CSCI4211: Introduction to Computer Networks

Homework Assignment III

Due 11:55PM Nov 18th, 2015

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

Name: _Mohamed Yunis   Student ID: _4597863_

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**)

    (12 * 16 + 11). (4 * 16 + 4). (10 * 16 + 15). (10 * 16 + 2)
    203.68.175.162

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

    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**)

    255.255.192.0 = 11111111.11111111.11000000.00000000
    14 bits left for the host identification.
    Maximum number of hosts = 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

    129.17.129.96/98,
    129.17.129.104/98,
    129.17.129.112/98,
    129.17.129.120/98

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)

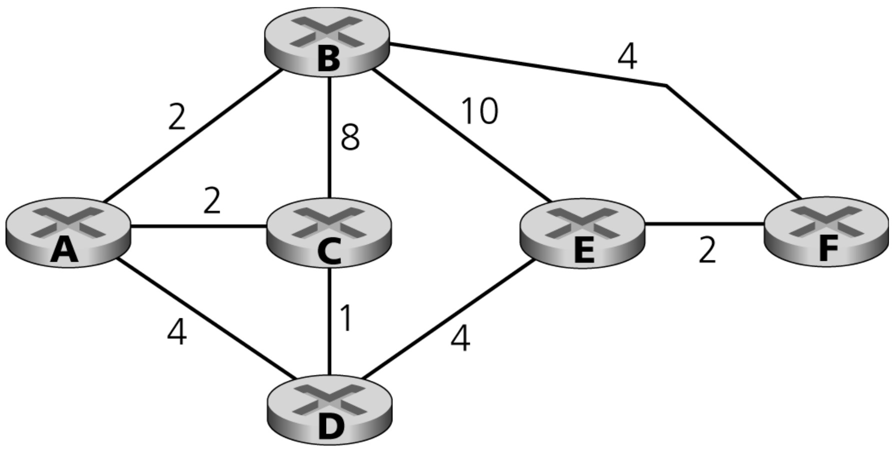
Prefix (binary)	*Interface*
1	*0*
101	*1*
101110	*2*
Otherwise	*3*

10000000 through 10011111
10100000 through 10110111
10111000 through 11111111
Otherwise(128 address)
Number of hosts of interface 2 = 2^(8-6) – 2 = 2
Number of hosts of interface 1 = 2^(8-3) – 2^2 – 2 = 26
Number of hosts of interface 0 = 2^(8-1) – 2^5 - 2 = 94

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**)?
The shortest path is ABF and the cost of that path is 6.

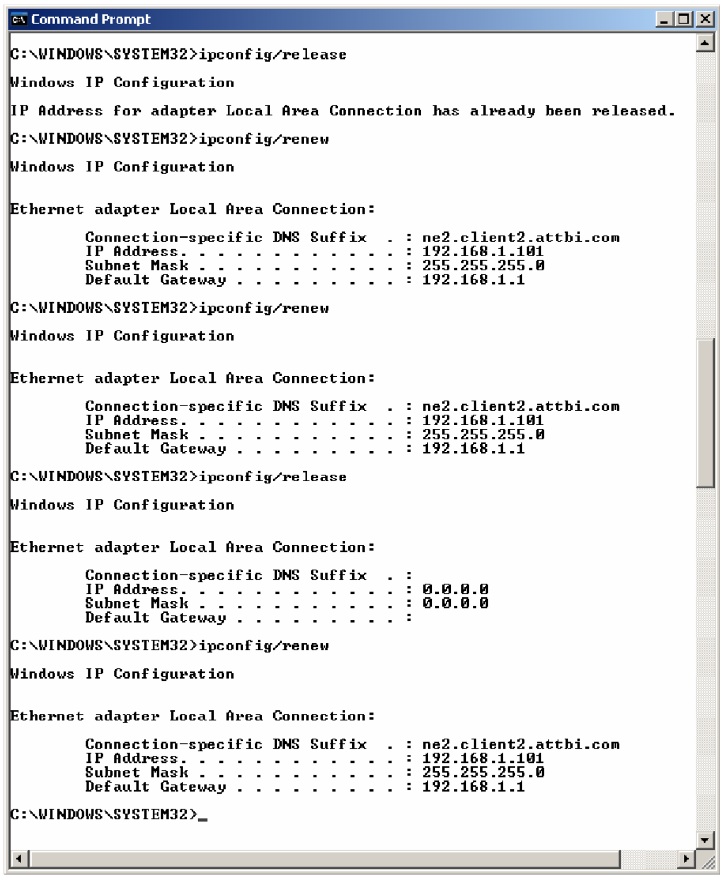
Step	**N**	**D(B),p(B)**	**D(C),p(C)**	**D(D),p(D)**	**D(E),p(E)**	**D(F),p(F)**
0	A	2,A	2,A	4,A	Infinite	infinite
1	AB	2,A	2,A	4,A	12,B	6,B
2	ABC	2,A	2,A	3,C	12,B	6,B
3	ABCD	2,A	2,A	3,C	7,D	6,B
4	ABCDE	2,A	2,A	3,C	7,D	6,B
5	ABCDEF	2,A	2,A	3,C	7,D	6,B



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:

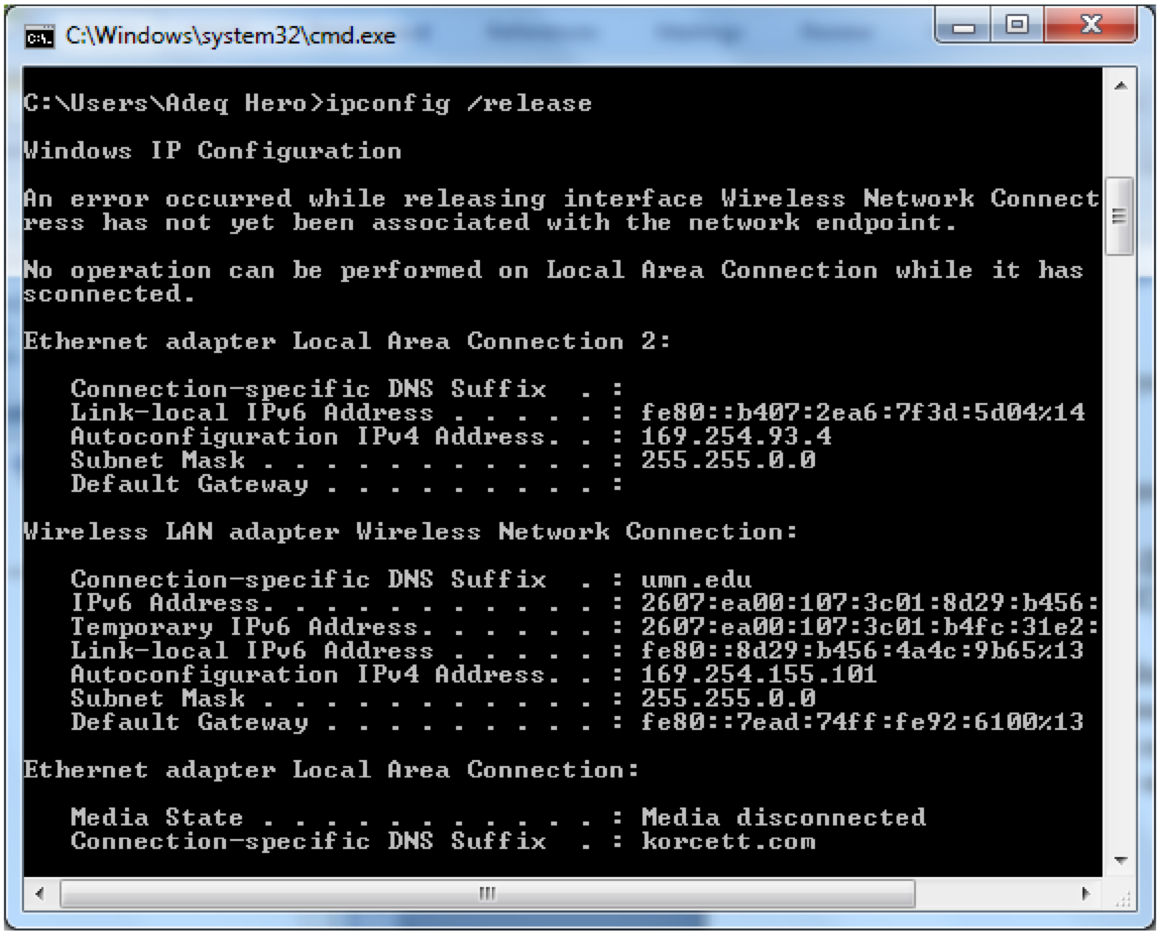
1. Enter "*ipconfig /release*" to releases your current IP address, so that your host's IP address becomes 0.0.0.0. (sudo dhclient –r is used in linux)
2. Start up the Wireshark packet sniffer, with *"bootp"* as the filter (Note to see DHCP packets in the current version of Wireshark, you need to enter *"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>_
```

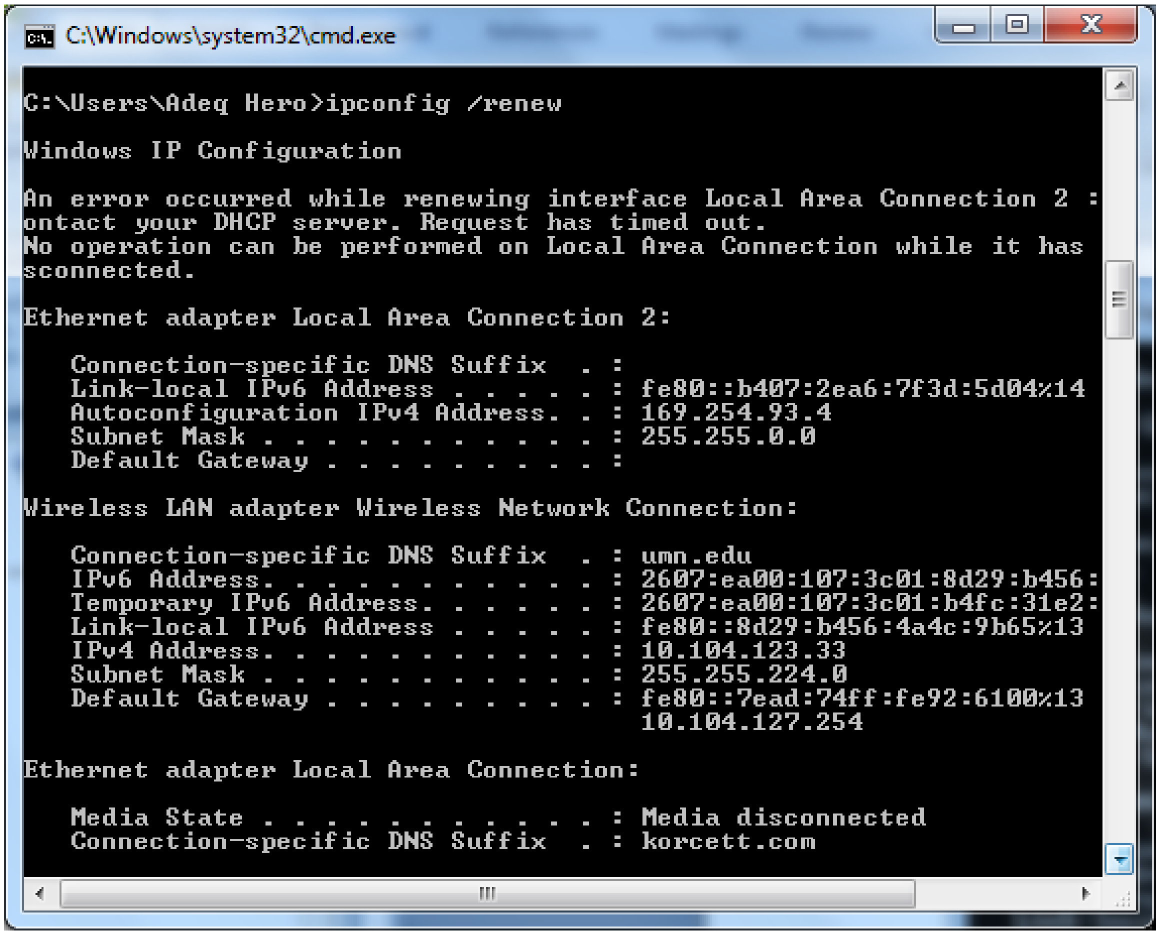
Provide two screen shots: command line screen similar to the figure above and a wireshark screen that captures the DHCP interaction (you can have similar screenshots in unix)

The two screen below show the result of ipconfig/release and ipconfig/renew
The detail that applies to this assignment is found in the IPV4 section of the "Wireless LAN adapter Wireless Network Connection" since I was connected to the wireless network during the operartions.



```
C:\Users\Adeq Hero>ipconfig /release

Windows IP Configuration

An error occurred while releasing interface Wireless Network Connect
ress has not yet been associated with the network endpoint.

No operation can be performed on Local Area Connection while it has
sconnected.

Ethernet adapter Local Area Connection 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b407:2ea6:7f3d:5d04%14
   Autoconfiguration IPv4 Address. . : 169.254.93.4
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . : umn.edu
   IPv6 Address. . . . . . . . . . . : 2607:ea00:107:3c01:8d29:b456:
   Temporary IPv6 Address. . . . . . : 2607:ea00:107:3c01:b4fc:31e2:
   Link-local IPv6 Address . . . . . : fe80::8d29:b456:4a4c:9b65%13
   Autoconfiguration IPv4 Address. . : 169.254.155.101
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::7ead:74ff:fe92:6100%13

Ethernet adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : korcett.com
```

On this screen you cannot see the information for the IPV4 sine its released.



```
C:\Users\Adeq Hero>ipconfig /renew

Windows IP Configuration

An error occurred while renewing interface Local Area Connection 2 :
ontact your DHCP server. Request has timed out.
No operation can be performed on Local Area Connection while it has
sconnected.

Ethernet adapter Local Area Connection 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b407:2ea6:7f3d:5d04%14
   Autoconfiguration IPv4 Address. . : 169.254.93.4
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . : umn.edu
   IPv6 Address. . . . . . . . . . . : 2607:ea00:107:3c01:8d29:b456:
   Temporary IPv6 Address. . . . . . : 2607:ea00:107:3c01:b4fc:31e2:
   Link-local IPv6 Address . . . . . : fe80::8d29:b456:4a4c:9b65%13
   IPv4 Address. . . . . . . . . . . : 10.104.123.33
   Subnet Mask . . . . . . . . . . . : 255.255.224.0
   Default Gateway . . . . . . . . . : fe80::7ead:74ff:fe92:6100%13
                                       10.104.127.254

Ethernet adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : korcett.com
```

On this other screen you can see the IPV4 information after renewing the ip adderss. Based on the screenshots, answer the following questions:

1. Are DHCP messages sent over UDP or TCP? (**2pts**)

   DHCP messages are sent over UDP

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

   The router line indicates where the client should send messages by default. The subnet mask line tells which subnet should be used

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

   The purpose is to tell how long the DHCP assigns ip address before it is assigned to another client.

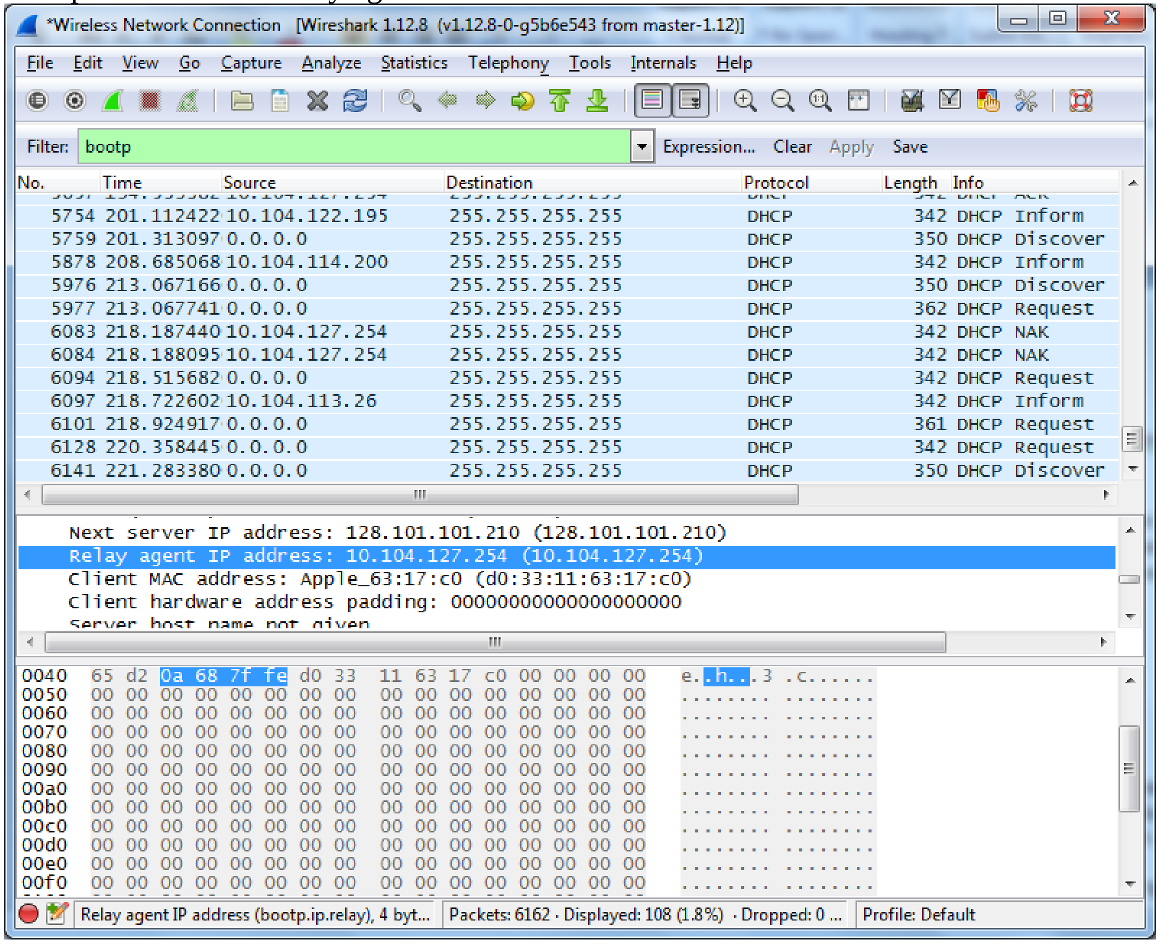
4. Explain the purpose of the DHCP release message? What would happen if the client's DHCP release message is lost? (**2pts**)

   The purpose of the release message is to release the assigned ip to the client back to the DHCP server.

If the message is lost that ip gets released by the client but the server reassign that address back until the client lease on the IP expires.

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**)

   There is a relay agent: look at the screen shot.
   The ip address of the relay agent is: 10.104.127.254


*Wireless Network Connection [Wireshark 1.12.8 (v1.12.8-0-g5b6e543 from master-1.12)]

Filter: bootp

```
No.   Time        Source          Destination      Protocol  Length Info
5754 201.112422 10.104.122.195  255.255.255.255  DHCP        342 DHCP Inform
5759 201.313097 0.0.0.0         255.255.255.255  DHCP        350 DHCP Discover
5878 208.685068 10.104.114.200  255.255.255.255  DHCP        342 DHCP Inform
5976 213.067166 0.0.0.0         255.255.255.255  DHCP        350 DHCP Discover
5977 213.067741 0.0.0.0         255.255.255.255  DHCP        362 DHCP Request
6083 218.187440 10.104.127.254  255.255.255.255  DHCP        342 DHCP NAK
6084 218.188095 10.104.127.254  255.255.255.255  DHCP        342 DHCP NAK
6094 218.515682 0.0.0.0         255.255.255.255  DHCP        342 DHCP Request
6097 218.722602 10.104.113.26   255.255.255.255  DHCP        342 DHCP Inform
6101 218.924917 0.0.0.0         255.255.255.255  DHCP        361 DHCP Request
6128 220.358445 0.0.0.0         255.255.255.255  DHCP        342 DHCP Request
6141 221.283380 0.0.0.0         255.255.255.255  DHCP        350 DHCP Discover
```

```
Next server IP address: 128.101.101.210 (128.101.101.210)
Relay agent IP address: 10.104.127.254 (10.104.127.254)
Client MAC address: Apple_63:17:c0 (d0:33:11:63:17:c0)
Client hardware address padding: 00000000000000000000
```

```
0040  65 d2 0a 68 7f fe d0 33  11 63 17 c0 00 00 00 00   e..h...3 .c......
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

Relay agent IP address (bootp.ip.relay), 4 byt... | Packets: 6162 · Displayed: 108 (1.8%) · Dropped: 0 ... | Profile: Default

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

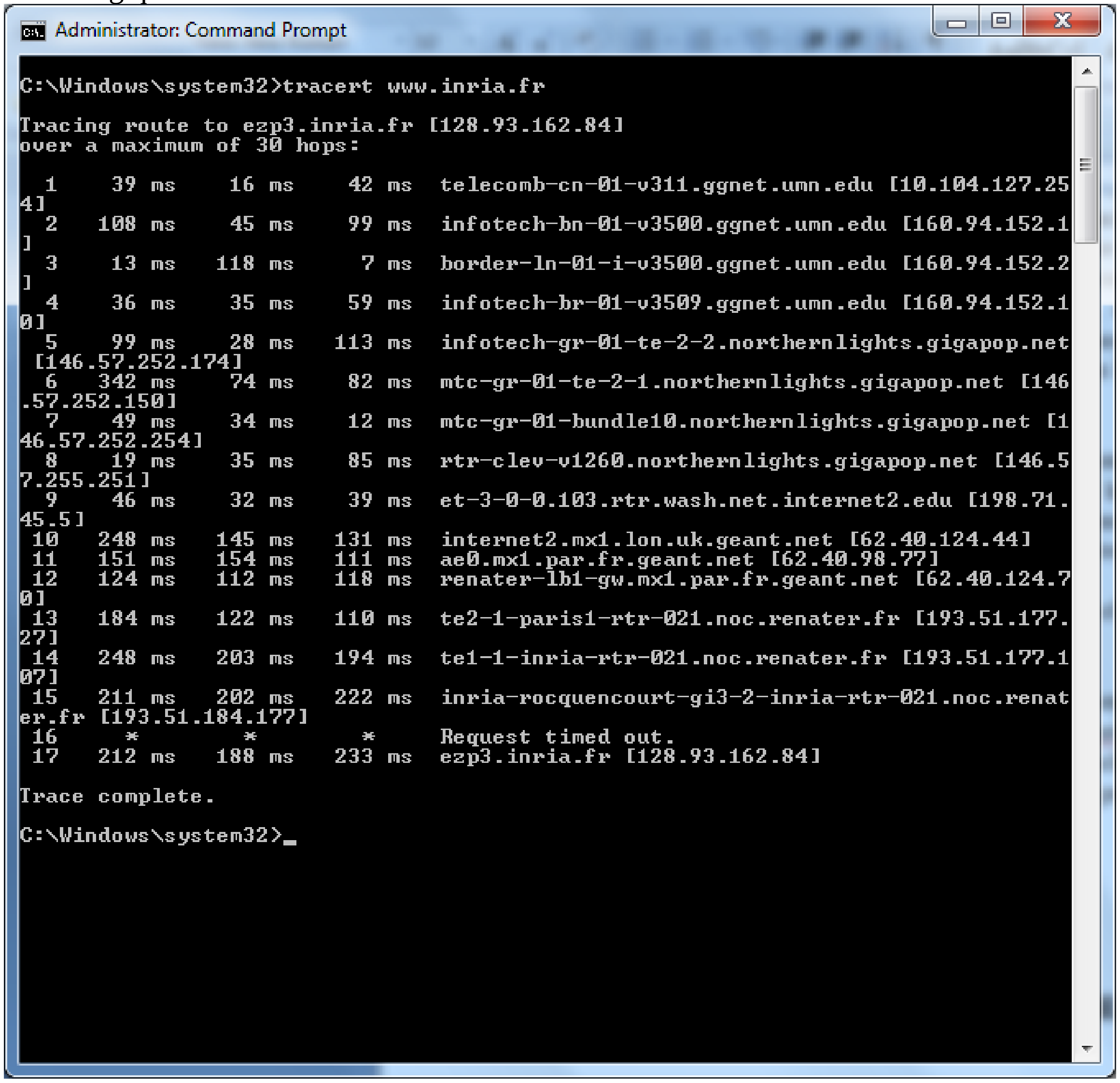
In this practice, we capture the packets generated by the Traceroute program. You may recall that the Traceroute program can be used to figure out the path a packet takes from source to destination. Traceroute is discussed in Section 1.3 and in Section 4.4 of the text.

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:

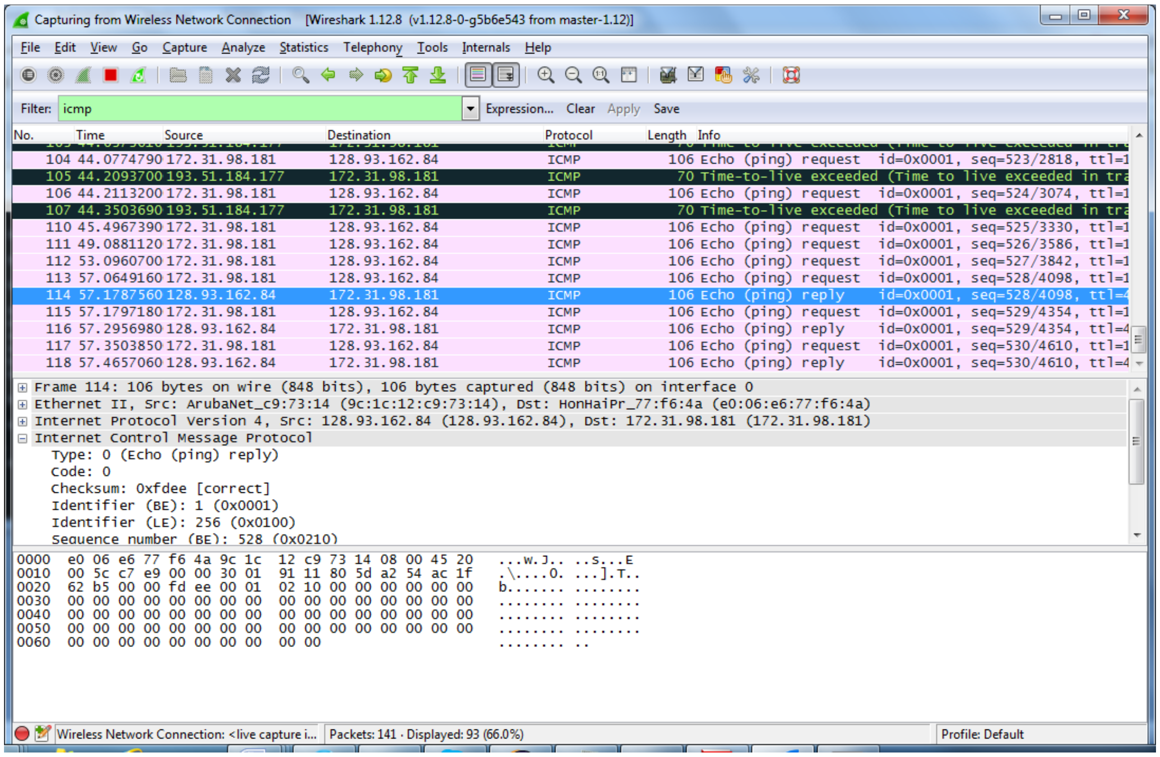


```
C:\Windows\system32>tracert www.inria.fr

Tracing route to ezp3.inria.fr [128.93.162.84]
over a maximum of 30 hops:

  1    39 ms    16 ms    42 ms  telecomb-cn-01-v311.ggnet.umn.edu [10.104.127.25
4]
  2   108 ms    45 ms    99 ms  infotech-bn-01-v3500.ggnet.umn.edu [160.94.152.1
]
  3    13 ms   118 ms     7 ms  border-ln-01-i-v3500.ggnet.umn.edu [160.94.152.2
]
  4    36 ms    35 ms    59 ms  infotech-br-01-v3509.ggnet.umn.edu [160.94.152.1
0]
  5    99 ms    28 ms   113 ms  infotech-gr-01-te-2-2.northernlights.gigapop.net
 [146.57.252.174]
  6   342 ms    74 ms    82 ms  mtc-gr-01-te-2-1.northernlights.gigapop.net [146
.57.252.150]
  7    49 ms    34 ms    12 ms  mtc-gr-01-bundle10.northernlights.gigapop.net [1
46.57.252.254]
  8    19 ms    35 ms    85 ms  rtr-clev-v1260.northernlights.gigapop.net [146.5
7.255.251]
  9    46 ms    32 ms    39 ms  et-3-0-0.103.rtr.wash.net.internet2.edu [198.71.
45.5]
 10   248 ms   145 ms   131 ms  internet2.mx1.lon.uk.geant.net [62.40.124.44]
 11   151 ms   154 ms   111 ms  ae0.mx1.par.fr.geant.net [62.40.98.77]
 12   124 ms   112 ms   118 ms  renater-lb1-gw.mx1.par.fr.geant.net [62.40.124.7
0]
 13   184 ms   122 ms   110 ms  te2-1-paris1-rtr-021.noc.renater.fr [193.51.177.
27]
 14   248 ms   203 ms   194 ms  te1-1-inria-rtr-021.noc.renater.fr [193.51.177.1
07]
 15   211 ms   202 ms   222 ms  inria-rocquencourt-gi3-2-inria-rtr-021.noc.renat
er.fr [193.51.184.177]
 16     *        *        *     Request timed out.
 17   212 ms   188 ms   233 ms  ezp3.inria.fr [128.93.162.84]

Trace complete.

C:\Windows\system32>_
```

1. How many ICMP echo packets are sent? Justify the observed number from wireshark, based on the output of tracert command. (**4pts**)

   Three attempts were made for each time to live group [0 -17] so there were a total of 18 * 3 = 54 ICMP echo packets.

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 last three ICMP echo replies were sent from the destination which the original tracert command was targeted to i.e. inria.fr. They indicate the control packet reached the destination before the time to live ended. Examining these packets on wireshark shows that they have type 0, which proves this point. It shows that there were no errors. The window below shows the screenshot of the Wireshark window.



```
Filter: icmp

No.  Time        Source          Destination    Protocol  Length Info
104 44.0774790 172.31.98.181   128.93.162.84  ICMP      106 Echo (ping) request  id=0x0001, seq=523/2818, ttl=1
105 44.2093700 193.51.184.177  172.31.98.181  ICMP       70 Time-to-live exceeded (Time to live exceeded in tra
106 44.2113200 172.31.98.181   128.93.162.84  ICMP      106 Echo (ping) request  id=0x0001, seq=524/3074, ttl=1
107 44.3503690 193.51.184.177  172.31.98.181  ICMP       70 Time-to-live exceeded (Time to live exceeded in tra
110 45.4967390 172.31.98.181   128.93.162.84  ICMP      106 Echo (ping) request  id=0x0001, seq=525/3330, ttl=1
111 49.0881120 172.31.98.181   128.93.162.84  ICMP      106 Echo (ping) request  id=0x0001, seq=526/3586, ttl=1
112 53.0960700 172.31.98.181   128.93.162.84  ICMP      106 Echo (ping) request  id=0x0001, seq=527/3842, ttl=1
113 57.0649160 172.31.98.181   128.93.162.84  ICMP      106 Echo (ping) request  id=0x0001, seq=528/4098, ttl=1
114 57.1787560 128.93.162.84   172.31.98.181  ICMP      106 Echo (ping) reply    id=0x0001, seq=528/4098, ttl=4
115 57.1797180 172.31.98.181   128.93.162.84  ICMP      106 Echo (ping) request  id=0x0001, seq=529/4354, ttl=1
116 57.2956980 128.93.162.84   172.31.98.181  ICMP      106 Echo (ping) reply    id=0x0001, seq=529/4354, ttl=4
117 57.3503850 172.31.98.181   128.93.162.84  ICMP      106 Echo (ping) request  id=0x0001, seq=530/4610, ttl=1
118 57.4657060 128.93.162.84   172.31.98.181  ICMP      106 Echo (ping) reply    id=0x0001, seq=530/4610, ttl=4

⊞ Frame 114: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
⊞ Ethernet II, Src: ArubaNet_c9:73:14 (9c:1c:12:c9:73:14), Dst: HonHaiPr_77:f6:4a (e0:06:e6:77:f6:4a)
⊞ Internet Protocol Version 4, Src: 128.93.162.84 (128.93.162.84), Dst: 172.31.98.181 (172.31.98.181)
⊟ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xfdee [correct]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 528 (0x0210)

0000  e0 06 e6 77 f6 4a 9c 1c  12 c9 73 14 08 00 45 20   ...w.J.. ..s...E
0010  00 5c c7 e9 00 00 30 01  91 11 80 5d a2 54 ac 1f   .\....0. ...].T..
0020  62 b5 00 00 fd ee 00 01  02 10 00 00 00 00 00 00   b....... ........
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060  00 00 00 00 00 00 00 00  00 00                     ........ ..

Wireless Network Connection: <live capture i...   Packets: 141 · Displayed: 93 (66.0%)   Profile: Default
```

Capturing from Wireless Network Connection [Wireshark 1.12.8 (v1.12.8-0-g5b6e543 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

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.

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 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.

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.

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).

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.
Sender	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**