CSCI4211: Introduction to Computer Networks

Homework Assignment III

Due 11:55PM Nov 18th, 2015

Help-hot-line: csci4211-help@cs.umn.edu

Name: Sihan Chen Student ID: 4452332

Important Notes:

Please submit your solutions as a single archive file (.zip or .tar or .tar.gz) on moodle. You may download the MS Word version of this assignment from moodle and edit it directly. Only online submissions are accepted for this assignment - do not attempt to submit hard copies. All textbook references pertain to the 6th edition.

You may discuss ideas and ask for clarifications freely with others on or off the class forum, and with Professor He or with the TAs. You must not provide or accept other assistance on the assignments. Feel free to post any queries you might have on the moodle discussion forum for this assignment.

Please do not write any thing other than name and student ID on this cover page

For TA only:

Problem	Points	Score
1	10	
2	10	
3	10	
4	10	
5	10	
6	50	
Total	**100**	

1. IP Address (10 pts)

1. Convert the IP address whose hexadecimal representation is CB44AFA2 to dotted decimal notation. (**2 pts**)

Hex: 0xCB44AFA2:
Binary: 1100 1011.0100 0100.1010 1111.1010 0010
Decimal: 203.68.175.162

2. What is the 32-bit binary equivalent of the IP address 100.114.20.107? (**2pts**)

Decimal: 100.114.20.107
Binary: 01100100. 01110010. 00010100. 01101011

3. A network on the Internet has a subnet mask of 255.255.192.0. What is the maximum number of hosts it can handle (note: network address and local broadcast address are not assigned to individual hosts)? (**2 pts**)

225.225.192.0 = 11111111.11111111.11000000.00000000
Based on the conversion above, we know we have 14 bits for the hosts. So the maximum number of hosts it can handle is $2^{14} - 2 = 16382$

4. Suppose an organization owns the block of addresses of the form 129.17.129.96/28. Suppose it wants to create four IP subnets from this block, with each block having the same number of IP addresses. What are the prefixes (of form xxx.xxx.xxx/y) for the four IP subnets? (**4pts**)

129.17.129.96 = 10000001.00010001.10000001.01100000
129.17.129.96/28 = 10000001.00010001.10000001.0110****
Since we want to create 4 IP subnets, that is 4 = 2^2, 2 bits.
So prefixes for the four IP subnets will be:
129.17.129.96/30 = 10000001.00010001.10000001.011000**
129.17.129.96/30 = 10000001.00010001.10000001.011001**
129.17.129.96/30 = 10000001.00010001.10000001.011010**
129.17.129.96/30 = 10000001.00010001.10000001.011011**

2. IP Datagram Forwarding (10 pts)

Considering a datagram network using 8-bit host addressing (totally 254 hosts), suppose a router use longest prefix matching and has following forwarding table. How many host addresses will the router route through interface 0, interface 1 respectively (5pt each)? (hint: when host id is all 0, the address is network address and when host id is all 1's, the address is local broadcast address. Both are not host addresses)

Prefix (binary)	*Interface*
1	*0*
101	*1*
101110	*2*
Otherwise	*3*

Since the prefix of interface 3 is longer than interface 2 and longer than interface 1. And based on the longest prefix matching, if a prefix is 101110, then we will match it with interface 2 instead of interface 1 and 0. So we start our calculation on interface 2.

For interface 2, 101110** we have 2 bits left so the number of hosts for interface 2 is

$$2^2 - 2 = 2$$

For interface 1, 101***** we have 5 bits left so the number of hosts for interface 1 is

$$2^5 - 2^2 - 2 = 26$$

Where 2^2 is just the number of hosts, which prefix is 101110.
And for the interface 0, 1******* we have 7 bits left so the number of hosts for interface 0 is

$$2^7 - (2^5 - 2^2) - 2^2 - 2 = 94$$

Where $2^5 - 2^2$ is just 32 hosts which prefix is 101, and 2^2 is 4 hosts which prefix is

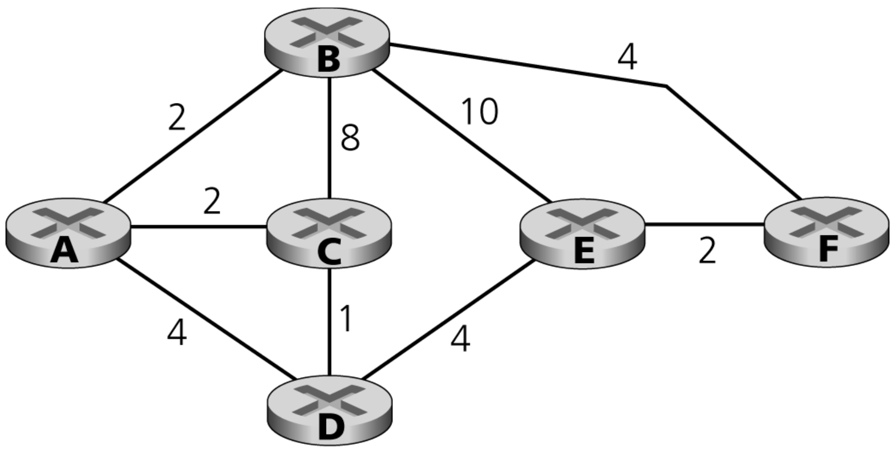
101110. And all the minus 2 in the calculation above is due to all 0s and all 1s situation.

3. Link State Routing (10 pts)

Consider the network shown below. Show the operation of Dijkstra's (link-state) algorithm for computing the least cost path from A to all destinations using the table below (**8 pts**).
What is the shortest path from A to F, and what is the cost of this path (**2 pts**)?

Step	**N**	**D(B),p(B)**	**D(C),p(C)**	**D(D),p(D)**	**D(E),p(E)**	**D(F),p(F)**
0	A	2,A	2,A	4,A	∞	∞
1	AB	2,A	2,A	4,A	12,B	6,B
2	ABC	2,A	2,A	3,C	12,B	6,B
3	ABCD	2,A	2,A	3,C	7,D	6,B
4	ABCDE	2,A	2,A	3,C	7,D	6,B
5	ABCDE F	2,A	2,A	3,C	7,D	6,B

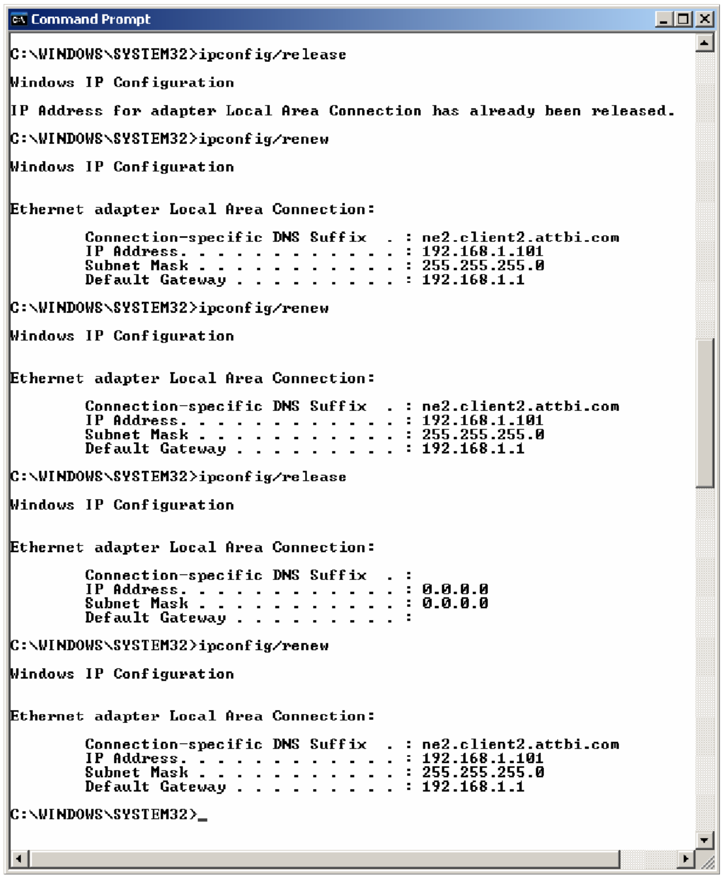
As shown in the calculated result above, the minimum cost from A to F is 6, and its predecessor is node B, so the path is A-B-F.



4. Hand-on Practice: DHCP (10 pts)

In this practice, we'll take a quick look at DHCP. In order to observe DHCP in action, we'll perform several DHCP-related commands and capture the DHCP messages exchanged as a result of executing these commands. Do the following as in Figure:

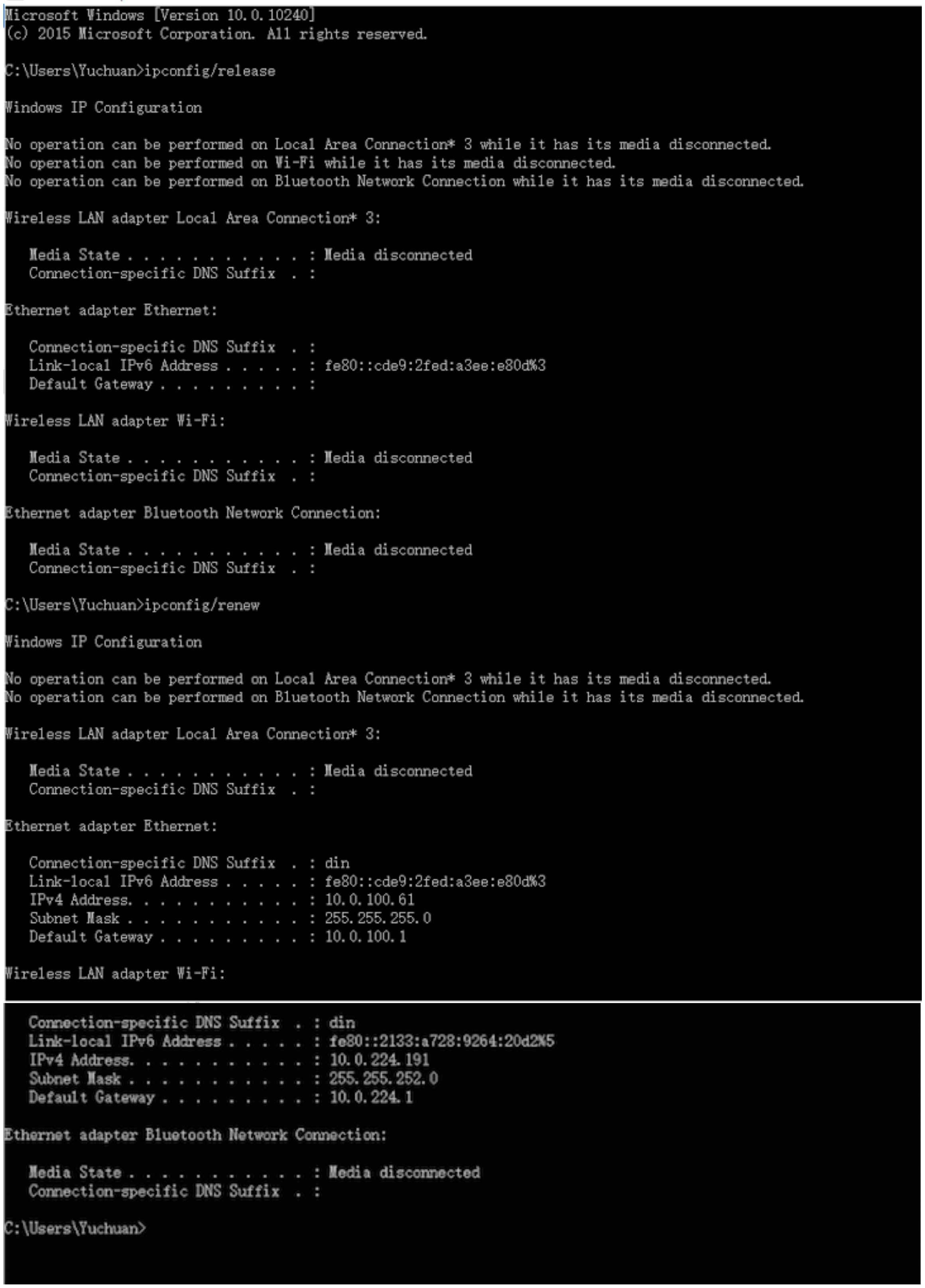
1. Enter "*ipconfig /release*" to releases your current IP address, so that your host's IP address becomes 0.0.0.0. (sudo dhclient –r is used in linux)
2. Start up the Wireshark packet sniffer, with *"bootp"* as the filter (Note to see DHCP packets in the current version of Wireshark, you need to enter *"bootp"* and not "dhcp" in the filter.)
3. Enter "*ipconfig /renew*". This instructs your host to obtain a network configuration, including a new IP address.
1. Stop Wireshark packet capture.

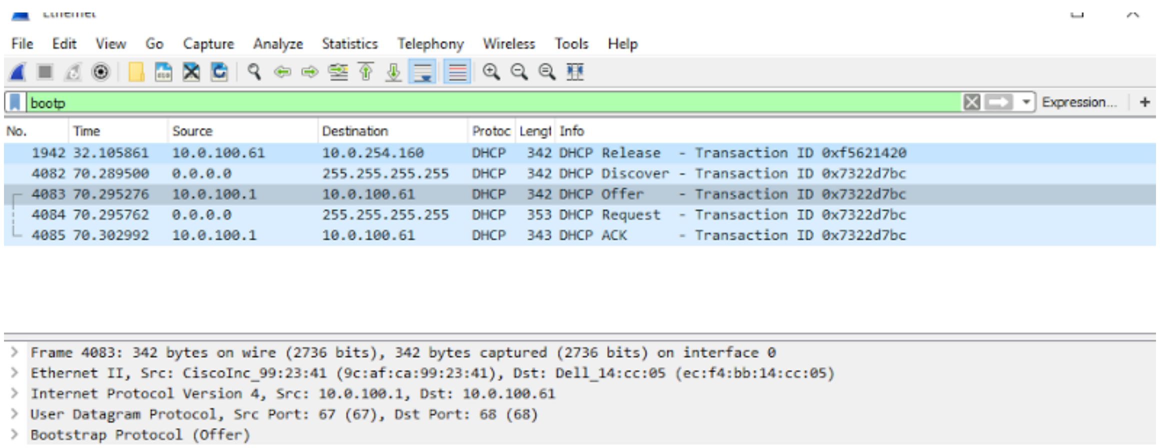


```
C:\WINDOWS\SYSTEM32>ipconfig/release

Windows IP Configuration

IP Address for adapter Local Area Connection has already been released.

C:\WINDOWS\SYSTEM32>ipconfig/renew

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : ne2.client2.attbi.com
        IP Address. . . . . . . . . . . . : 192.168.1.101
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

C:\WINDOWS\SYSTEM32>ipconfig/renew

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : ne2.client2.attbi.com
        IP Address. . . . . . . . . . . . : 192.168.1.101
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

C:\WINDOWS\SYSTEM32>ipconfig/release

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 0.0.0.0
        Subnet Mask . . . . . . . . . . . : 0.0.0.0
        Default Gateway . . . . . . . . . :

C:\WINDOWS\SYSTEM32>ipconfig/renew

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : ne2.client2.attbi.com
        IP Address. . . . . . . . . . . . : 192.168.1.101
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1

C:\WINDOWS\SYSTEM32>_
```

Provide two screen shots: command line screen similar to the figure above and a wireshark screen that captures the DHCP interaction (you can have similar screenshots in unix)

Based on the screenshots, answer the following questions:


```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Yuchuan>ipconfig/release

Windows IP Configuration

No operation can be performed on Local Area Connection* 3 while it has its media disconnected.
No operation can be performed on Wi-Fi while it has its media disconnected.
No operation can be performed on Bluetooth Network Connection while it has its media disconnected.

Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::cde9:2fed:a3ee:e80d%3
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wi-Fi:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\Yuchuan>ipconfig/renew

Windows IP Configuration

No operation can be performed on Local Area Connection* 3 while it has its media disconnected.
No operation can be performed on Bluetooth Network Connection while it has its media disconnected.

Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : din
   Link-local IPv6 Address . . . . . : fe80::cde9:2fed:a3ee:e80d%3
   IPv4 Address. . . . . . . . . . . : 10.0.100.61
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.100.1

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : din
   Link-local IPv6 Address . . . . . : fe80::2133:a728:9264:20d2%5
   IPv4 Address. . . . . . . . . . . : 10.0.224.191
   Subnet Mask . . . . . . . . . . . : 255.255.252.0
   Default Gateway . . . . . . . . . : 10.0.224.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\Yuchuan>
```


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
bootp
Expression...

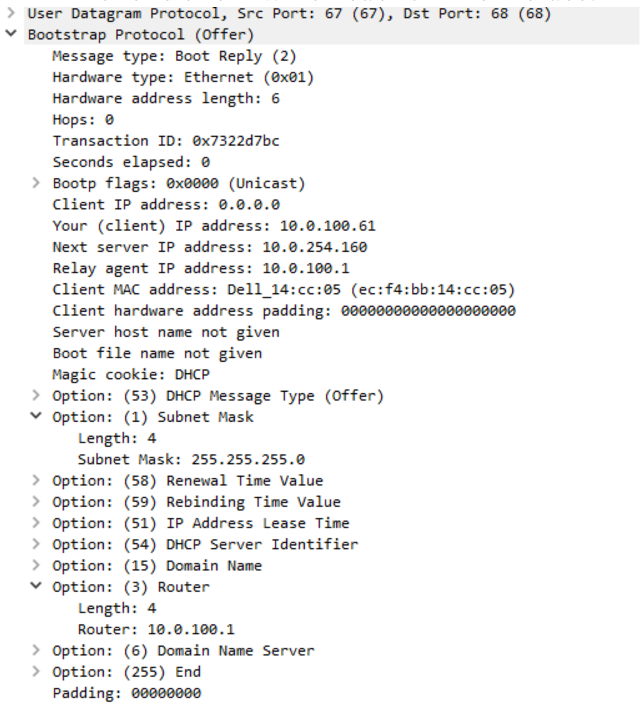
```
No.   Time       Source         Destination      Protoc Lengt Info
 1942 32.105861  10.0.100.61    10.0.254.160     DHCP   342 DHCP Release  - Transaction ID 0xf5621420
 4082 70.289500  0.0.0.0        255.255.255.255  DHCP   342 DHCP Discover - Transaction ID 0x7322d7bc
 4083 70.295276  10.0.100.1     10.0.100.61      DHCP   342 DHCP Offer    - Transaction ID 0x7322d7bc
 4084 70.295762  0.0.0.0        255.255.255.255  DHCP   353 DHCP Request  - Transaction ID 0x7322d7bc
 4085 70.302992  10.0.100.1     10.0.100.61      DHCP   343 DHCP ACK      - Transaction ID 0x7322d7bc
```

```
> Frame 4083: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
> Ethernet II, Src: CiscoInc_99:23:41 (9c:af:ca:99:23:41), Dst: Dell_14:cc:05 (ec:f4:bb:14:cc:05)
> Internet Protocol Version 4, Src: 10.0.100.1, Dst: 10.0.100.61
> User Datagram Protocol, Src Port: 67 (67), Dst Port: 68 (68)
> Bootstrap Protocol (Offer)
```

1. Are DHCP messages sent over UDP or TCP? (**2pts**)
   Based on the graph above, DHCP messages were sent over UDP.

2. Explain the purpose of the router and subnet mask lines in the DHCP offer message. (**2pts**)

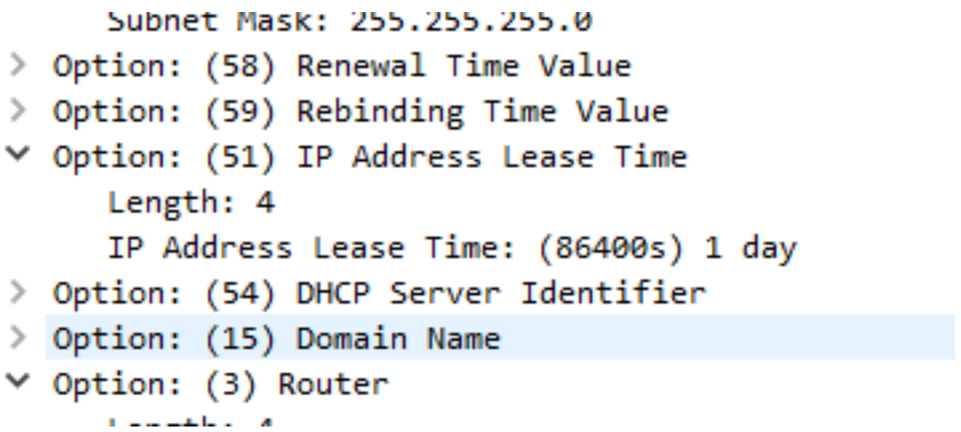
Router line tells client where to send the message by default, while subnet mask line tells client which subnet mask to use.



```
> User Datagram Protocol, Src Port: 67 (67), Dst Port: 68 (68)
v Bootstrap Protocol (Offer)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x7322d7bc
    Seconds elapsed: 0
  > Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 10.0.100.61
    Next server IP address: 10.0.254.160
    Relay agent IP address: 10.0.100.1
    Client MAC address: Dell_14:cc:05 (ec:f4:bb:14:cc:05)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  > Option: (53) DHCP Message Type (Offer)
  v Option: (1) Subnet Mask
      Length: 4
      Subnet Mask: 255.255.255.0
  > Option: (58) Renewal Time Value
  > Option: (59) Rebinding Time Value
  > Option: (51) IP Address Lease Time
  > Option: (54) DHCP Server Identifier
  > Option: (15) Domain Name
  v Option: (3) Router
      Length: 4
      Router: 10.0.100.1
  > Option: (6) Domain Name Server
  > Option: (255) End
    Padding: 00000000
```

3. Explain the purpose of the lease time. How long is the lease time in your experiment? (**2pts**)

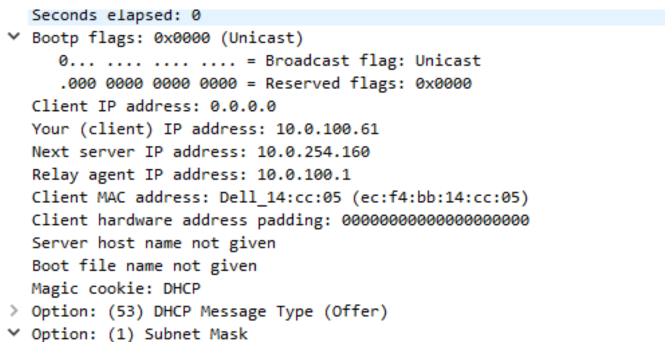
The lease time is the maximum time a client can use the IP address assigned by the server. In my experiment, it was 86400s, 1 day.



```
    Subnet Mask: 255.255.255.0
> Option: (58) Renewal Time Value
> Option: (59) Rebinding Time Value
v Option: (51) IP Address Lease Time
    Length: 4
    IP Address Lease Time: (86400s) 1 day
> Option: (54) DHCP Server Identifier
> Option: (15) Domain Name
v Option: (3) Router
```

4. Explain the purpose of the DHCP release message? What would happen if the client's DHCP release message is lost? (**2pts**)
   The purpose of the DHCP release message is to release IP address back to the server. It tells the DHCP server that the client does not need the IP address anymore and that IP address can be assigned to the other clients. And if the client's DHCP release message is lost, the client releases the IP address, but IP address will still be assigned to that client, until its IP address lease time is over.

5. In a certain network configuration, the DHCP server might not be located at the same network as your machine. In this case, DHCP request are relayed by a relay agent. Is there a relay agent in your experiment? Justify your answer. (**2pts**)
   Yes, there is a relay agent in my experiment, since Relay agent IP address in the picture below is not 0.0.0.0. And the IP address for the relay agent is 10.0.100.1.



```
  Seconds elapsed: 0
v Bootp flags: 0x0000 (Unicast)
    0... .... .... .... = Broadcast flag: Unicast
    .000 0000 0000 0000 = Reserved flags: 0x0000
  Client IP address: 0.0.0.0
  Your (client) IP address: 10.0.100.61
  Next server IP address: 10.0.254.160
  Relay agent IP address: 10.0.100.1
  Client MAC address: Dell_14:cc:05 (ec:f4:bb:14:cc:05)
  Client hardware address padding: 00000000000000000000
  Server host name not given
  Boot file name not given
  Magic cookie: DHCP
> Option: (53) DHCP Message Type (Offer)
v Option: (1) Subnet Mask
```

5. Hand-on Practice: ICMP (10 pts.)

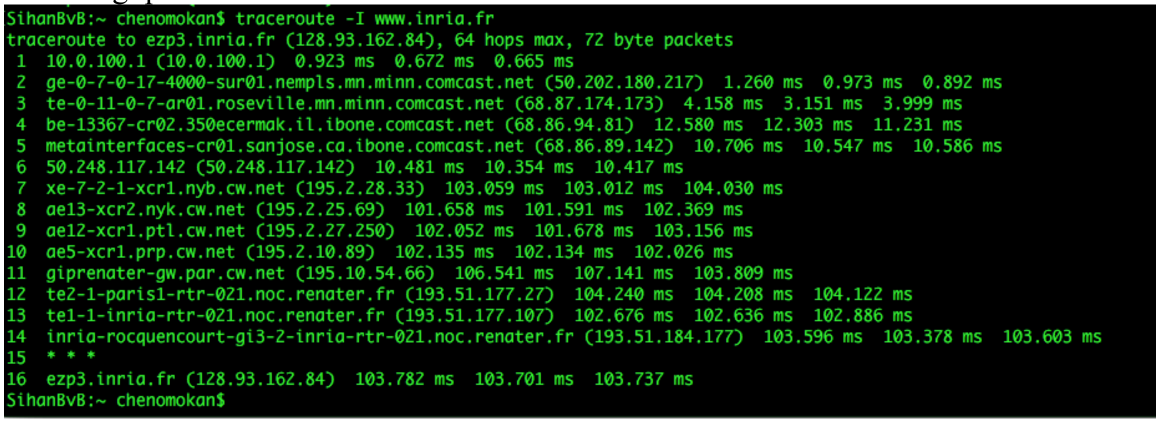
In this practice, we capture the packets generated by the Traceroute program. You may recall that the Traceroute program can be used to figure out the path a packet takes from source to destination. Traceroute is discussed in Section 1.3 and in Section 4.4 of the text.

Traceroute is implemented in different ways in Unix/Linux and in Windows. In Unix/Linux, the source sends a series of UDP packets to the target destination using an unlikely destination port number; in Windows, the source sends a series of ICMP packets to the target destination. For both operating systems, the program sends the first packet with TTL=1, the second packet with TTL=2, and so on. Recall that a router will decrement a packet's TTL value as the packet passes through the router. When a packet arrives at a router with TTL=1, the router sends an ICMP error packet back to the source.

Do the following:

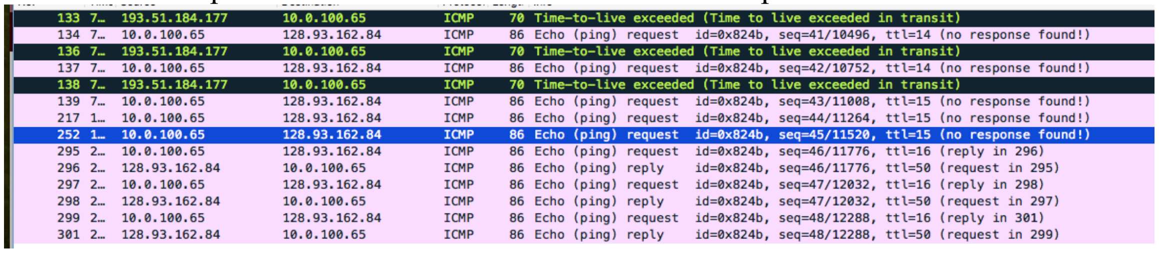
1. Start up the Wireshark packet sniffer, and begin Wireshark packet capture.
2. Type "tracert hostname". Choose a host outside of north America such as www.inria.fr , a computer science research institute in France.
3. When the Traceroute program terminates, stop packet capture in Wireshark.

You should hand in a screen shot of the output of tracert command, then answering the following question:



```
SihanBvB:~ chenomokan$ traceroute -I www.inria.fr
traceroute to ezp3.inria.fr (128.93.162.84), 64 hops max, 72 byte packets
 1  10.0.100.1 (10.0.100.1)  0.923 ms  0.672 ms  0.665 ms
 2  ge-0-7-0-17-4000-sur01.nempls.mn.minn.comcast.net (50.202.180.217)  1.260 ms  0.973 ms  0.892 ms
 3  te-0-11-0-7-ar01.roseville.mn.minn.comcast.net (68.87.174.173)  4.158 ms  3.151 ms  3.999 ms
 4  be-13367-cr02.350ecermak.il.ibone.comcast.net (68.86.94.81)  12.580 ms  12.303 ms  11.231 ms
 5  metainterfaces-cr01.sanjose.ca.ibone.comcast.net (68.86.89.142)  10.706 ms  10.547 ms  10.586 ms
 6  50.248.117.142 (50.248.117.142)  10.481 ms  10.354 ms  10.417 ms
 7  xe-7-2-1-xcr1.nyb.cw.net (195.2.28.33)  103.059 ms  103.012 ms  104.030 ms
 8  ae13-xcr2.nyk.cw.net (195.2.25.69)  101.658 ms  101.591 ms  102.369 ms
 9  ae12-xcr1.ptl.cw.net (195.2.27.250)  102.052 ms  101.678 ms  103.156 ms
10  ae5-xcr1.prp.cw.net (195.2.10.89)  102.135 ms  102.134 ms  102.026 ms
11  giprenater-gw.par.cw.net (195.10.54.66)  106.541 ms  107.141 ms  103.809 ms
12  te2-1-paris1-rtr-021.noc.renater.fr (193.51.177.27)  104.240 ms  104.208 ms  104.122 ms
13  te1-1-inria-rtr-021.noc.renater.fr (193.51.177.107)  102.676 ms  102.636 ms  102.886 ms
14  inria-rocquencourt-gi3-2-inria-rtr-021.noc.renater.fr (193.51.184.177)  103.596 ms  103.378 ms  103.603 ms
15  * * *
16  ezp3.inria.fr (128.93.162.84)  103.782 ms  103.701 ms  103.737 ms
SihanBvB:~ chenomokan$
```

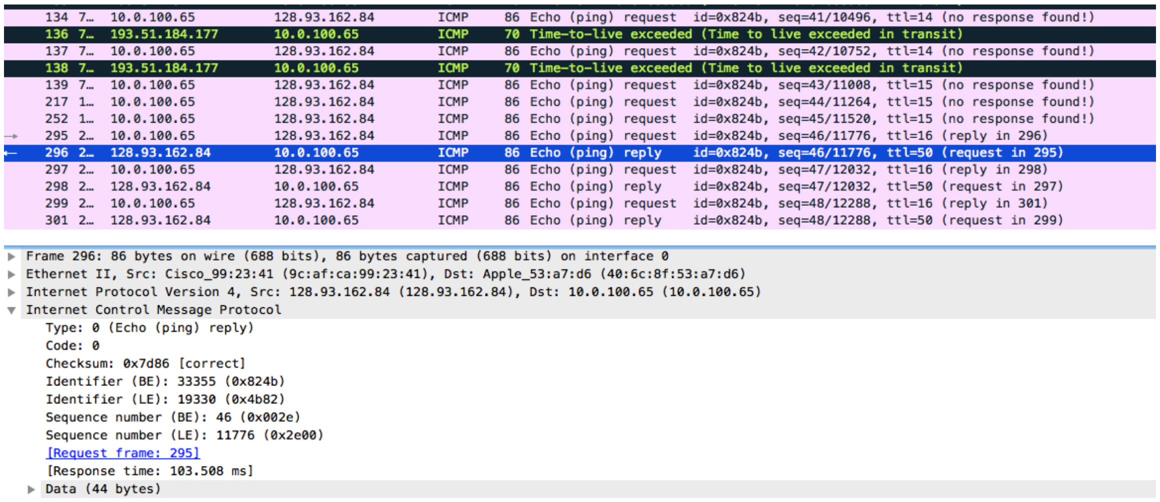
1. How many ICMP echo packets are sent? Justify the observed number from wireshark, based on the output of tracert command. (**4pts**)
   16*3 = 48 echo packets are sent. Each TTL has three attempts.



```
133 7…  193.51.184.177   10.0.100.65      ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
134 7…  10.0.100.65      128.93.162.84    ICMP  86 Echo (ping) request  id=0x824b, seq=41/10496, ttl=14 (no response found!)
136 7…  193.51.184.177   10.0.100.65      ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
137 7…  10.0.100.65      128.93.162.84    ICMP  86 Echo (ping) request  id=0x824b, seq=42/10752, ttl=14 (no response found!)
138 7…  193.51.184.177   10.0.100.65      ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
139 7…  10.0.100.65      128.93.162.84    ICMP  86 Echo (ping) request  id=0x824b, seq=43/11008, ttl=15 (no response found!)
217 1…  10.0.100.65      128.93.162.84    ICMP  86 Echo (ping) request  id=0x824b, seq=44/11264, ttl=15 (no response found!)
252 1…  10.0.100.65      128.93.162.84    ICMP  86 Echo (ping) request  id=0x824b, seq=45/11520, ttl=15 (no response found!)
295 2…  10.0.100.65      128.93.162.84    ICMP  86 Echo (ping) request  id=0x824b, seq=46/11776, ttl=16 (reply in 296)
296 2…  128.93.162.84    10.0.100.65      ICMP  86 Echo (ping) reply    id=0x824b, seq=46/11776, ttl=50 (request in 295)
297 2…  10.0.100.65      128.93.162.84    ICMP  86 Echo (ping) request  id=0x824b, seq=47/12032, ttl=16 (reply in 298)
298 2…  128.93.162.84    10.0.100.65      ICMP  86 Echo (ping) reply    id=0x824b, seq=47/12032, ttl=50 (request in 297)
299 2…  10.0.100.65      128.93.162.84    ICMP  86 Echo (ping) request  id=0x824b, seq=48/12288, ttl=16 (reply in 301)
301 2…  128.93.162.84    10.0.100.65      ICMP  86 Echo (ping) reply    id=0x824b, seq=48/12288, ttl=50 (request in 299)
```

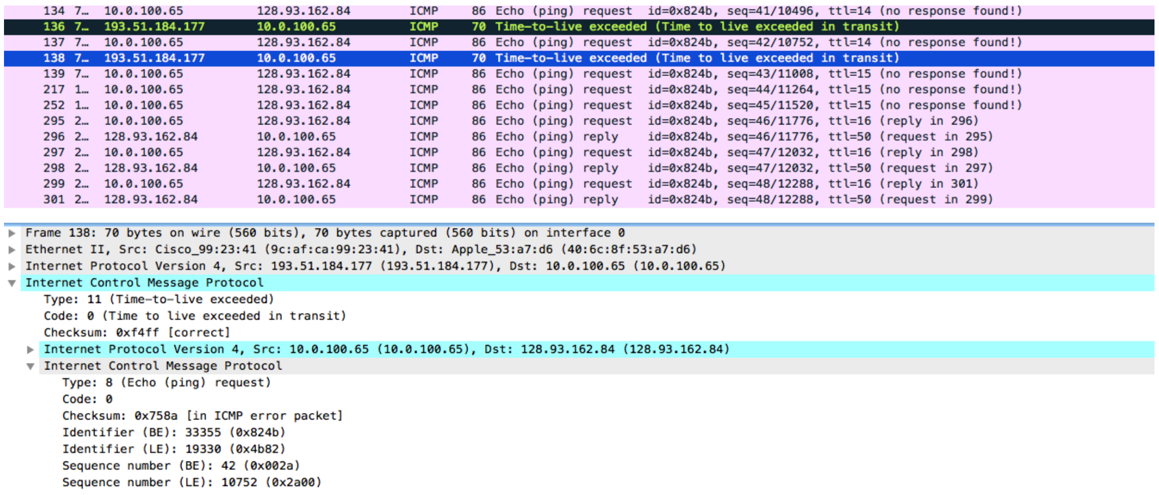
2. Examine the last three ICMP error packets received by the source host. How are these packets different from the previous ICMP error packets? Why are they different? (**6pts**)

The screenshot below is the screenshot of one of the last three ICMP echo packets received by the source host.



```
134 7…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=41/10496, ttl=14 (no response found!)
136 7…  193.51.184.177   10.0.100.65      ICMP   70 Time-to-live exceeded (Time to live exceeded in transit)
137 7…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=42/10752, ttl=14 (no response found!)
138 7…  193.51.184.177   10.0.100.65      ICMP   70 Time-to-live exceeded (Time to live exceeded in transit)
139 7…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=43/11008, ttl=15 (no response found!)
217 1…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=44/11264, ttl=15 (no response found!)
252 1…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=45/11520, ttl=15 (no response found!)
295 2…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=46/11776, ttl=16 (reply in 296)
296 2…  128.93.162.84    10.0.100.65      ICMP   86 Echo (ping) reply    id=0x824b, seq=46/11776, ttl=50 (request in 295)
297 2…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=47/12032, ttl=16 (reply in 298)
298 2…  128.93.162.84    10.0.100.65      ICMP   86 Echo (ping) reply    id=0x824b, seq=47/12032, ttl=50 (request in 297)
299 2…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=48/12288, ttl=16 (reply in 301)
301 2…  128.93.162.84    10.0.100.65      ICMP   86 Echo (ping) reply    id=0x824b, seq=48/12288, ttl=50 (request in 299)

▶ Frame 296: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Cisco_99:23:41 (9c:af:ca:99:23:41), Dst: Apple_53:a7:d6 (40:6c:8f:53:a7:d6)
▶ Internet Protocol Version 4, Src: 128.93.162.84 (128.93.162.84), Dst: 10.0.100.65 (10.0.100.65)
▼ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x7d86 [correct]
    Identifier (BE): 33355 (0x824b)
    Identifier (LE): 19330 (0x4b82)
    Sequence number (BE): 46 (0x002e)
    Sequence number (LE): 11776 (0x2e00)
    [Request frame: 295]
    [Response time: 103.508 ms]
  ▶ Data (44 bytes)
```

The screenshot below is the screenshot of the previous ICMP error packets received by the source host.



```
134 7…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=41/10496, ttl=14 (no response found!)
136 7…  193.51.184.177   10.0.100.65      ICMP   70 Time-to-live exceeded (Time to live exceeded in transit)
137 7…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=42/10752, ttl=14 (no response found!)
138 7…  193.51.184.177   10.0.100.65      ICMP   70 Time-to-live exceeded (Time to live exceeded in transit)
139 7…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=43/11008, ttl=15 (no response found!)
217 1…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=44/11264, ttl=15 (no response found!)
252 1…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=45/11520, ttl=15 (no response found!)
295 2…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=46/11776, ttl=16 (reply in 296)
296 2…  128.93.162.84    10.0.100.65      ICMP   86 Echo (ping) reply    id=0x824b, seq=46/11776, ttl=50 (request in 295)
297 2…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=47/12032, ttl=16 (reply in 298)
298 2…  128.93.162.84    10.0.100.65      ICMP   86 Echo (ping) reply    id=0x824b, seq=47/12032, ttl=50 (request in 297)
299 2…  10.0.100.65      128.93.162.84    ICMP   86 Echo (ping) request  id=0x824b, seq=48/12288, ttl=16 (reply in 301)
301 2…  128.93.162.84    10.0.100.65      ICMP   86 Echo (ping) reply    id=0x824b, seq=48/12288, ttl=50 (request in 299)

▶ Frame 138: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▶ Ethernet II, Src: Cisco_99:23:41 (9c:af:ca:99:23:41), Dst: Apple_53:a7:d6 (40:6c:8f:53:a7:d6)
▶ Internet Protocol Version 4, Src: 193.51.184.177 (193.51.184.177), Dst: 10.0.100.65 (10.0.100.65)
▼ Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)
    Code: 0 (Time to live exceeded in transit)
    Checksum: 0xf4ff [correct]
  ▶ Internet Protocol Version 4, Src: 10.0.100.65 (10.0.100.65), Dst: 128.93.162.84 (128.93.162.84)
  ▼ Internet Control Message Protocol
      Type: 8 (Echo (ping) request)
      Code: 0
      Checksum: 0x758a [in ICMP error packet]
      Identifier (BE): 33355 (0x824b)
      Identifier (LE): 19330 (0x4b82)
      Sequence number (BE): 42 (0x002a)
      Sequence number (LE): 10752 (0x2a00)
```

As shown in the two graphs above, the last three received echo packets have type 0, while previous error packets have type 11, which means that for the last three packets, they successfully reaches the destination, while previous packets didn't reach the destination because TTL exceeded.

6. <u>Programming assignment: Host lookup and performance measurement (50 points)</u>

Implementation requirements and suggestions (must read)

1. This problem requires building a directory server, an app-server and an app-client. You can use the provided db-server and do not have to develop it.

2. You may implement the programs in the language of your choice. Provide a short description about compiling/running your program in the README file.

3. During development, it may be simpler to run all the programs on the same machine using the loopback interface. However, for actual measurement, please follow the instructions provided.

4. Please use port numbers between 9000 and 10000 when running your programs.

5. In order for these servers and clients to properly communicate with each other, it is important that all of them agree upon the message format. For this assignment, you can assume that:
   ‘\r’ (Carriage return) indicates end of a line
   ‘\r\n’ (Carriage return + newline) indicates end of a message
   When print messages on the screen, print each part of the message on a new line. See the example below for clarity.

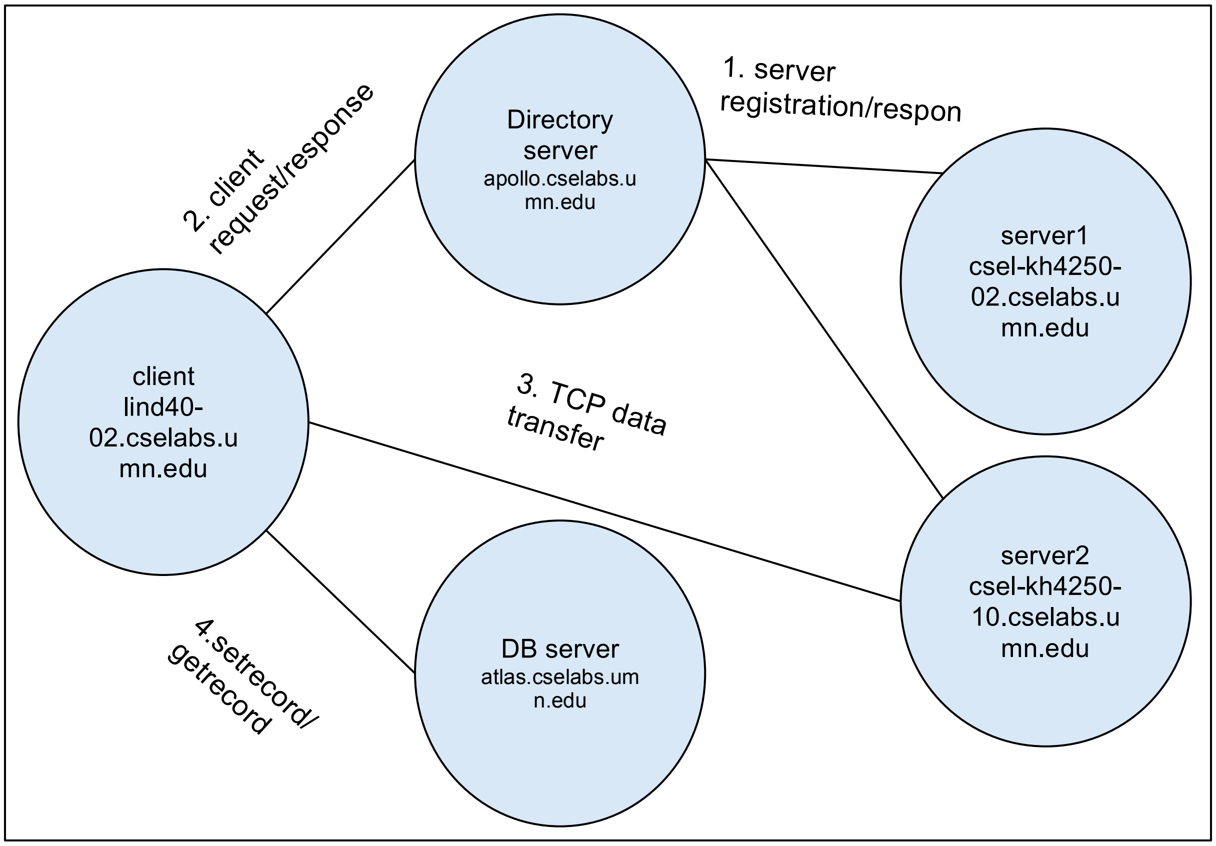
   For example:
   an app-server’s registration message should be formatted like this:
   ```
   “register 128.101.37.1 9123\r\n”
   ```

   an app-client’s list-servers message should be formatted like this:
   ```
   “list-servers\r\n”
   ```

   The directory server’s response to the list-servers message will be like this:
   ```
   “success\r128.101.37.1 9123\r128.101.37.2 9321\r128.101.38.1 9321\r\n”
   ```

   Print the response on the console like this:
   ```
   success
   128.101.37.1 9123
   128.101.37.2 9321
   ```

6. All performance measurement must be performed on the client side.


Directory server
apollo.cselabs.umn.edu
1. server registration/respon
2. client request/response
server1
csel-kh4250-02.cselabs.umn.edu
client
lind40-02.cselabs.umn.edu
3. TCP data transfer
server2
csel-kh4250-10.cselabs.umn.edu
DB server
atlas.cselabs.umn.edu
4.setrecord/ getrecord

For this assignment, you will build a dir-server, an app-server and an app-client. You will transfer data from the app-client to the app-server and measure the performance of the network path between them. You will then upload the performance results to the provided db-server.

Here is the sequence of operations that must be followed:

1. Start the dir-server on a specified port on **`apollo.cselabs.umn.edu`** and the db-server on a specified port on **`atlas.cselabs.umn.edu`**

2. Start two or more app-servers on different UNIX machines in Keller Hall. Each app-server must register itself with the directory server by providing its IP address and port number.

3. Start the app-client on one of the UNIX machines in Lind Hall. The app-client must query the directory server for the list of app servers.

4. The app-client must then connect to one of the app-servers and upload some data over a TCP connection and measure the time taken.

5. The app-client must then connect to the provided db-server and upload the results.

6. Finally, the app-client must download the results from the db-server and display them on the console.

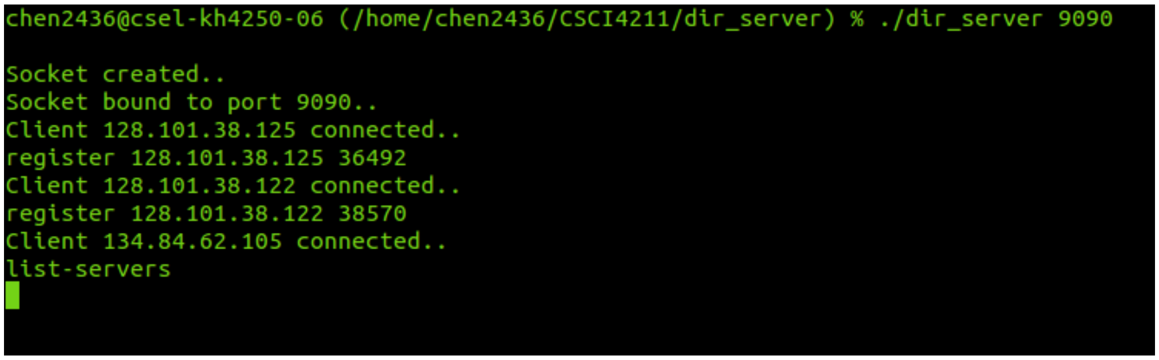
Directory server
You must run the dir-server on **`apollo.cselabs.umn.edu`**.

1. The dir-server will be invoked as follows:
   `$ ./dir-server <ds_port>`
   The dir-server must start listening for TCP connections on the port specified by **`ds_port`**.

2. The app-server will try to register itself by sending the **register message**. The dir-server must respond and indicate success or failure.

3. The app-client will query the dir-server for a list of available servers by sending the **list-servers message**. The dir-server must respond and indicate success or failure. If successful, the list of servers needs to be returned to the client.

4. Print all received messages on the console and provide a screenshot of the same.



```
chen2436@csel-kh4250-06 (/home/chen2436/CSCI4211/dir_server) % ./dir_server 9090

Socket created..
Socket bound to port 9090..
Client 128.101.38.125 connected..
register 128.101.38.125 36492
Client 128.101.38.122 connected..
register 128.101.38.122 38570
Client 134.84.62.105 connected..
list-servers
```

app-server
You must run at least two app-servers. Each app-server must run on one of the UNIX machines in Keller Hall.
The app-server will be invoked as follows:
`$ ./app-server <ds_port>`
The app-server must:

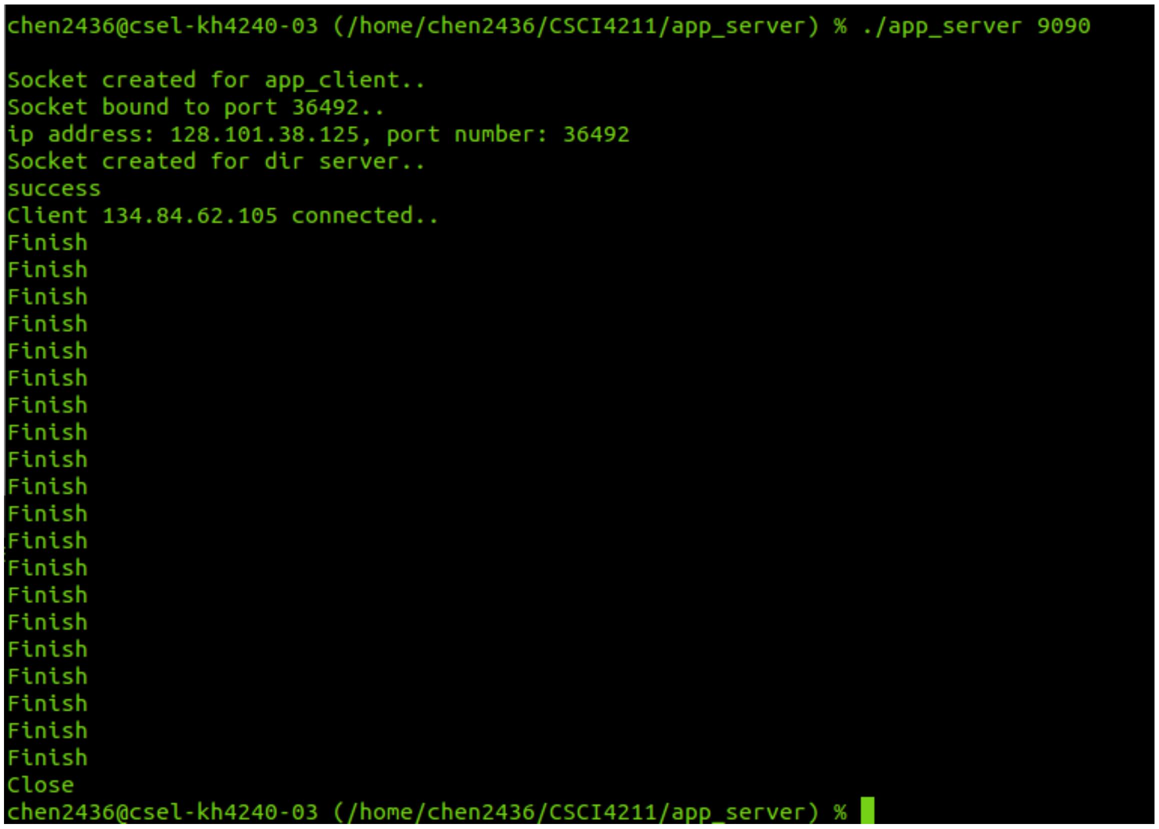
1. Create a TCP socket bound to a random port assigned by the operating system. (Hint: Set **`sin_port = htons(0);`** before calling **`bind()`**)

2. Print the following message on the console (without the quotes):
   “`<ip_address>, <port>`”
   `<ip_address>` is the address of the machine on which the app-server is running, and **`<port>`** is the port number which has been assigned to this socket by the operating system.

(Hint: Use **`getsockname()`** or equivalent)

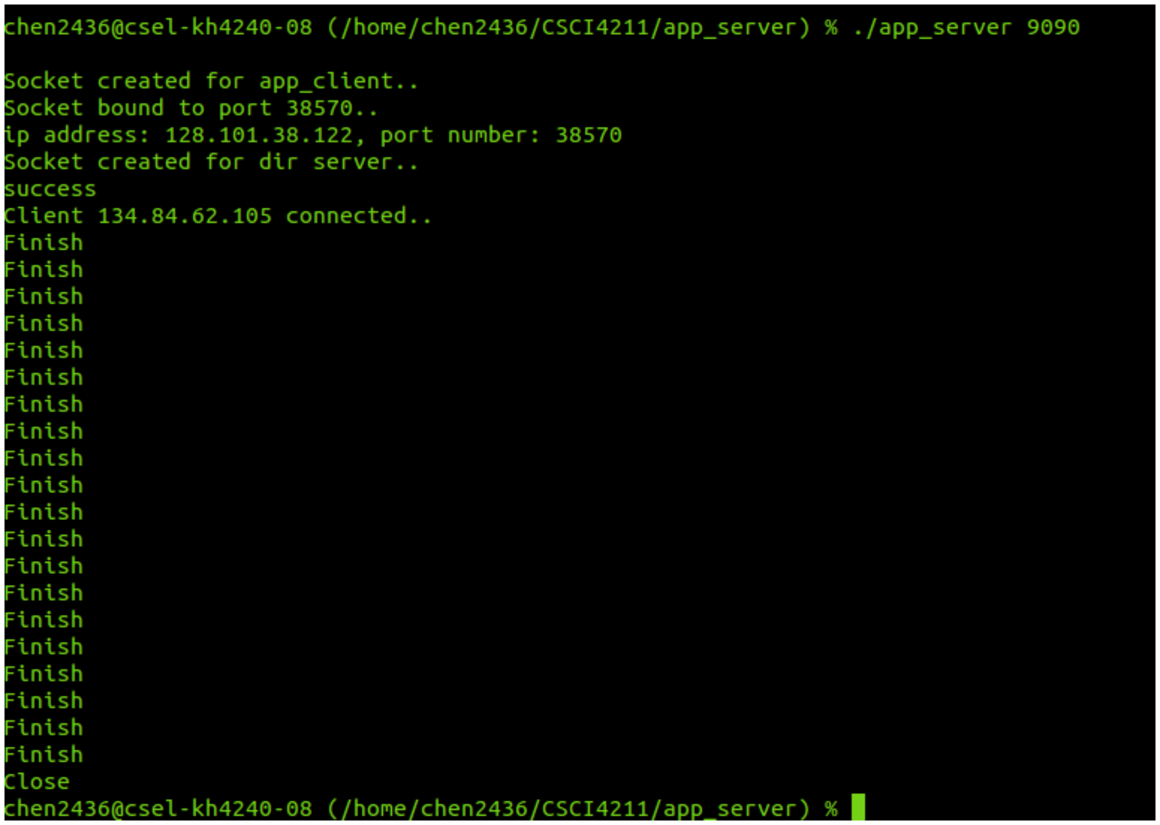
3. Establish a TCP connection with the dir-server, which should be running on **`apollo.cselabs.umn.edu`** on the port specified by **`ds_port`**.
4. Register with the dir-server, by sending the **register message**, using the IP address and port number determined in step 2. Print the response from the dir-server on the console.
5. Listen for app-clients on the socket created in step 1. An app-client will connect to this app-server and upload some data. The app-server must respond with an acknowledgement after all the data has been uploaded.
6. Print all received messages on the console and provide a screenshot of the same.

Server1:



```
chen2436@csel-kh4240-03 (/home/chen2436/CSCI4211/app_server) % ./app_server 9090

Socket created for app_client..
Socket bound to port 36492..
ip address: 128.101.38.125, port number: 36492
Socket created for dir server..
success
Client 134.84.62.105 connected..
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Close
chen2436@csel-kh4240-03 (/home/chen2436/CSCI4211/app_server) %
```

Server 2:


```
chen2436@csel-kh4240-08 (/home/chen2436/CSCI4211/app_server) % ./app_server 9090

Socket created for app_client..
Socket bound to port 38570..
ip address: 128.101.38.122, port number: 38570
Socket created for dir server..
success
Client 134.84.62.105 connected..
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Finish
Close
chen2436@csel-kh4240-08 (/home/chen2436/CSCI4211/app_server) %
```

app-client

You must run the app-client on one of the machines in Lind Hall. The app-client will be invoked as follows:

`$ ./app-client <ds_port> <db_port>`

The app-client must:

1. Connect to the dir-server and obtain a list of available app-servers by sending the **list-servers message.** Print the response from the dir-server on the console.

2. Establish a TCP connection to one of the app-servers in the list by using the provided IP address and port number. Upload **`10KB`** of data to the app-server. The app-server will respond with an acknowledgement after all the data has been uploaded. Measure the time taken for transmission (Time from start of transmission till reception of application level acknowledgement; Do not include time for I/O). Repeat this operation at least five (5) times and compute the average time taken for upload.

3. Upload the performance measurements to the db-server by sending the **set-record message.** Print the response from the db-server on the console.

4. Repeat step 3 for different data sizes **`(10KB, 100KB, 1000KB and 10000KB)`**

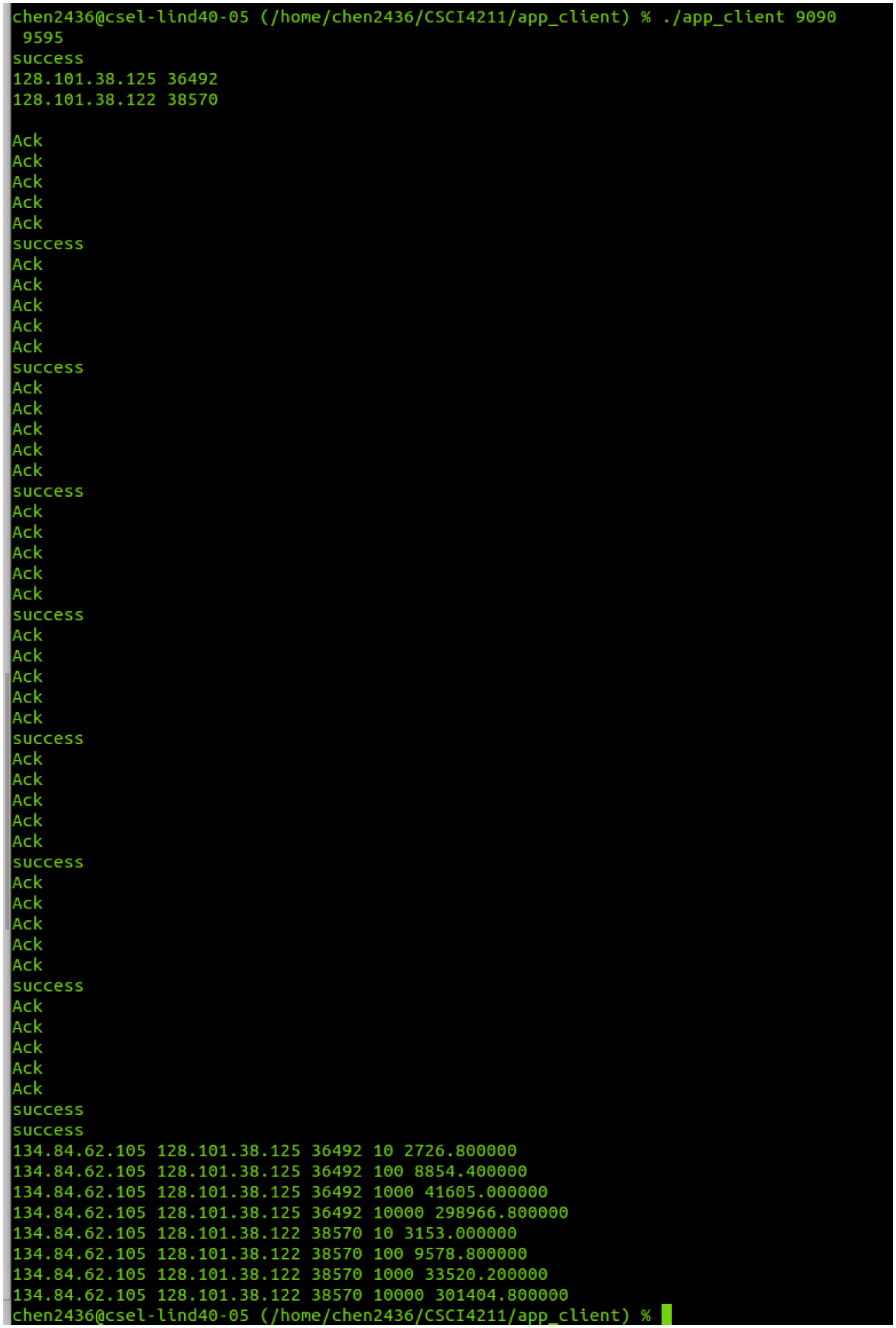
5. Connect to the other app-server and repeat steps (2), (3) and (4).

6. Connect to the db-server which should be running on **`atlas.cselabs.umn.edu`** on the port specified by **`db_port`**.

7. Fetch records from the db-server by sending a **get-records message** to the db-server. Print the response from the db-server on the console.

8. Print all received messages on the console. Attach one or more screenshots showing the results of steps (1), (3) and (7).

The performance measurement are shown in the screenshot below.



```
chen2436@csel-lind40-05 (/home/chen2436/CSCI4211/app_client) % ./app_client 9090
 9595
success
128.101.38.125 36492
128.101.38.122 38570

Ack
Ack
Ack
Ack
Ack
success
Ack
Ack
Ack
Ack
Ack
success
Ack
Ack
Ack
Ack
Ack
success
Ack
Ack
Ack
Ack
Ack
success
Ack
Ack
Ack
Ack
Ack
success
Ack
Ack
Ack
Ack
Ack
success
Ack
Ack
Ack
Ack
Ack
success
Ack
Ack
Ack
Ack
Ack
success
success
134.84.62.105 128.101.38.125 36492 10 2726.800000
134.84.62.105 128.101.38.125 36492 100 8854.400000
134.84.62.105 128.101.38.125 36492 1000 41605.000000
134.84.62.105 128.101.38.125 36492 10000 298966.800000
134.84.62.105 128.101.38.122 38570 10 3153.000000
134.84.62.105 128.101.38.122 38570 100 9578.800000
134.84.62.105 128.101.38.122 38570 1000 33520.200000
134.84.62.105 128.101.38.122 38570 10000 301404.800000
chen2436@csel-lind40-05 (/home/chen2436/CSCI4211/app_client) %
```

db-server

You must run the db-server on `atlas.cselabs.umn.edu`.

You may use the provided db-server - You **DO NOT** have to develop your own.

You can start the db-server by executing:

```
$ ./db-server <db_port>
```

1. An app-client can query the db-server for a list of available records by sending the **get-records message**.

2. An app-client can try to set a record in the db-server by sending the **set-record message.**

3. The db-server will store all the records in a plaintext file called `client_records.dat`. Delete this file if you want to clear the records in the db-server and start again.

Message type	register message
Description	This message is sent by the app-server to the dir-server to register itself. The dir-server must respond with a **“`success\r\n`”** or **“`failure\r\n`”** **`<ip_address>`** is the IP address of the app-server **`<port>`** is the port on which the app-server is listening
Sender	app-server
Receiver	dir-server
Message format	`“register <ip_address> <port>\r\n”`
Message response	`“success\r\n” or “failure\r\n”`
Example request	`“register 128.101.37.1 9123\r\n”`
Example response	`“success\r\n”`

Message type	list-servers message
Description	This message is sent by the app-client to the dir-server. The dir-server must respond with failure if no app-servers are available or indicate success and send a list of app-server IP addresses and port numbers back to the app-client.
1q`	app-client
Receiver	dir-server
Message format	`“list-servers\r\n”`
Message response	`“success\r<server1_IP> <server1_port>\r<server2_IP> <server2_port>\r\n” or “failure\r\n”`
Example request	`“list-servers\r\n”`
Example response	`“success\r128.101.37.1 9123\r128.101.38.1 9321\r\n”`

Message type	set-record
Description	This message is sent by the app-client to the db-server. The db-server will respond with failure if it is unable to save the record or will respond with success.
Sender	app-client
Receiver	db-server
Message format	`“set-record <client_IP> <server_IP> <port> <data_size> <upload_time>\r\n”`
Message response	`“success\r\n” or “failure\r\n”`
Example request	`“set-record 10.10.10.10 20.20.20.20 9123 10 0.05\r\n”`
Example response	`“success\r\n”`

Message type	get-records
Description	This message is sent by the app-client to the db-server. The db-server will respond with failure if it is unable to fetch the records or will respond with success and a list of available records.
Sender	app-client
Receiver	db-server
Message format	`“get-records\r\n”`
Message response	`“success\r<client_IP> <server_IP> <port> <data_len> <time>\r\n” or “failure\r\n”`
Example request	`“get-records\r\n”`
Example response	`“success\r10.10.10.10 20.20.20.20 9123 10 0.05\r10.10.10.10 20.20.20.20 9123 100 0.5\r\n”`

Deliverables:

1. You must upload a single archive file (.zip or .tar or .tar.gz) on moodle.
   When extracted, the archive file must be a single folder. The name of the folder should be your **student ID**. The folder should contain the following files:
   - Readme
   - app-server source files
   - app-client source files
   - MS Word/PDF document
2. DO NOT include the test files

For example, here is a sample submission:

```
1234567/
    Readme
    PA3.doc
    app-server/
        app-server.c
        Makefile
    app-client/
        app-client.c
        Makefile
    dir-server/
        dir-server.c
        Makefile
```

You can create an archive file from the contents of the above directory as follows:

```
$ tar cvf 1234567.tar.gz 1234567/
```

Grading:

README, Makefiles, comments, readability: **3 points**
Packaging the submission as specified: **2 points**
app-server registration & screenshots: **5 points**
app-client list-servers screenshot: **10 points**
app-client data upload and performance measurement: **10 points**
app-client set-records: **10 points**
app-client get-records screenshot: **10 points**