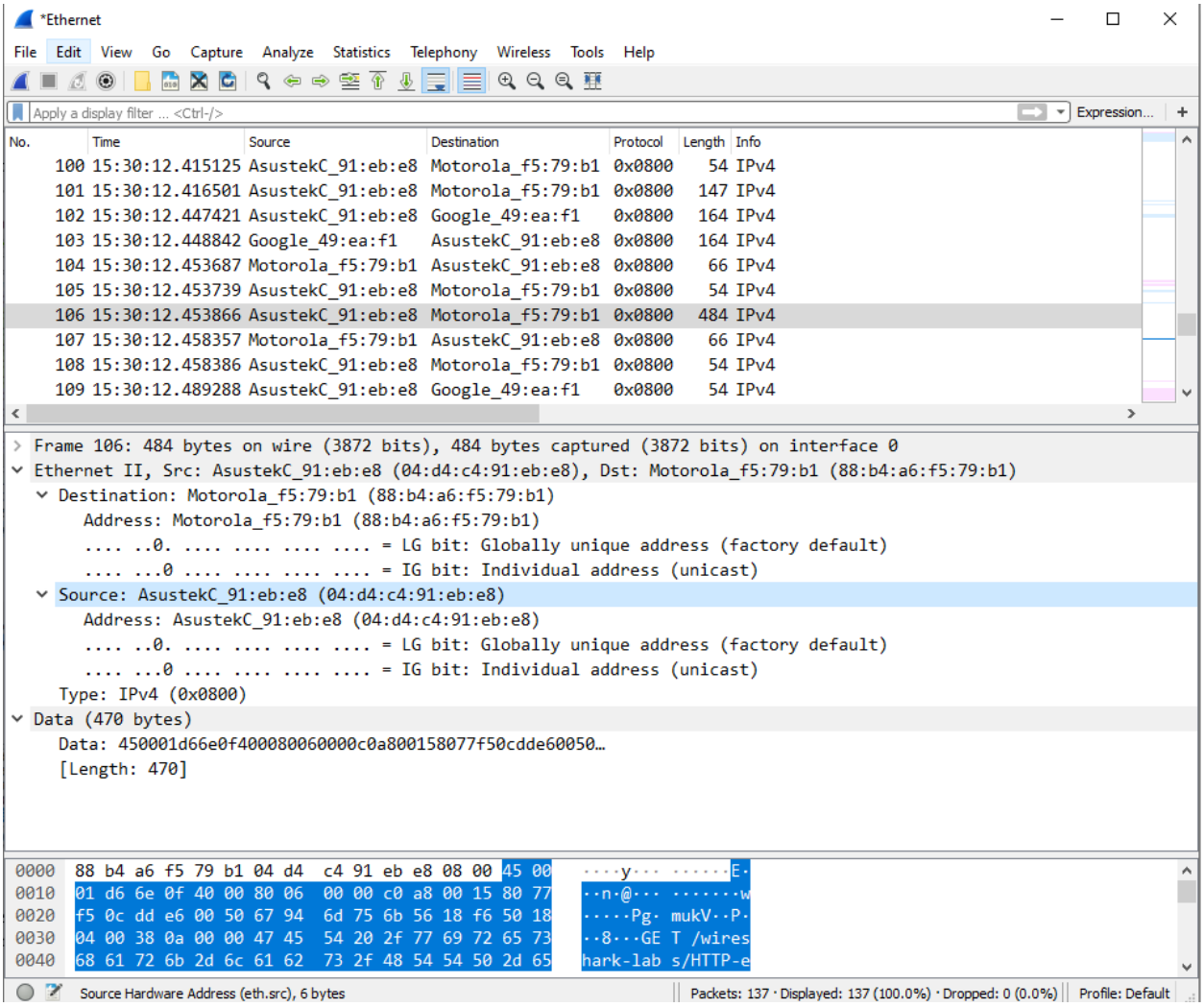
CS 372-X001

12/6/2019

Lab 5

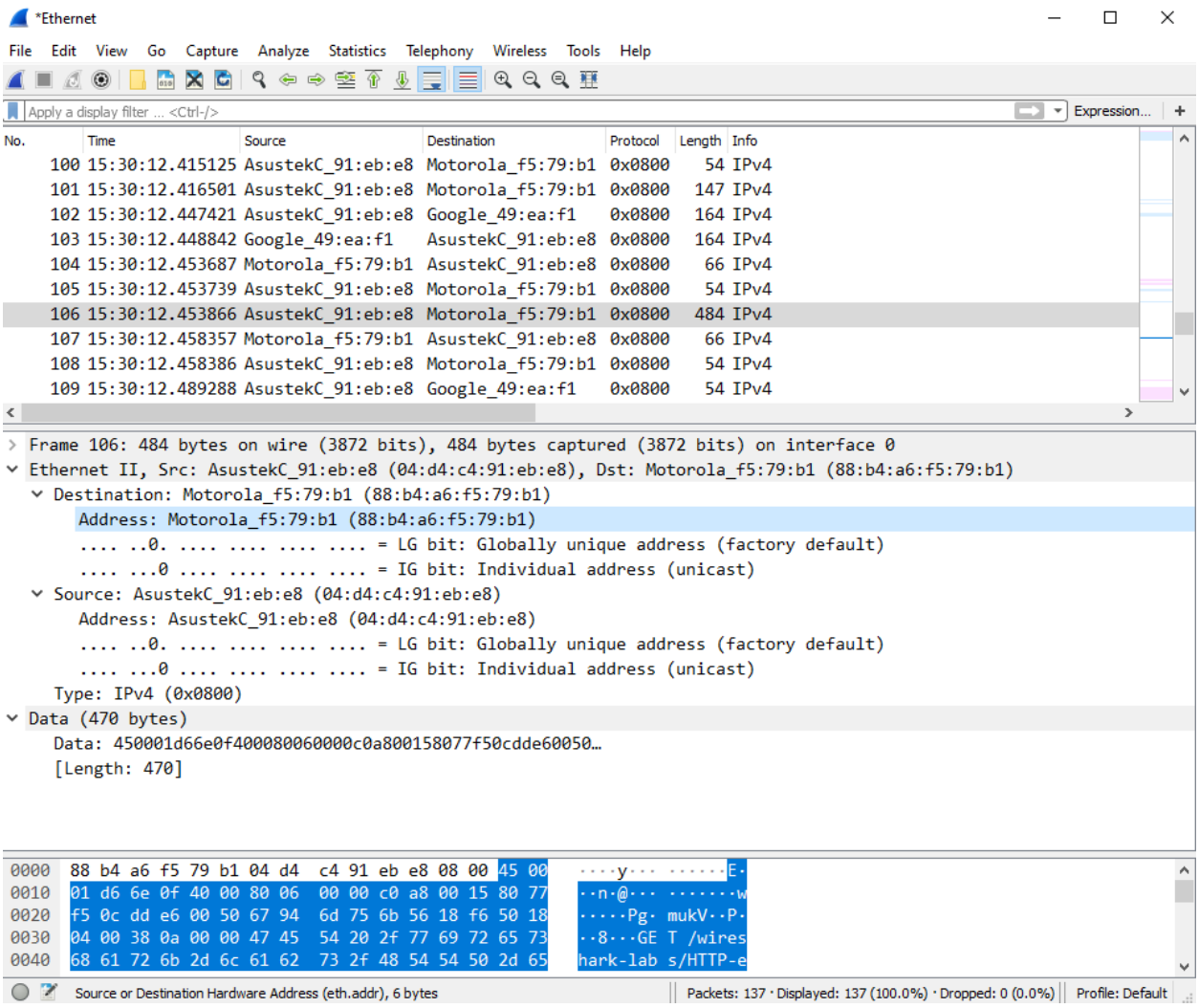
1. What is the 48-bit Ethernet address of your computer?





- My computer’s 48-bit Ethernet address is 04:d4:c4:91:eb:e8

2. What is the 48-bit destination address in the Ethernet frame? Is this the Ethernet address of gaia.cs.umass.edu? (Hint: the answer is no). What device has this as its Ethernet address? [Note: this is an important question, and one that students sometimes get wrong. Re-read pages 468-469 in the text and make sure you understand the answer here.]

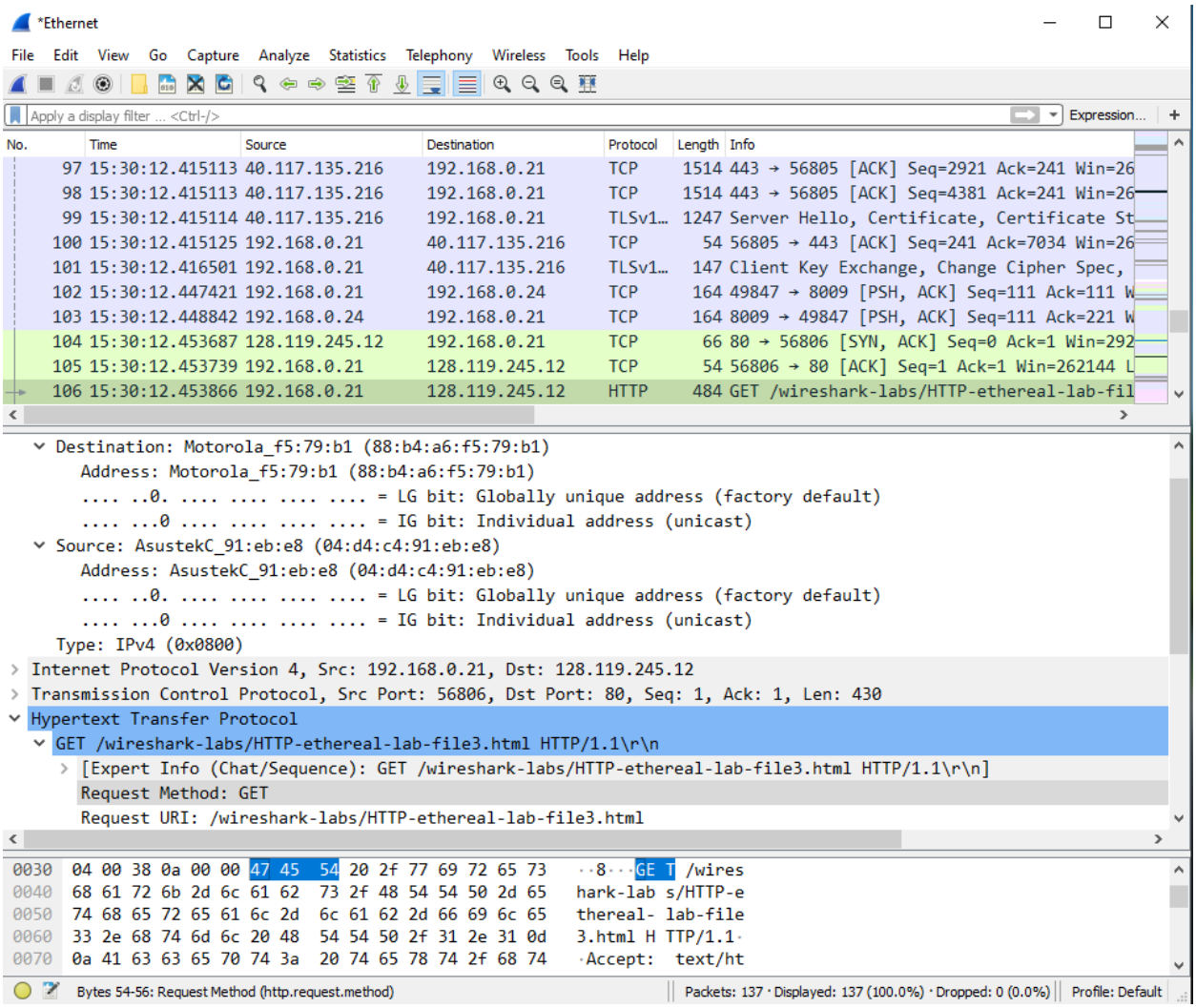


- The 48-bit destination address is 88:b4:a6:f5:79:b1. This is not the Ethernet address of gaia.cs.umass.edu. This 48-bit Ethernet address is for my router (Motorola router) to escape the subnet.

3. Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?

- The hexadecimal value for the two-byte Frame type field is 0x0800. This value corresponds to the IP protocol because the two-byte Frame type field will be passed the IP protocol layer as payload.

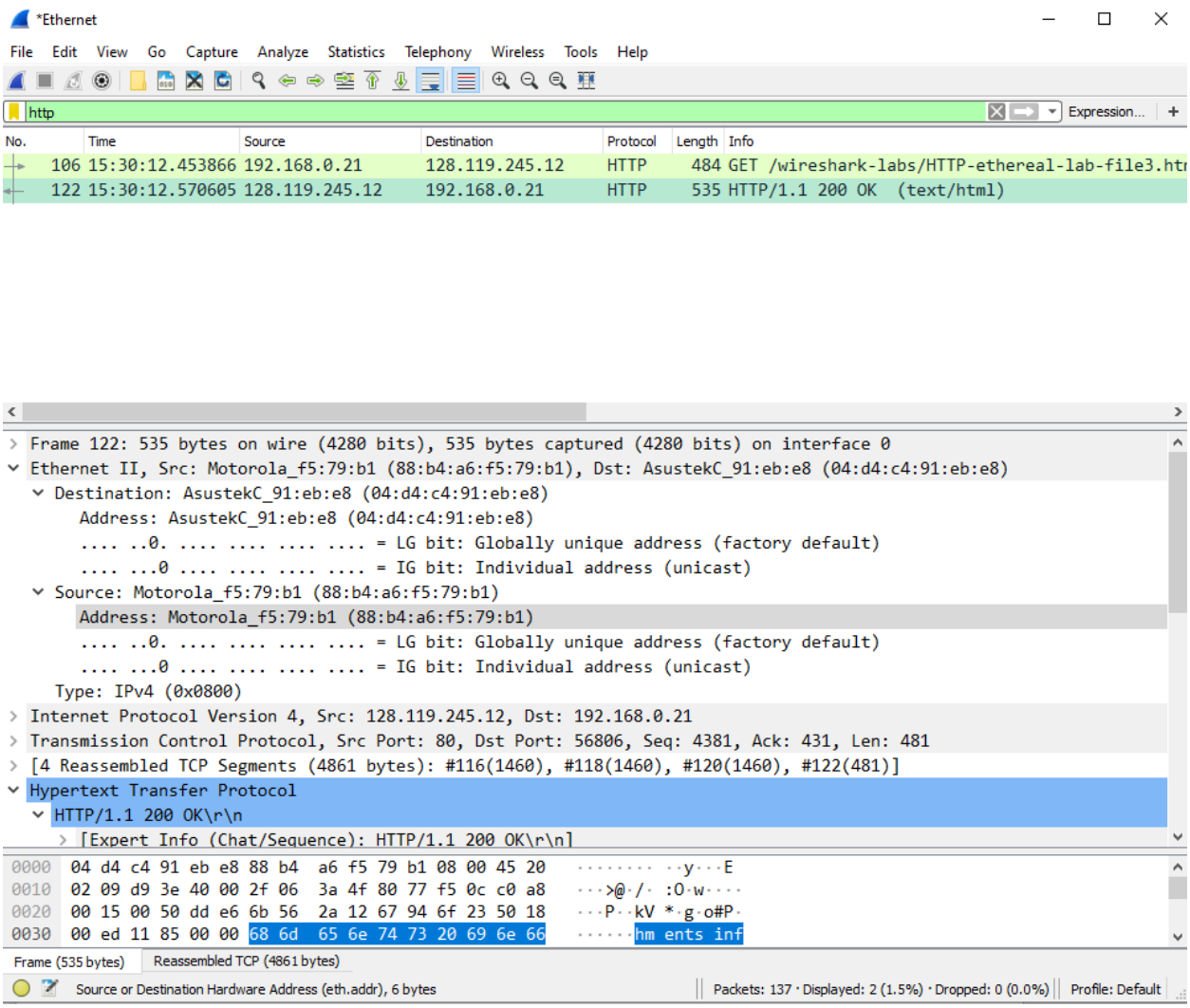
4. How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the Ethernet frame?

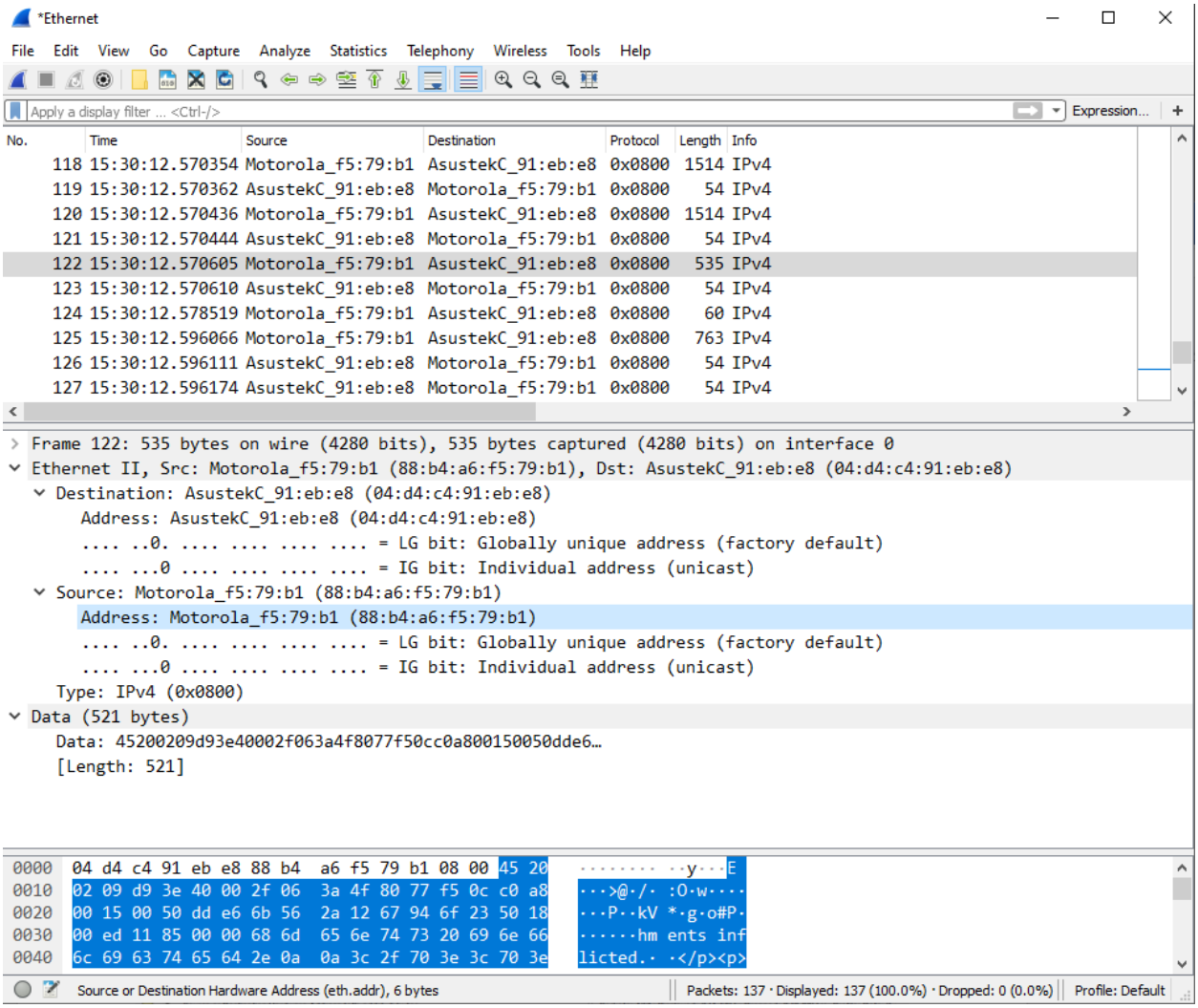


- After 54 bytes the ASCII “G” in “GET” appear in the Ethernet frame.

Next, answer the following questions, based on the contents of the Ethernet frame containing the first byte of the HTTP response message.

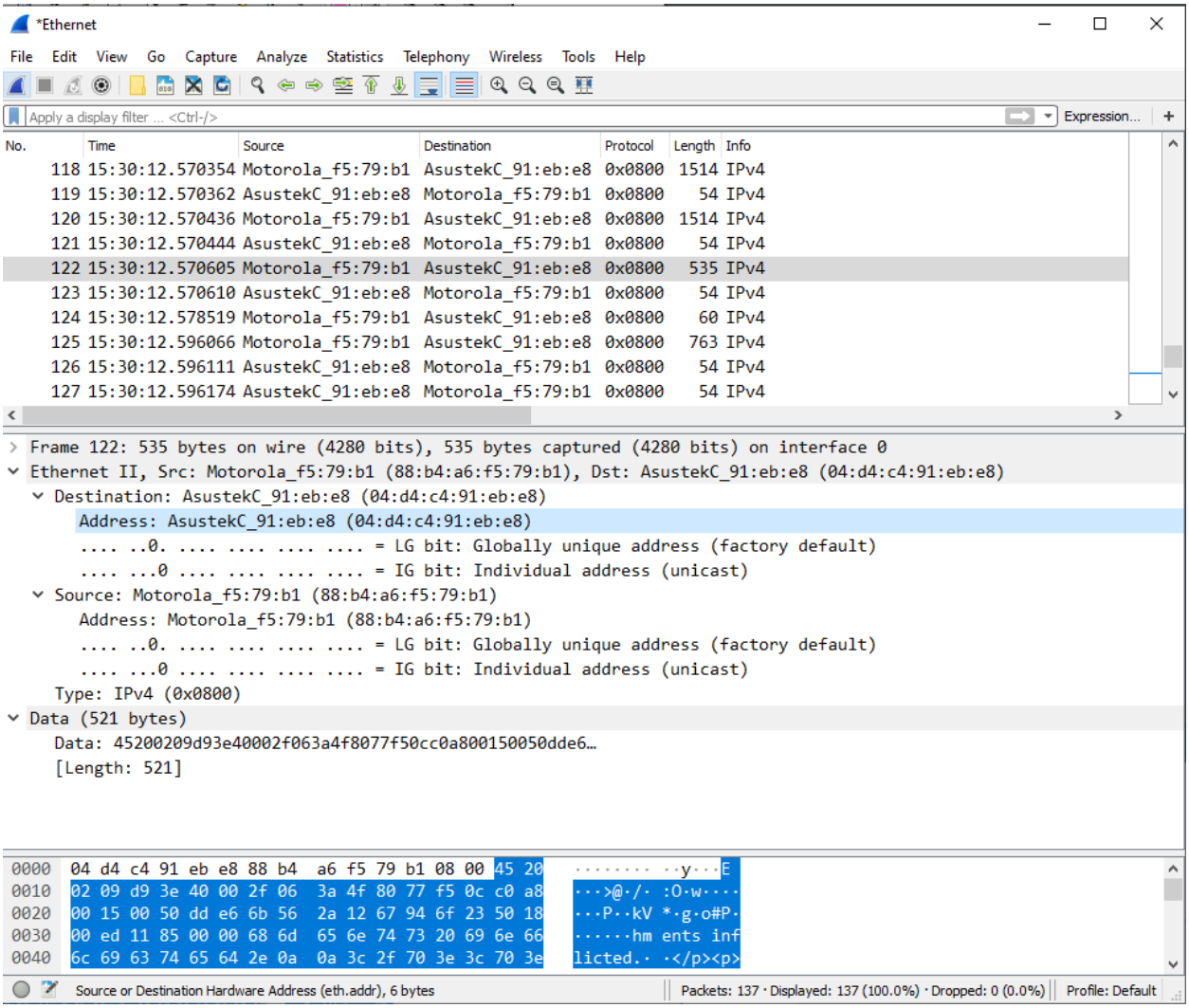
5. What is the value of the Ethernet source address? Is this the address of your computer, or of gaia.cs.umass.edu (Hint: the answer is no). What device has this as its Ethernet address?


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...
No. Time Source Destination Protocol Length Info
106 15:30:12.453866 192.168.0.21 128.119.245.12 HTTP 484 GET /wireshark-labs/HTTP-ethereal-lab-file3.htr
122 15:30:12.570605 128.119.245.12 192.168.0.21 HTTP 535 HTTP/1.1 200 OK (text/html)
Frame 122: 535 bytes on wire (4280 bits), 535 bytes captured (4280 bits) on interface 0
Ethernet II, Src: Motorola_f5:79:b1 (88:b4:a6:f5:79:b1), Dst: AsustekC_91:eb:e8 (04:d4:c4:91:eb:e8)
Destination: AsustekC_91:eb:e8 (04:d4:c4:91:eb:e8)
Address: AsustekC_91:eb:e8 (04:d4:c4:91:eb:e8)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Source: Motorola_f5:79:b1 (88:b4:a6:f5:79:b1)
Address: Motorola_f5:79:b1 (88:b4:a6:f5:79:b1)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.0.21
Transmission Control Protocol, Src Port: 80, Dst Port: 56806, Seq: 4381, Ack: 431, Len: 481
[4 Reassembled TCP Segments (4861 bytes): #116(1460), #118(1460), #120(1460), #122(481)]
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
[Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
0000 04 d4 c4 91 eb e8 88 b4 a6 f5 79 b1 08 00 45 20 ....y...E
0010 02 09 d9 3e 40 00 2f 06 3a 4f 80 77 f5 0c c0 a8 ...>@./. :O.w....
0020 00 15 00 50 dd e6 6b 56 2a 12 67 94 6f 23 50 18 ...P..kV *.g.o#P.
0030 00 ed 11 85 00 00 68 6d 65 6e 74 73 20 69 6e 66 ......hm ents inf
Frame (535 bytes)
Reassembled TCP (4861 bytes)
Source or Destination Hardware Address (eth.addr), 6 bytes
Packets: 137 · Displayed: 2 (1.5%) · Dropped: 0 (0.0%)
Profile: Default



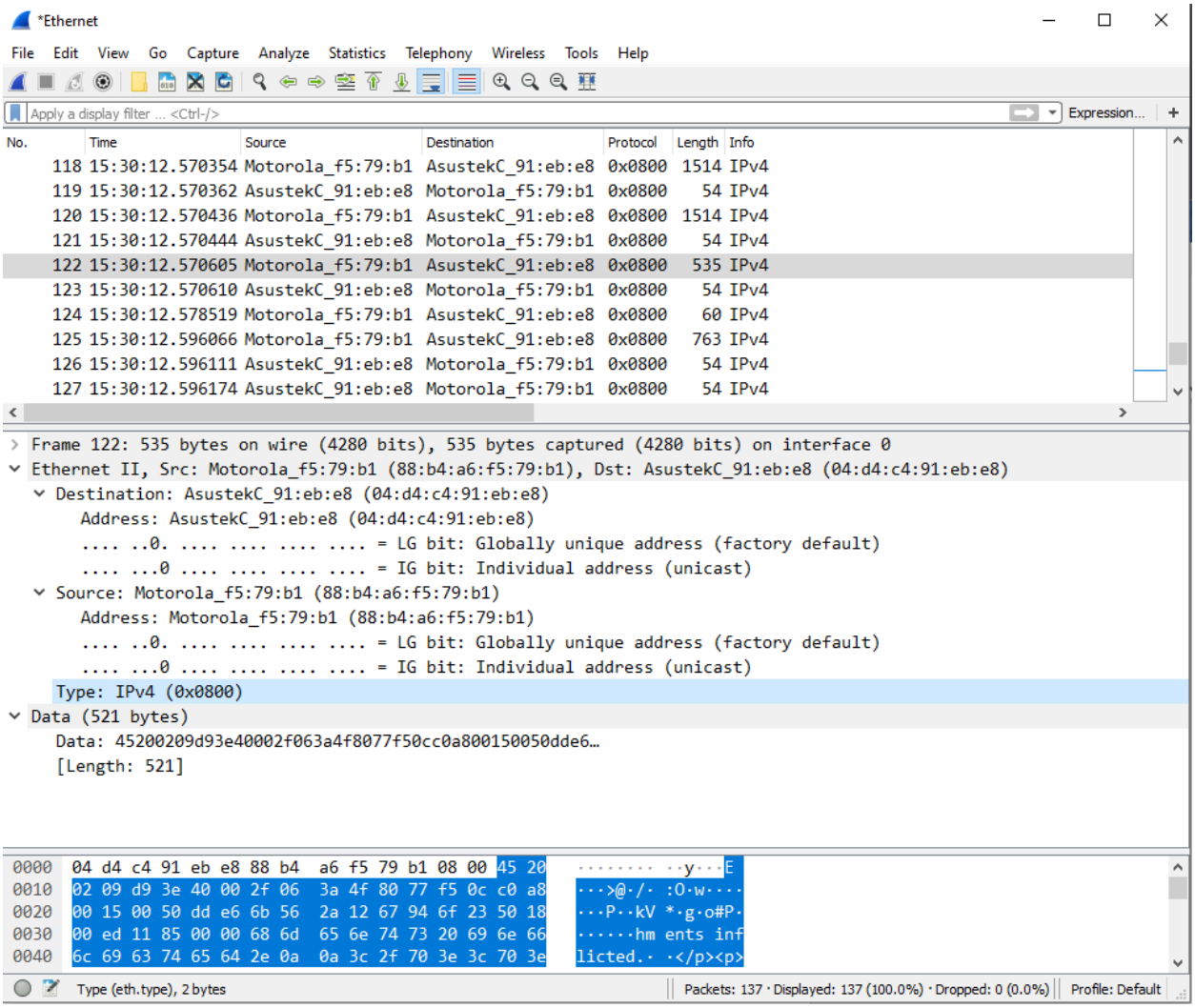
- The value of Ethernet source address is 88:b4:a6:f5:79:b1. This value is not the address of my computer nor of gaia.cs.umass.edu because it is the address of my Motorola router for getting on the subnet.

6. What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?



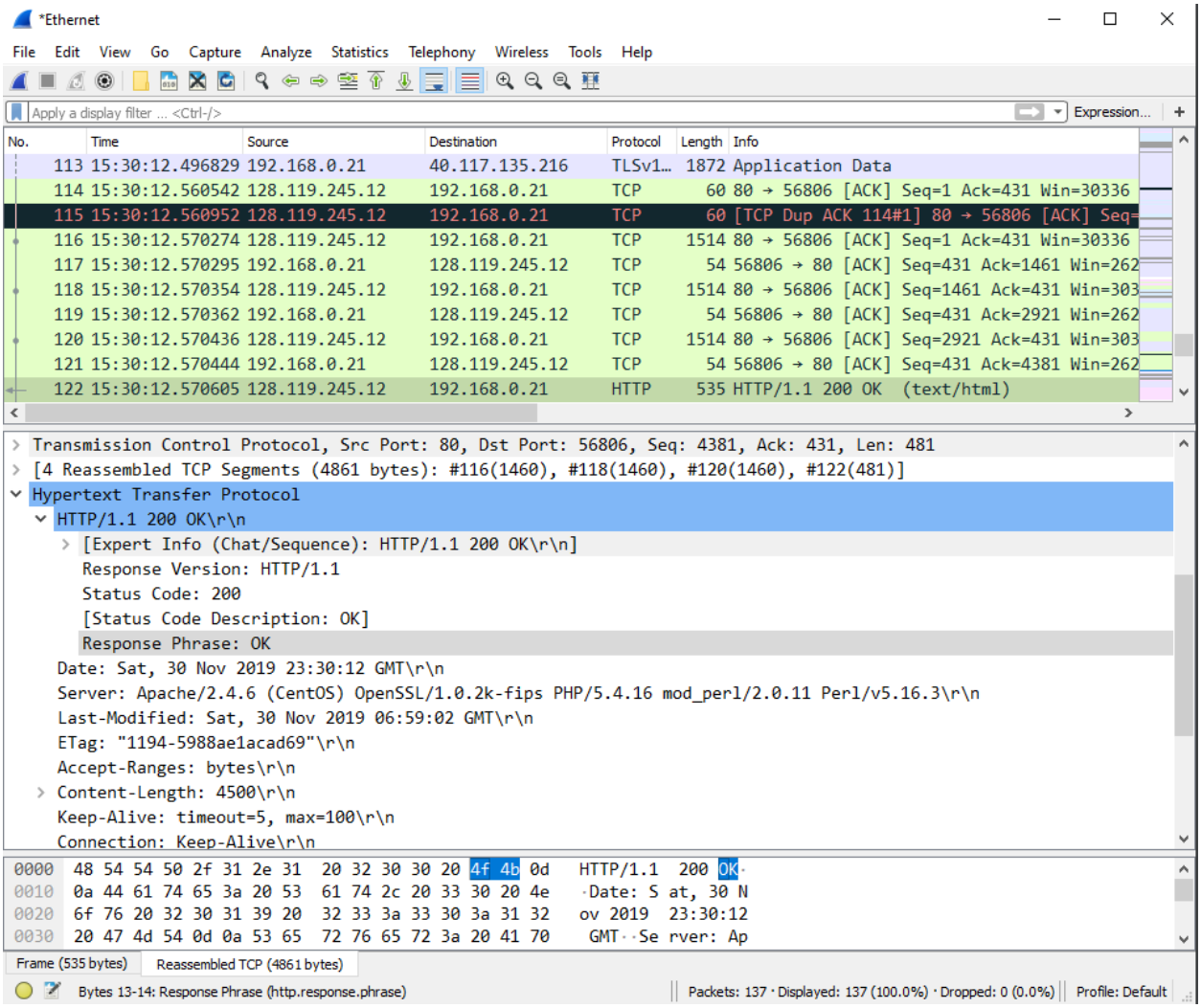
- The destination address is 04:d4:c4:91:eb:e8. This is the address of my computer which manufactured by AsustekC

7. Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?



- The hexadecimal value for the two-byte Frame type field is 0x0800. This value corresponds to the IP protocol like http request packet (Question #3).

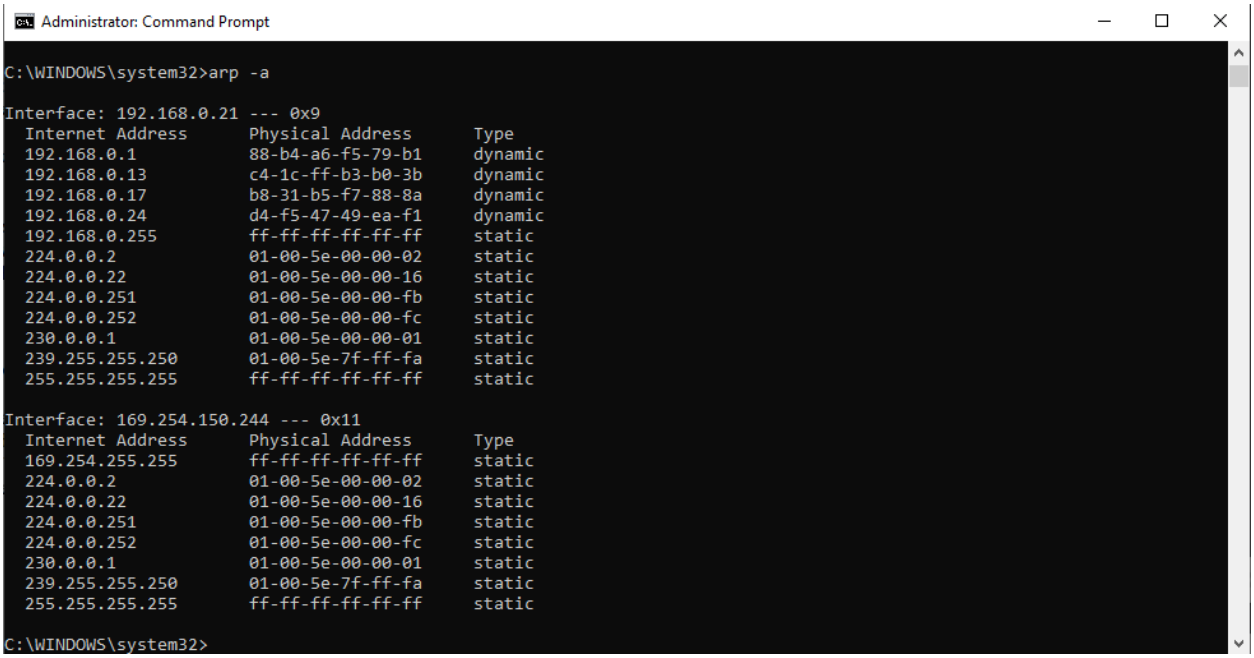
8. How many bytes from the very start of the Ethernet frame does the ASCII “O” in “OK” (i.e., the HTTP response code) appear in the Ethernet frame?



- After 13 bytes, the ASCII “O” in “OK” appears in the Ethernet frame of HTTP respond packet.

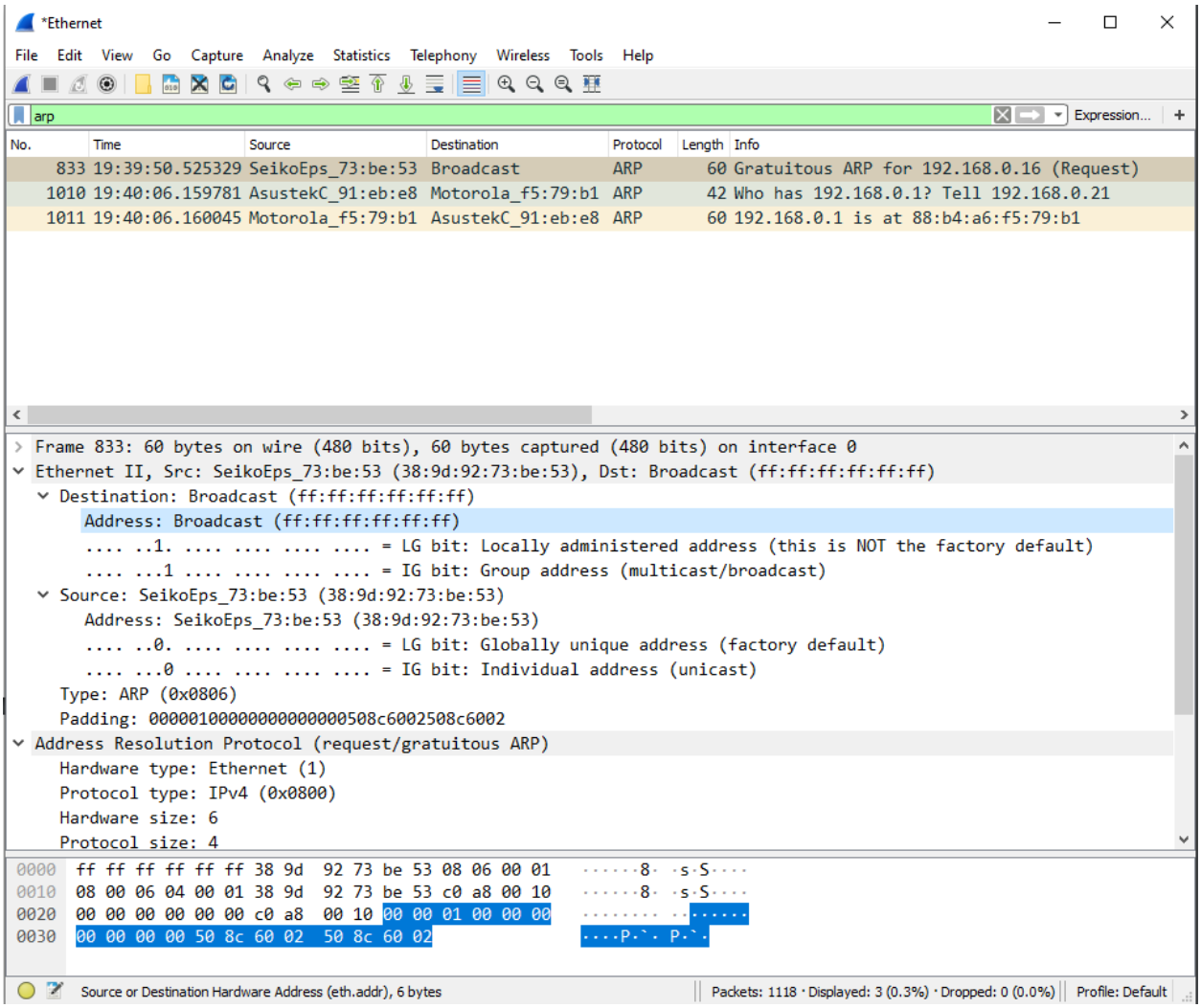
The Windows arp command with no arguments will display the contents of the ARP cache on your computer. Run the arp command.

9. Write down the contents of your computer’s ARP cache. What is the meaning of each column value?



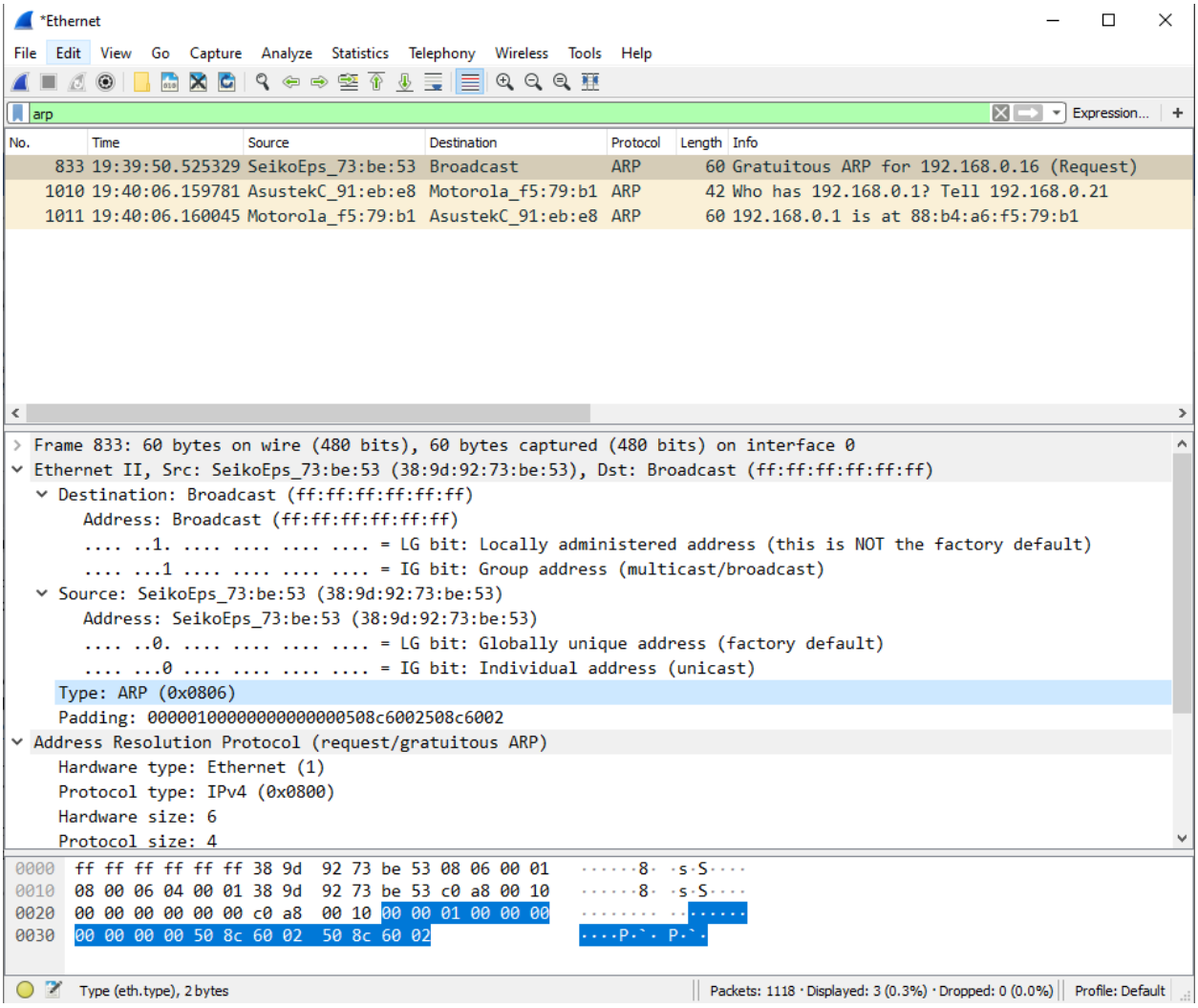
- At first, my desktop computer interface is 192.168.0.21. The Internet Address column means the IP address of each devices. The Physical Address column means the MAC address of each devices. The Type column shows that the protocol type of each devices. For example, my router's IP is 192.168.0.1, MAC address is 88-b4-a6-f5-79-b1, and the protocol type is dynamic.

10. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP request message?



- The source address is 38:9d:92:73:be:53 and the destination address is ff:ff:ff:ff:ff:ff in the Ethernet frame of ARP request message.

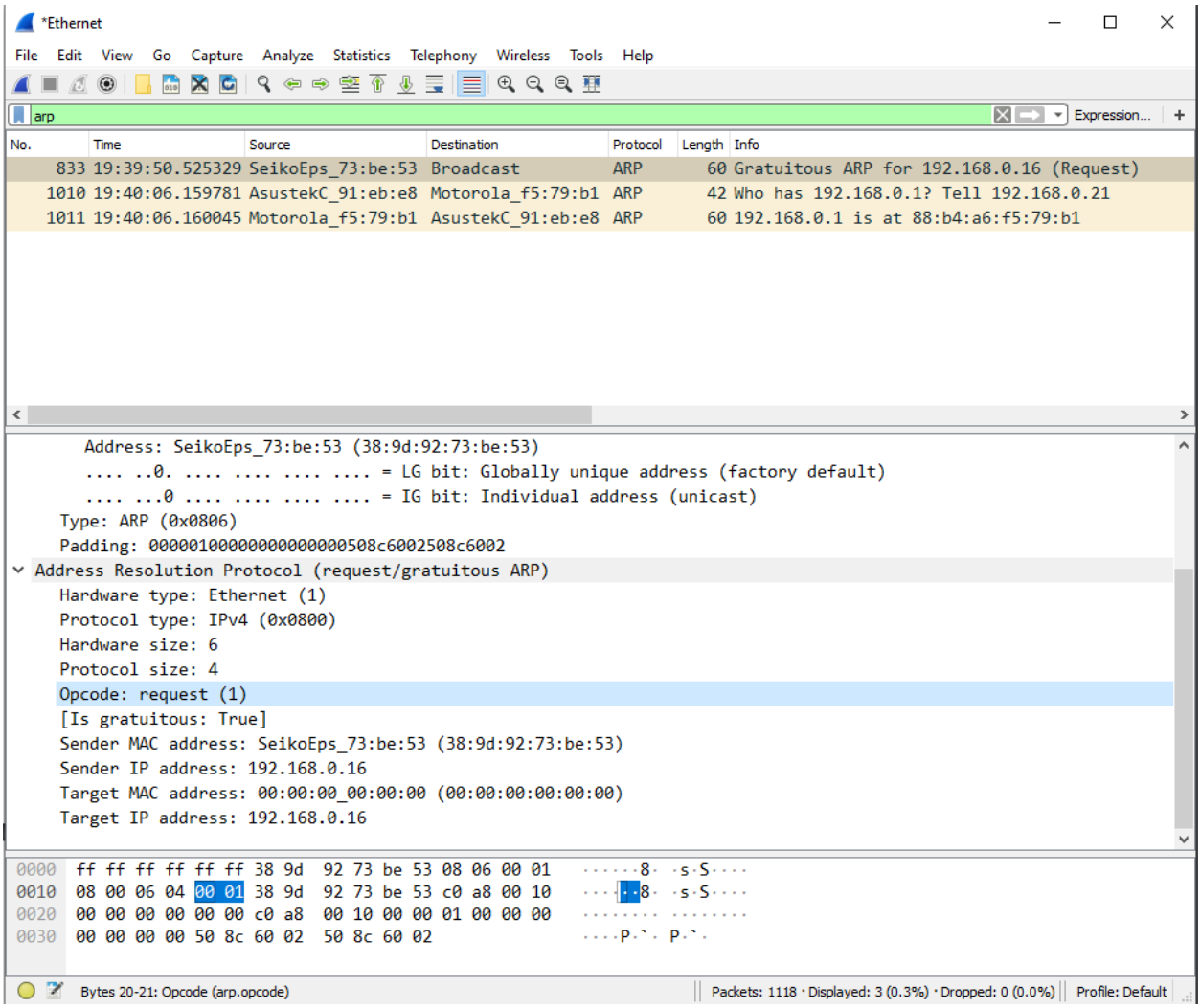
11. Give the hexadecimal value for the two-byte Ethernet Frame type field. What upper layer protocol does this correspond to?



- The hexadecimal value for two-byte Ethernet Frame type field is 0x0806. This value corresponds to the ARP protocol.

12. Download the ARP specification from ftp://ftp.rfc-editor.org/in-notes/std/std37.txt. A readable, detailed discussion of ARP is also at http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html.

a) How many bytes from the very beginning of the Ethernet frame does the ARP opcode field begin?



- The ARP Opcode field begins after 20 bytes from the very beginning of the Ethernet frame.

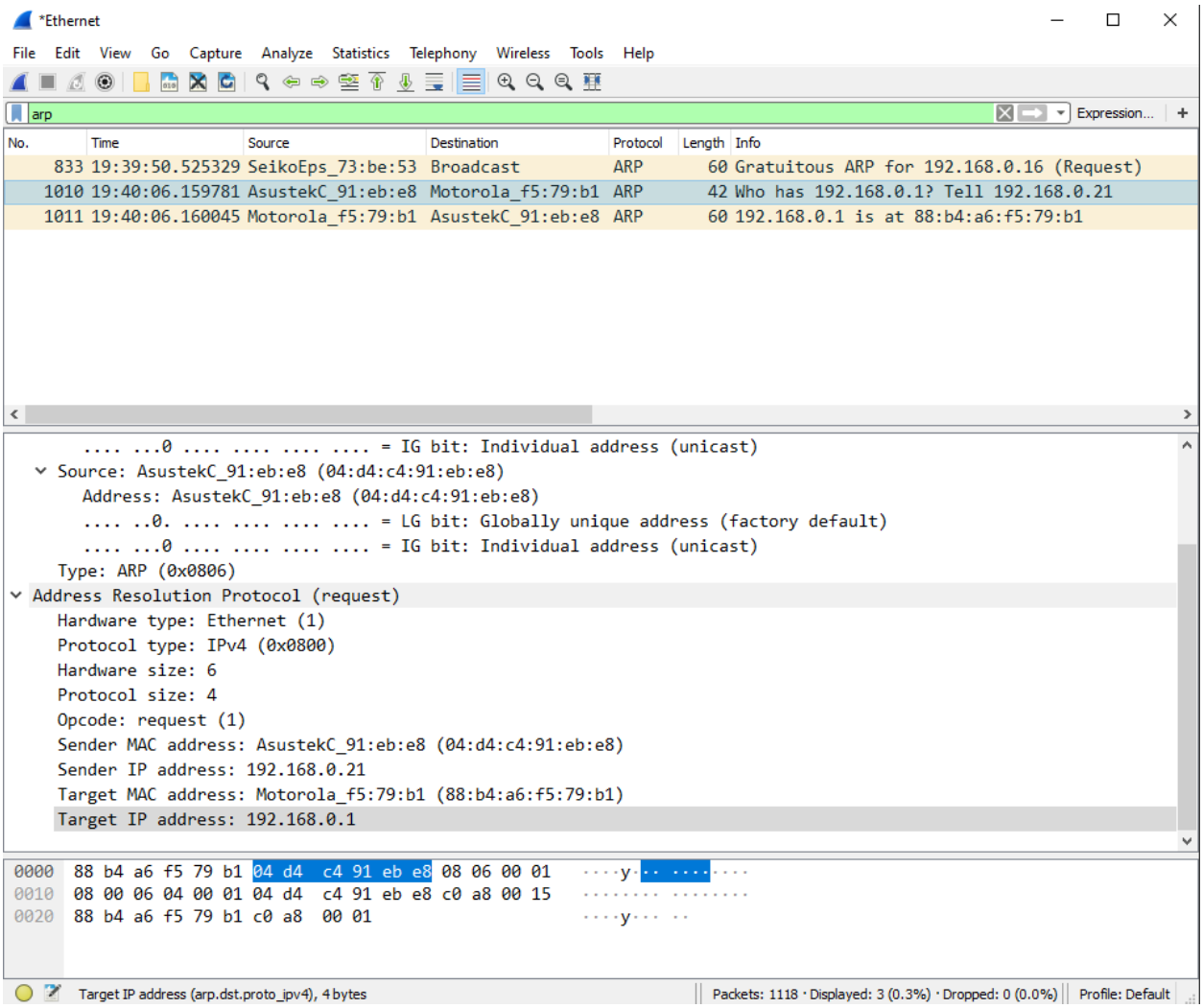
b) What is the value of the opcode field within the ARP-payload part of the Ethernet frame in which an ARP request is made?

- The value of the opcode field within the ARP-payload part of the Ethernet frame is 0x0001 for ARP request.

c) Does the ARP message contain the IP address of the sender?

- Yes, the IP address of the sender is 192.168.0.16

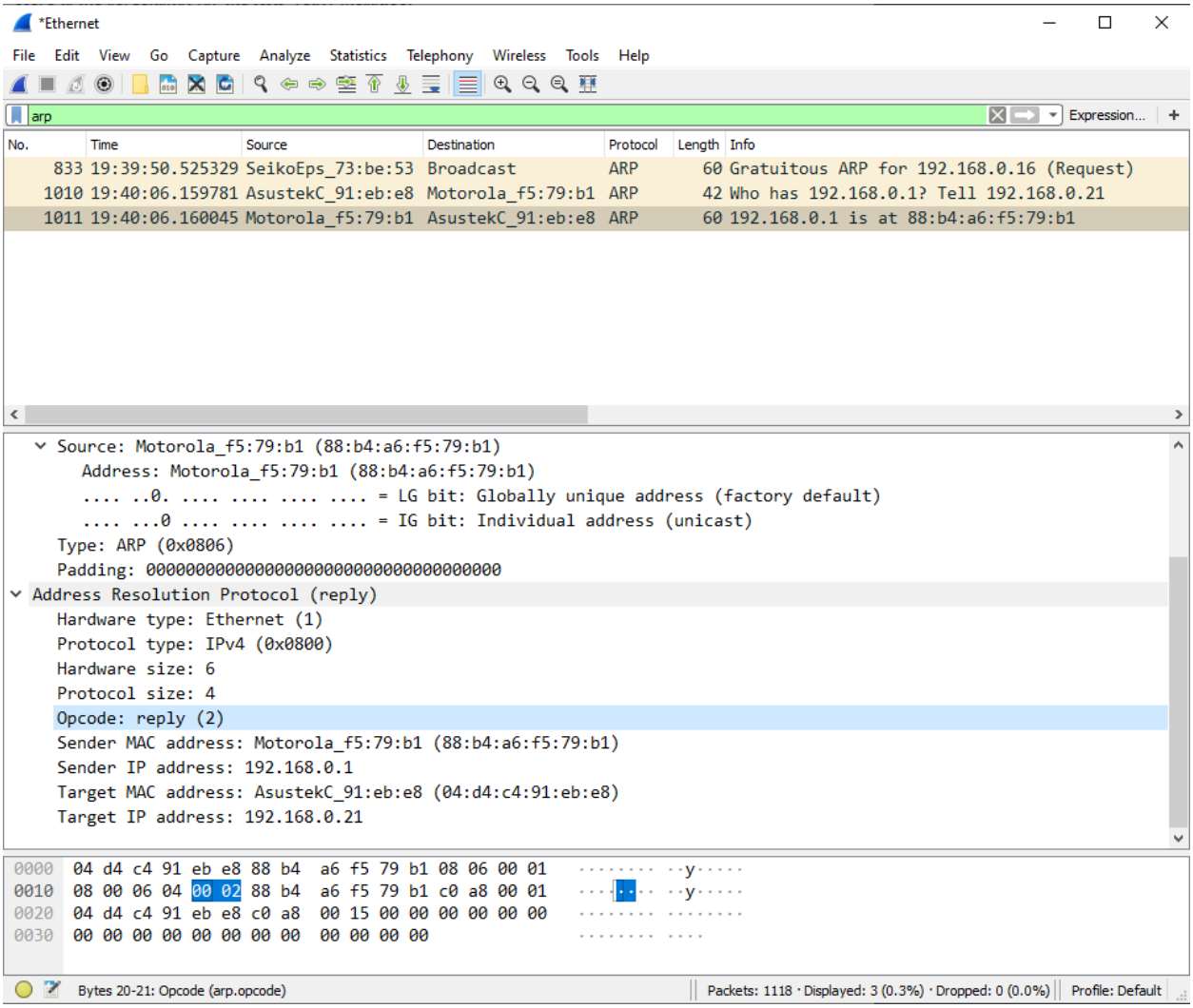
d) Where in the ARP request does the “question” appear – the Ethernet address of the machine whose corresponding IP address is being queried?



- The question “Who has 192.168.0.1?” appears in the target MAC address 88:b4:a6:f5:79:b1 which is the destination as my router.

13. Now find the ARP reply that was sent in response to the ARP request.

a) How many bytes from the very beginning of the Ethernet frame does the ARP opcode field begin?



- The ARP Opcode field begins after 20 bytes from the very beginning of the Ethernet frame.

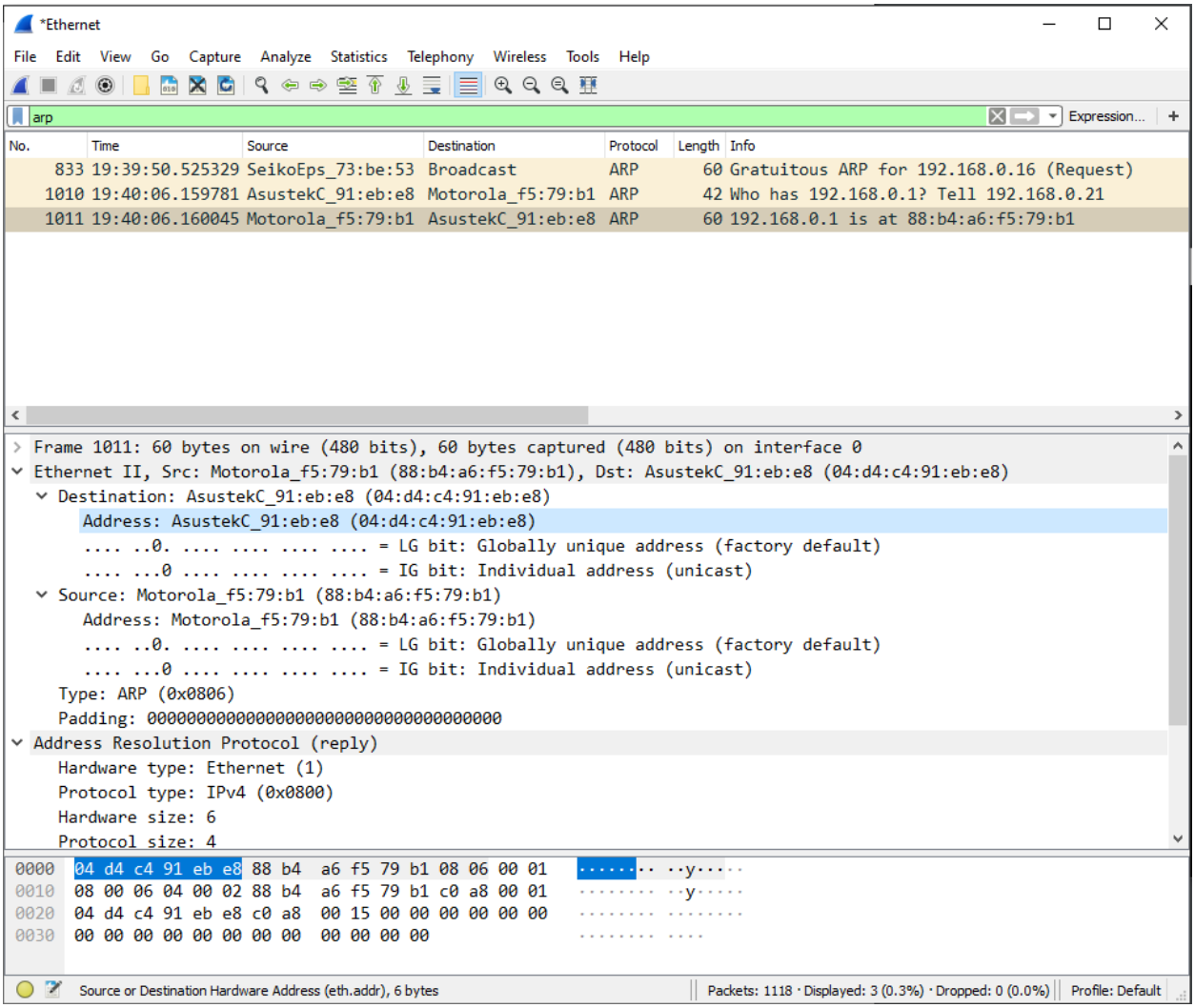
b) What is the value of the opcode field within the ARP-payload part of the Ethernet frame in which an ARP response is made?

- The value of the opcode field within the ARP-payload part of the Ethernet frame is 0x0002 for ARP respond.

c) Where in the ARP message does the “answer” to the earlier ARP request appear – the IP address of the machine having the Ethernet address whose corresponding IP address is being queried?

- The answer “192.168.0.1 is at 88:b4:a6:f5:79:b1” appears in the Sender MAC address: Motorola_f5:79:b1 (88:b4:a6:f5:79:b1) with IP address:192.168.0.1 which is my router.

14. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP reply message?



- The hexadecimal value for source address is 88:b4:a6:f5:79:b1 and for destination address is 04:d4:c4:91:eb:e8

15. Open the ethernet-ethereal-trace-1 trace file in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip. The first and second ARP packets in this trace correspond to an ARP request sent by the computer running Wireshark, and the ARP reply sent to the computer running Wireshark by the computer with the ARP-requested Ethernet address. But there is yet another computer on this network, as indicated by packet 6 – another ARP request. Why is there no ARP reply (sent in response to the ARP request in packet 6) in the packet trace?

- Since we are not at the machine that sent the packet 6's original ARP request (different source between packet 1 and 6), we can't see the reply. We can only see the ARP reply from our machine's ARP request because it sent to our machine directly.

Extra Credit

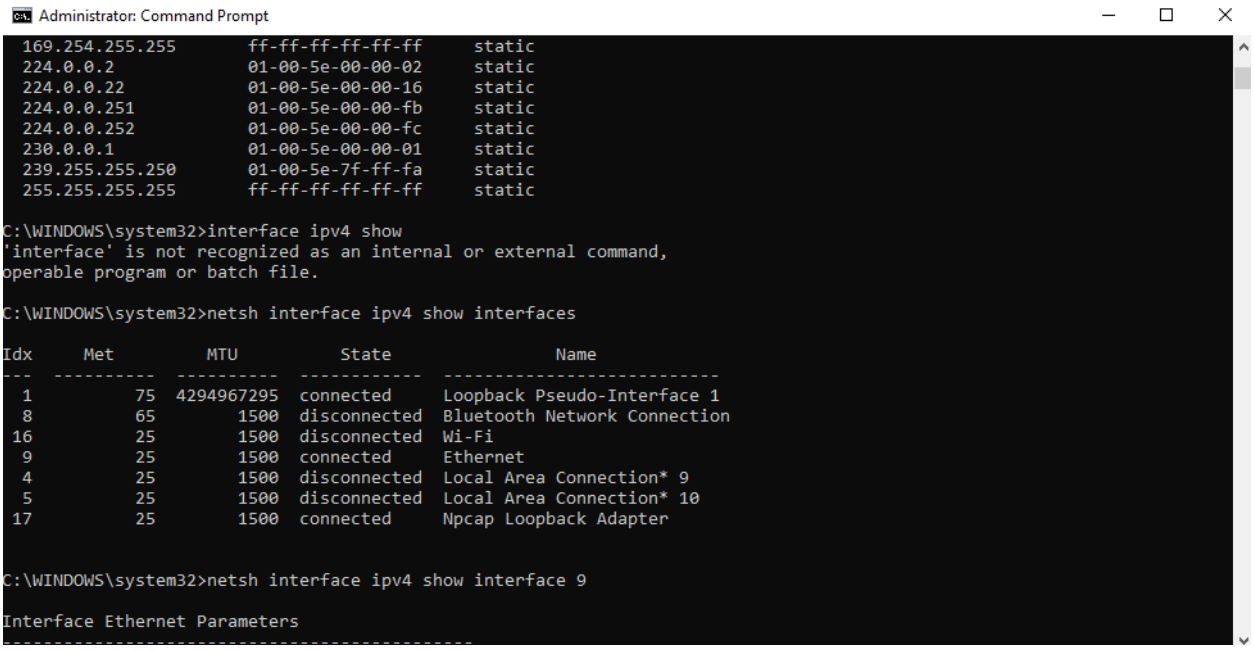
EX-1. The arp command:

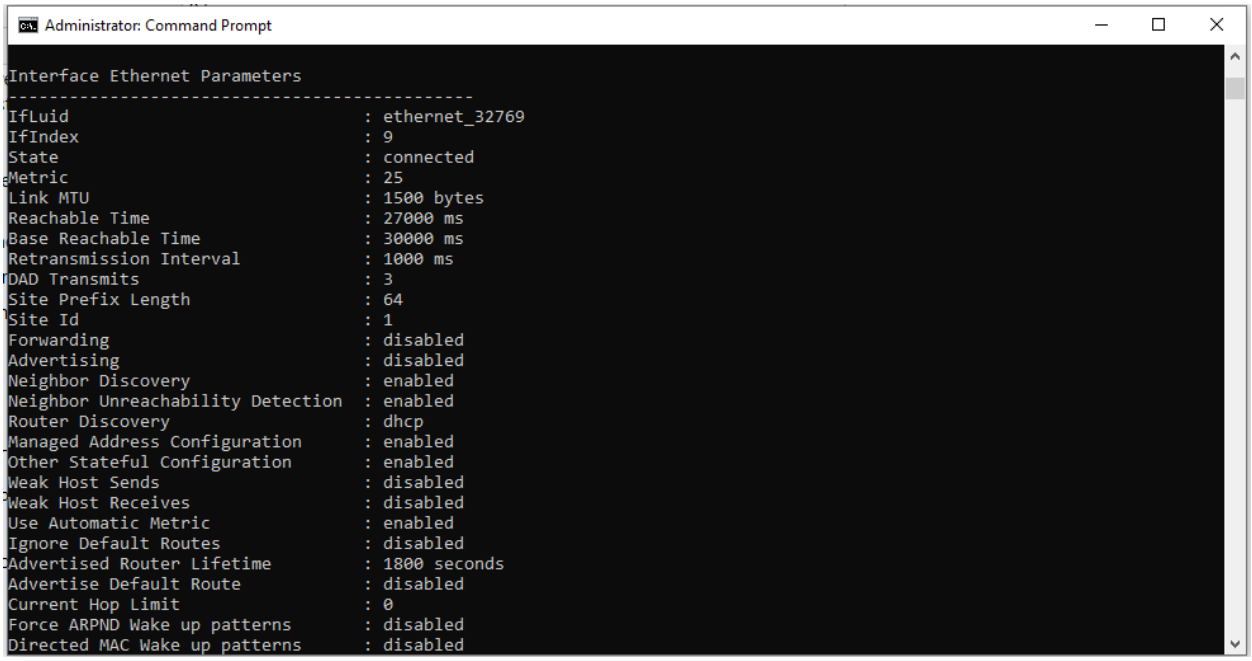
arp -s InetAddr EtherAddr

allows you to manually add an entry to the ARP cache that resolves the IP address InetAddr to the physical address EtherAddr. What would happen if, when you manually added an entry, you entered the correct IP address, but the wrong Ethernet address for that remote interface?

- At first, ARP cache two kind of entries which are static and dynamic. The ARP command will assign the IP address dynamically which is static entry if I manually add an entry to the ARP cache with correct IP address. However, since I entered the wrong Ethernet address, full packet loss will be happened during data transmission.

EX-2. What is the default amount of time that an entry remains in your ARP cache before being removed. You can determine this empirically (by monitoring the cache contents) or by looking this up in your operation system documentation. Indicate how/where you determined this value.





- Since my computer Ethernet interface shows that the default amount of time that an entry remains in my ARP cache before being removed is 27 seconds (27000 ms).

extra credit #2 – Socket programming

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);
```

```
    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);
```

```
        // 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

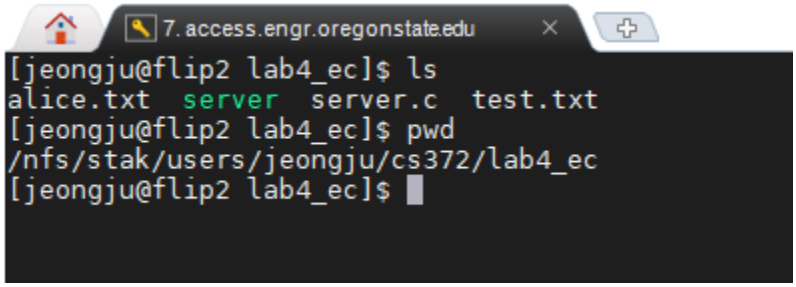
server.c | client.c

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

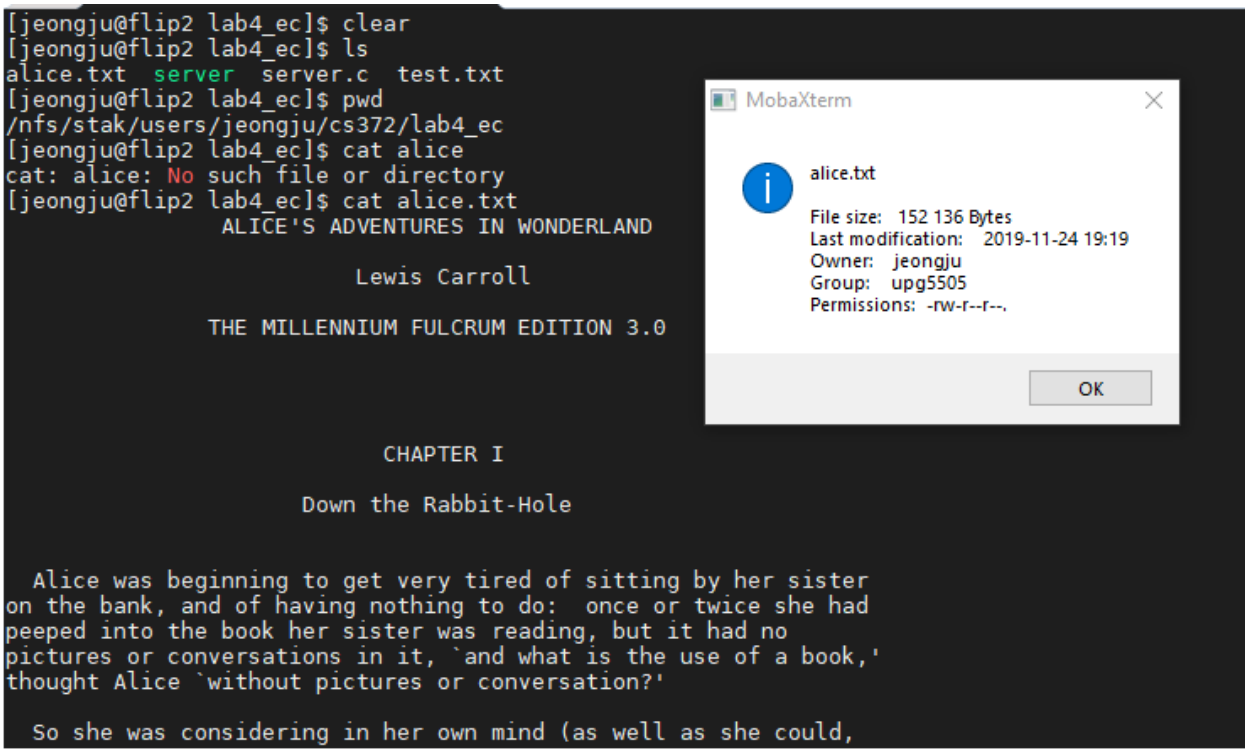
```c
//written by Junhyeok Jeong

// 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");
        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)



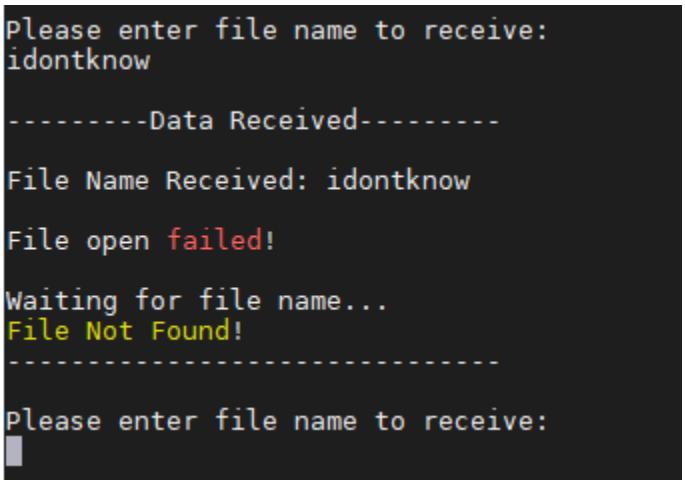
5. Compile server.c on child process for listening

```
[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



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;
```