*Project report*

# Final Project
# DHCP Server & Client

Course Title: Internet Applications

Name: WANG Yinuo (2014213051)

LIU Sihong (2014213142)

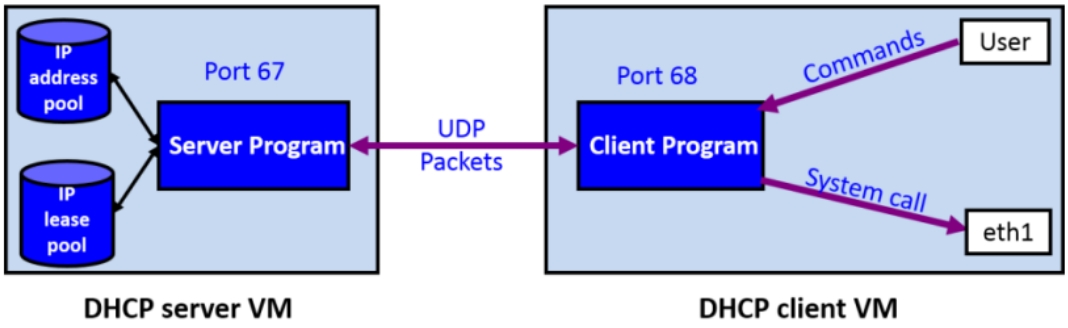
Date: June 15th, 2017.

# Content

# 1. Overview

## 1.1 General Requirements

The project requires us to implement DHCP Client and DHCP Server program with command line interface under Linux operating system. And the communication between the two systems should be achieved by the programs with the following program model.




DHCP Server & Client Program Model

- ❖ Support DHCP messages: DISCOVER/OFFER/REQUEST/ACK, RELEASE, REQUEST/NAK, REQUEST/ACK, INFORM/ACK.
- ❖ Support DHCP options:

- 1 (Subnet Mask Value)
- 3 (Router addresses)
- 6 (DNS Server addresses)
- 51 (IP Address Lease Time)
- 53 (DHCP Message Type)
- 54 (DHCP Server Identification)
- 55 (Parameter request list)
- 58 (DHCP Renewal Time T1)
- 59 (DHCP Rebinding Time T2)
- 60 (Class Identifier, please set as your student number)
- 255 (END)

- ❖ Four messages during address acquisition can be delivered on broadcast packets.
- ❖ Support the following DHCP Procedures:

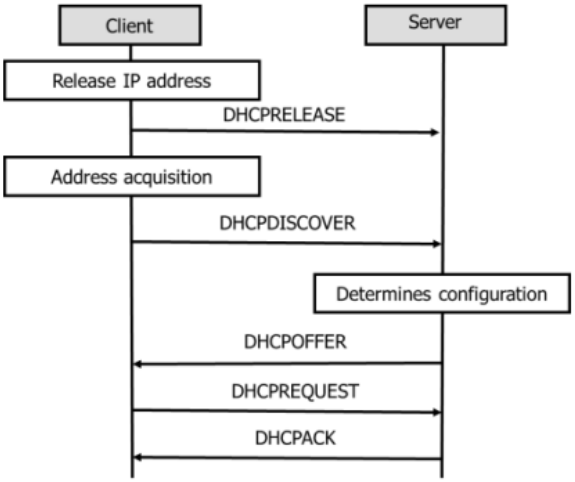



Fig. 1 release + address acquisition

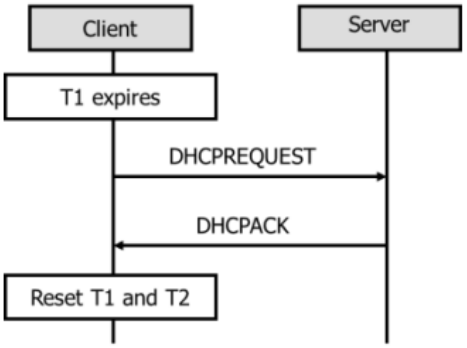



Fig. 2 successful lease renew

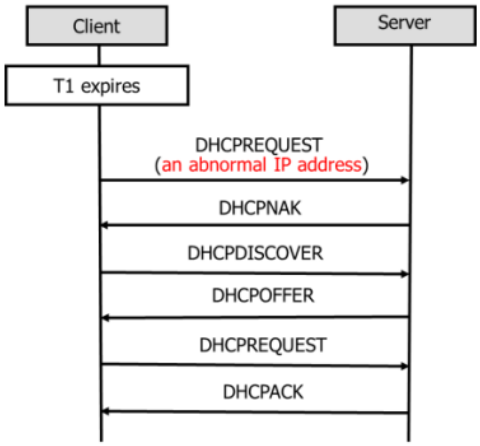



Fig. 3 failed lease renew (with an abnormal IP address)+address acquisition again

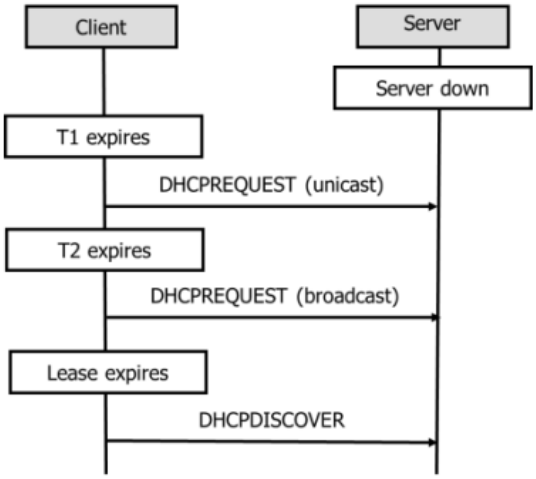



Fig. 4 failed lease renew (server down)

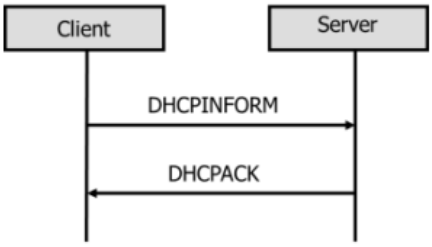



Fig. 5 inform

## 1.2 Goals

- ❖ Deeply understand the details of DHCP (Dynamic Host Configuration Protocol).
- ❖ Complete a DHCP server program and run it in one Ubuntu virtual machine.
- ❖ Complete a DHCP client program and run it in another Ubuntu virtual machine.

# 2. Requirements Analysis

## 2.1 Preparation

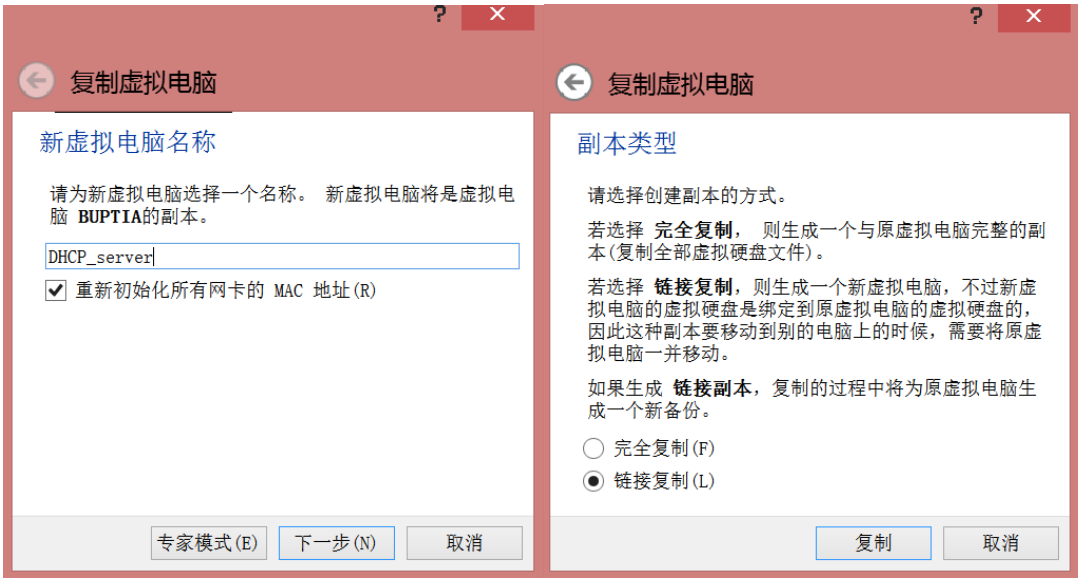
The environment for this project is as follows:

- C language
- Linux operation system
- gcc compiler and gdb debug tool

### 2.1.1 Virtual Machine

To satisfy those requirements above, we need to create two virtual systems firstly, one for DHCP Server and the other for DHCP Client.

- ❖ Run the Oracle VM, and there has been an Ubuntu system named BUPTIA which is used as DHCP Client in our project.
- ❖ Then copy this system and set necessary configurations to create the DHCP Server, showing as follows.




Then there are two virtual systems that we can use as DHCP Client and DHCP Server.

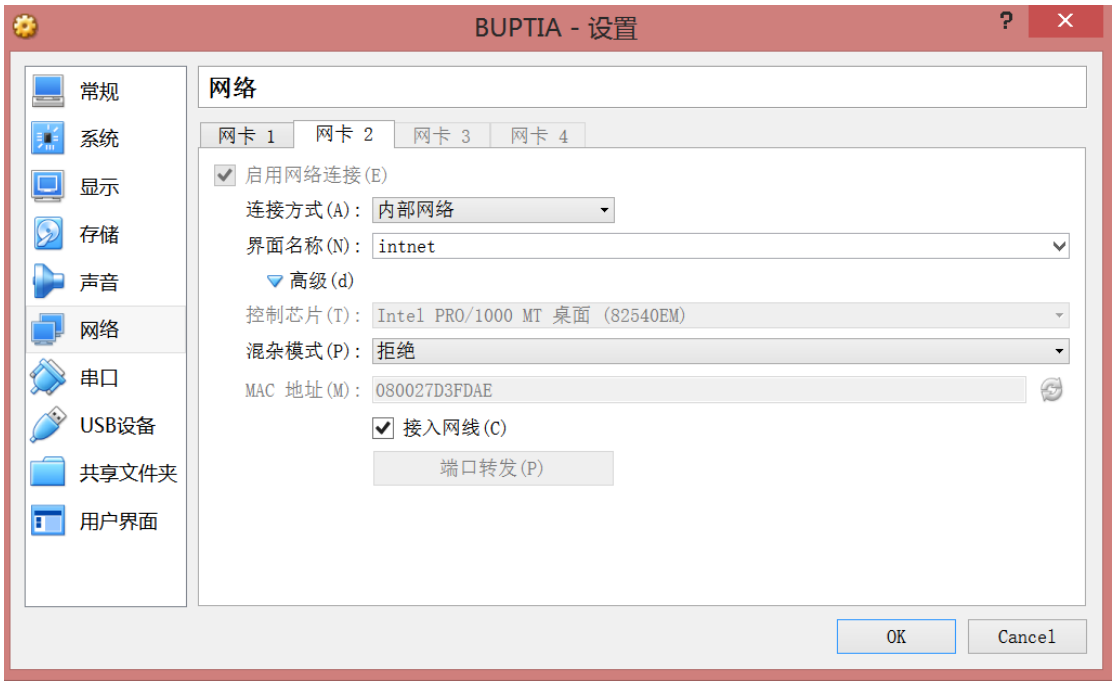

### 2.1.2 DHCP Client

Preparations of DHCP Client are as follows.

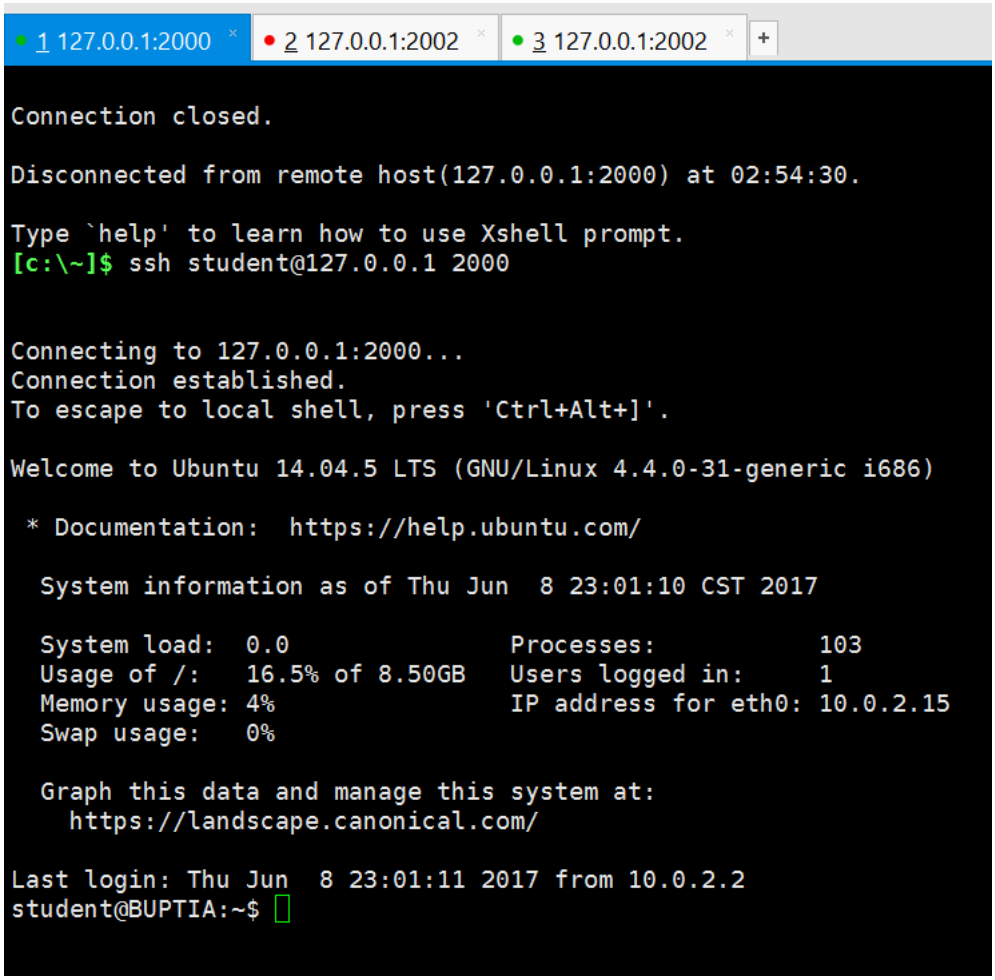
❖ Set NIC1 for DHCP Client.

- Click the button “Port Forwarding” and add a port forwarding rule.
- Then set the Host IP as 127.0.0.1, the Host Port as 2000, the Subsystem IP as 0.0.0.0, and the Subsystem Port as 22.


- Set NIC2 for DHCP Client.




- Run the DHCP Client system, open XShell and connect it to 127.0.0.1 port 2000.




❖ Change network configurations, save and restart the system.

```
student@BUPTIA:~$ sudo vim /etc/network/interfaces
[sudo] password for student:

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

# The host-only interface
auto eth1
iface eth1 inet static
address 0.0.0.0
netmask 0.0.0.0
~
```

❖ Add new iptables rule.

This step is necessary because in the Linux System, when the destination IP address is 255.255.255.255, which is a broadcast address, packages can only be handled by using 0.0.0.0 as the source IP address. And the iptables is used to change the source IP address of broadcast packages.

```
student@BUPTIA:~$ sudo iptables -t nat -A POSTROUTING -d 255.255.255.255 -o eth1 -j SNAT --to-source 0.0.0.0
student@BUPTIA:~$ sudo iptables-save > /etc/network/iptables.rules
```

❖ Change “exit 0” to “iptables-restore < /etc/network/iptables.rules
exit 0”

```
student@BUPTIA:~$ sudo vim /etc/rc.local

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
iptables-restore < /etc/network/iptables.rules
exit 0
~
```

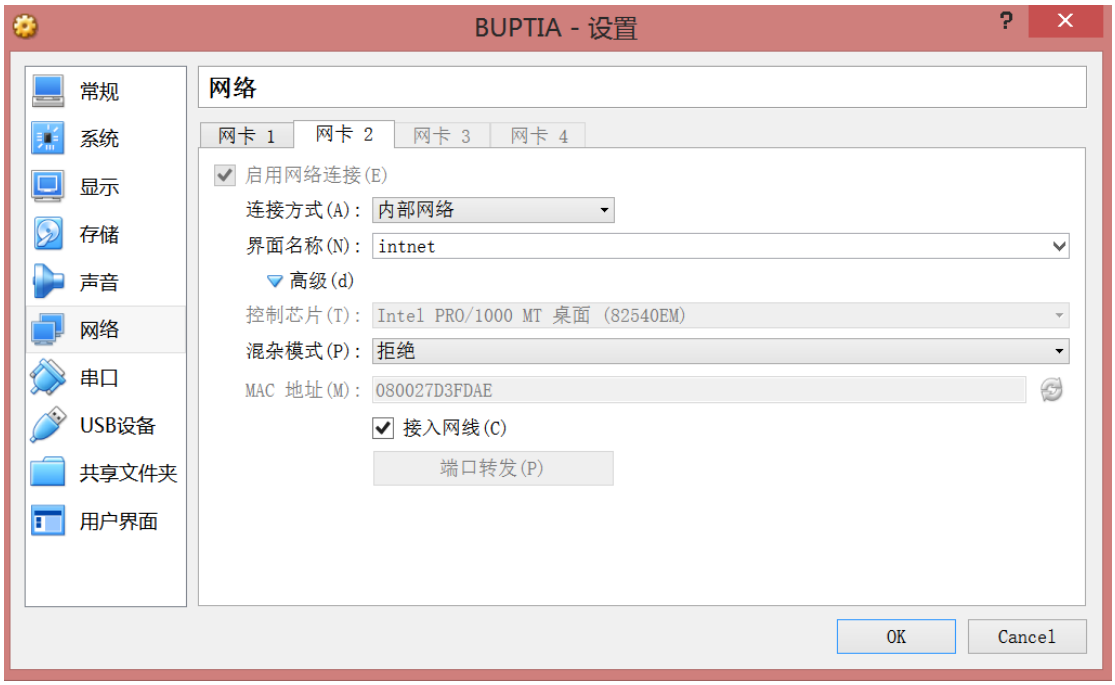
### 2.1.3 DHCP Server

Preparations of DHCP Server are as follows.

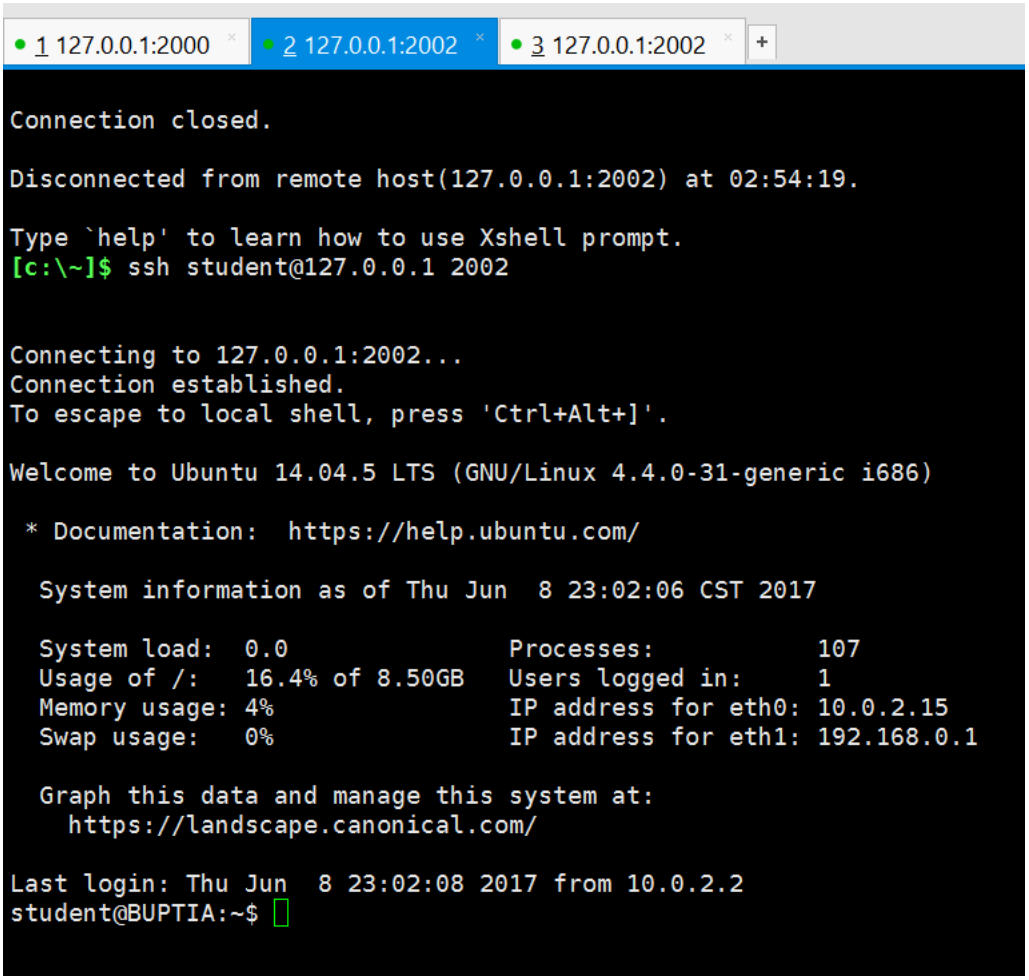
❖ Set NIC1 for DHCP Server.


- Click the button “Port Forwarding” and add a port forwarding rule.
- Then set the Host IP as 127.0.0.1, the Host Port as 2002, the Subsystem IP as 0.0.0.0, and the Subsystem Port as 22.


- Set NIC2 for DHCP Server.




❖ Run the DHCP Server system, open XShell and connect it to 127.0.0.1 port 2002.



❖ Change network configurations, save and restart the system.

```
student@BUPTIA:~$ sudo vim /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

# The host-only interface
auto eth1
iface eth1 inet static
address 192.168.0.1
netmask 255.255.255.0
~
```

❖ Add new iptables rule.

```
student@BUPTIA:~$ sudo iptables -t nat -A POSTROUTING -d 255.255.255.255 -o eth1 -j SNAT --to-source 0.0.0.0
student@BUPTIA:~$ sudo iptables-save > /etc/network/iptables.rules
```

❖ Change “exit 0” to “iptables-restore < /etc/network/iptables.rules
exit 0”

```
student@BUPTIA:~$ sudo vim /etc/rc.local

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
iptables-restore < /etc/network/iptables.rules
exit 0
~
```

### 2.1.4 Environment Testing

After set all of the configurations above, a test need to be carried out to see if the systems work well. The followings are steps of testing.

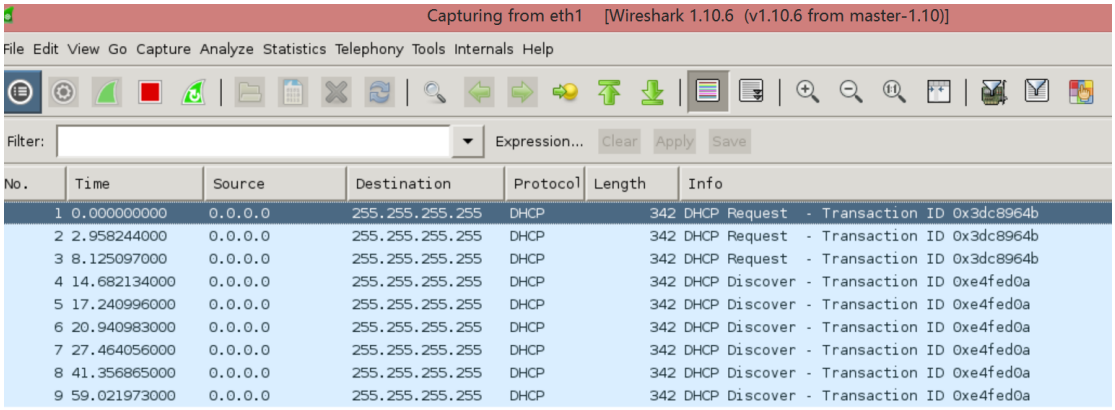
❖ In xShell or Terminal of DHCP-Server VM, set “sudo wireshark”.

```
student@BUPTIA:~$ sudo wireshark
```

❖ In xShell or Terminal of DHCP-Client VM, set “sudo dhclient eth1”.

```
student@BUPTIA:~$ sudo dhclient eth1
```

❖ DHCP DISCOVER displays in Wireshark, which means the environment for this project is established successfully.




## 2.2 Functional Requirement

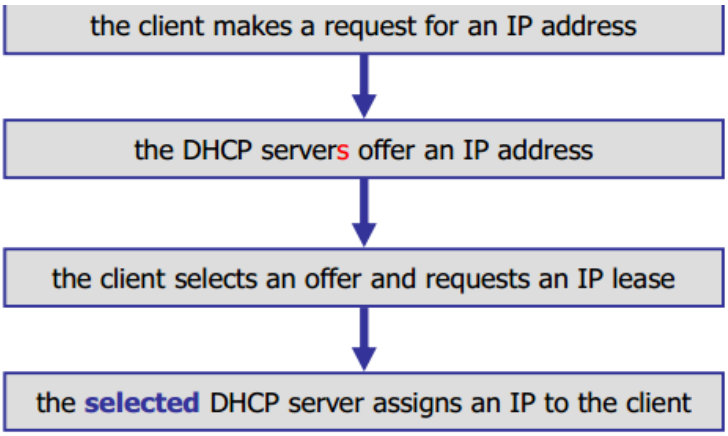
### 2.2.1 DHCP server functions

- Listen to UDP port 67.
- For first request, select free IP address from IP address pool and reply to client.
- For inform request, reply ACK with option value.
- IP range and value of options are stored in IP address pool (dhcp.config file),Assigned IP, client mac address and timestamp are stored in IP lease pool (dhcp.lease file).
- Print log messages.

### 2.2.2 DHCP client functions

- Listen to UDP port 68.
- User can specify command line arguments to control actions of client program. User can combine “sleep x” command with client program to control expire time, such as “./dhcpclient --default; sleep 10; ./dhcpclient --renew”.
- Client program can obtain IP address, netmask, gateway, dns server address, dhcp server ID and IP address lease time from DHCP server and configure IP address in client OS.
- Print log messages.

❖ Address acquisition: Getting an IP address

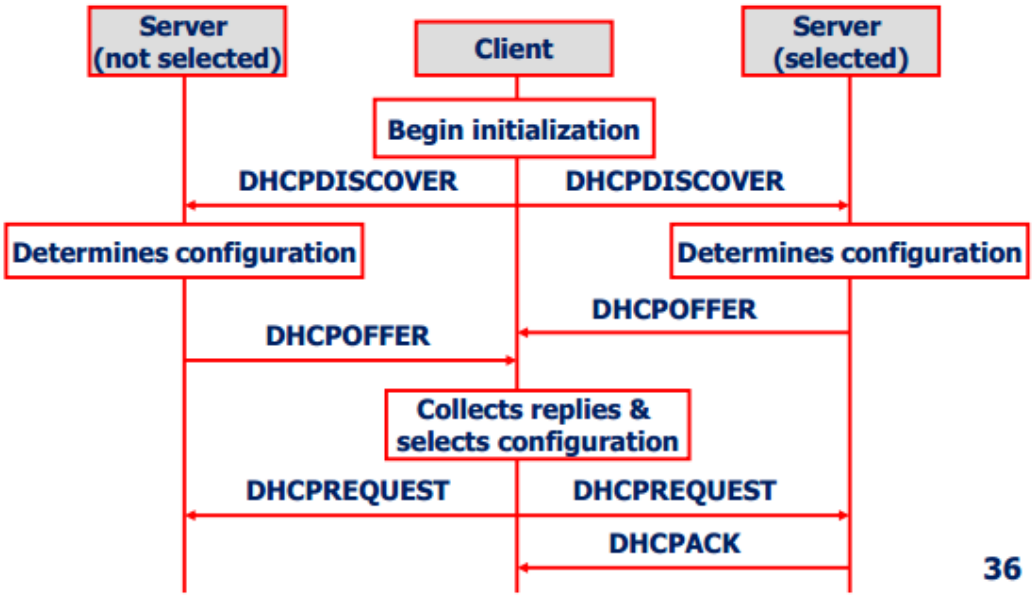



**DHCPDISCOVER** - Client broadcast to locate available servers.

**DHCPOFFER** - Server to client in response to DHCPDISCOVER with offer of configuration parameters.

**DHCPREQUEST** - Client message to servers either (a) requesting offered parameters from one server implicitly declining offers from all others, (b) confirming correctness of previously allocated address after, e.g., system reboot, or (c) extending the lease on a particular network address.

**DHCPACK** - Server to client with configuration parameters, including committed network address.33

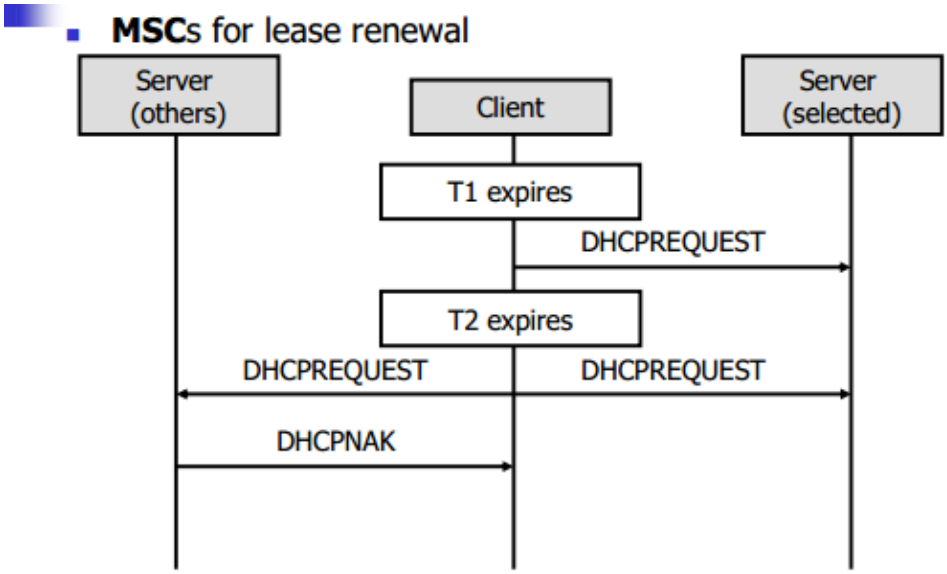



❖ Lease renewal: Returning IP address before lease expires

**Client: DHCPREQUEST-->Server: DHCPACK**

**if client renewal a wrong IP address then：**

**Server: DHCPNAK** - Server to client indicating client's notion of network address is incorrect (e.g., client has moved to new subnet) or client's lease as expired.




❖ Release the IP address

**DHCPRELEASE** - Client to server relinquishing network address and cancelling remaining lease.

❖ Inform the server client's IP address

**DHCPINFORM** - Client to server, asking only for local configuration parameters; client already has externally configured network address. For example, it can be used to obtain tunnel endpoint address.

### 2.2.3 Other Requirements

- Detailed design document and user manual.
- Detailed annotation of code and nice programming style.
- Two persons as a group.

# 3. Preliminary Design

## 3.1 Main Modules

### 3.1.1 Terminal of Server

❖ Judge the type of the packet received and reply for Discover packet.

```
if(recvpacket.MesgType[2] == 0x01)
{
/*DHCP_Offer*/
	printf("DHCP_offer part:----------\n");
	memset(&offer,0,sizeof(offer));
	memcpy(&offer.packet,Buffer,240);
	memset(Buffer,0,sizeof(Buffer));
	your_addr.s_addr = inet_addr(ip[choose].ipaddr);
	printf("choose is: %d\n",choose);
	printf("get ip is: %s\n",ip[choose].ipaddr);
	offer.packet.op = 0x02;	//op 01 request 02reply
	memcpy(offer.packet.your_addr,&your_addr.s_addr,4);
	printf("the address is wrong? %s\n",inet_ntoa(server_addr));
	memset(&request_renewal,0,sizeof(request_renewal));
	strcpy(MsegType,"350102");
	HexStrToByte(MsegType,offer.MsegType,6);
	HexStrToByte(server_identifier,offer.server_identifier,12);
	HexStrToByte(subnetmask,offer.subnetmask,12);
	HexStrToByte(router,offer.router,12);
	HexStrToByte(domain_name,offer.domain_name,12);
	offer.End = 0xff;				//11111111
	memcpy(Buffer,&offer,sizeof(offer));
	dhcpClntAddr.sin_addr.s_addr = inet_addr("255.255.255.255");
/*Send broadcast packets*/
	if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)& dhcpClntAddr, sizeof
(dhcpClntAddr))) != recvMsgSize)
	{
		printf("sendto() sent a different number of bytes than expected.\n");
	}
}
```

- Judge the type of the packet received and reply ACK for Request packet.

```
memcpy(&ack.packet,Buffer,240);
memset(Buffer,0,sizeof(Buffer));
ack.packet.op = 0x02;
memcpy(ack.packet.your_addr,&request.requested_IP[2],4);
strcpy(MsegType,"350105");
HexStrToByte(MsegType,ack.MsegType,6);
memcpy(ack.lease_time,lease_time,6);
HexStrToByte(server_identifier,ack.server_identifier,12);
HexStrToByte(subnetmask,ack.subnetmask,12);
HexStrToByte(router,ack.router,12);
HexStrToByte(domain_name,ack.domain_name,12);
ack.End = 0xff;
anack = 0;
ip[choose].available = 1;
```

❖ Judge the type of the packet received and reply ACK for Inform packet.

```
if(recvpacket.MesgType[2] == 0x08)
{
/*inform*/
memset(&inform,0,sizeof(inform));
memset(&inform_ack,0,sizeof(inform_ack));
memcpy(&inform,Buffer,sizeof(inform));
memcpy(&inform_ack.packet,Buffer,240);
memset(Buffer,0,sizeof(Buffer));
inform_ack.packet.op = 0x02;
memcpy(inform_ack.packet.your_addr,inform.packet.clint_addr,4);
strcpy(MsegType,"350105");
HexStrToByte(MsegType,inform_ack.MsegType,6);
HexStrToByte(server_identifier,inform_ack.server_identifier,12);
HexStrToByte(subnetmask,inform_ack.subnetmask,12);
HexStrToByte(router,inform_ack.router,12);
inform_ack.End = 0xff;
memcpy(Buffer,&inform_ack,sizeof(inform_ack));
dhcpClntAddr.sin_addr.s_addr = inet_addr(inet_ntoa(* (struct in_addr
*)(inform.packet.clint_addr)));
printf("is is is: %s\n",inet_ntoa(* (struct in_addr *)(inform.packet.clint_addr)));
if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)&dhcpClntAddr,sizeof
(dhcpClntAddr ))) != recvMsgSize)
{
   printf("sendto() sent a different number of bytes than expected.\n");
}
}
```

❖ ACK for automatically renew

```
memcpy(&ack.packet,&request.packet,240);
ack.packet.op = 0x02;
memcpy(ack.packet.your_addr,request.packet.clint_addr,4);
strcpy(MsegType,"350105");
HexStrToByte(MsegType,ack.MsegType,6);
memcpy(ack.lease_time,lease_time,6);
HexStrToByte(server_identifier,ack.server_identifier,12);
HexStrToByte(subnetmask,ack.subnetmask,12);
HexStrToByte(router,ack.router,12);
HexStrToByte(domain_name,ack.domain_name,12);
ack.End = 0xff;
anack = 0;
dhcpClntAddr.sin_addr.s_addr = inet_addr(inet_ntoa(* (struct in_addr
*)( request.packet.clint_addr )));
```

❖ NAK

```
memcpy(&nak.packet,&request.packet,240);
nak.packet.op = 0x02;
client_addr.s_addr = inet_addr("0.0.0.0");
your_addr.s_addr = inet_addr("0.0.0.0");
memcpy(nak.packet.clint_addr,&client_addr.s_addr,4);
memcpy(nak.packet.your_addr,&your_addr.s_addr,4);
memcpy(nak.packet.server_addr,&server_addr.s_addr,4);
strcpy(MsegType,"350106");
HexStrToByte(MsegType,nak.MsegType,6);
HexStrToByte(server_identifier,nak.server_identifier,12);
nak.End = 0xff;
anack = 1;
dhcpClntAddr.sin_addr.s_addr = inet_addr(inet_ntoa(* (struct in_addr
*)offer.packet.your_addr )));
```

❖ Response for Release packet.

```
memset(&release,0,sizeof(release));
memcpy(&release,Buffer,sizeof(release));
ip[choose-1].available = 0;
printf("which IP should release? %d",choose);
printf("IP is released, which is: %s\n",inet_ntoa(* (struct in_addr *)(release.packet.
clint_addr)));
```

❖ Record IP address in a file.

```
int fd2;
char filewriter[45];
memset(iplease,0,sizeof(iplease));
memset(filewriter,0,sizeof(filewriter));
if((fd2=open("dhcp.lease",O_CREAT|O_RDWR|O_APPEND))==-1)
printf("Error in opening");
memcpy(iplease[0].ipaddr,inet_ntoa(* (struct in_addr *)(ack.packet.your_addr)),13);
iplease[0].ipaddr[13] = '\0';
ByteToHexStr(ack.packet.client_hwaddr,iplease[0].clientmac,6);
printf("iplease.clientmac is :%s\n",iplease[0].clientmac);
HexByteToInt(&ack.lease_time[2],&iplease[0].lease_time,4);
printf("iplease.lease_time is :%d\n",iplease[0].lease_time);
sprintf(filewriter,"%s %s %d\n",iplease[0].ipaddr,iplease[0].clientmac,iplease[0].lease_
time);
int strlength = strlen(filewriter);
if(write(fd2,filewriter,strlength)!=strlength)
{
      printf("filewriter write error\n");
}
close(fd2);
```

### 3.1.2 Terminal of Client

❖ Send DHCP Discover packet

```
memset(&discover,0,sizeof(discover));
setCommonHeader(&discover.packet);            //240 header
strcpy(MsegType,"350101");
HexStrToByte(MsegType,discover.MsegType,6);
HexStrToByte(client_identifier,discover.client_identifier,18);
HexStrToByte(class_identifier,discover.class_identifier,20);
HexStrToByte(parameter_list,discover.parameter_list,30);
discover.End = 0xff;
memcpy(Buffer,&discover,sizeof(discover));
printf("Buffer is: %s\n",Buffer);
dhcpServAddr.sin_addr.s_addr = inet_addr("255.255.255.255");
if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)& dhcpServAddr,sizeof
(dhcpServAddr))) != recvMsgSize)
{printf("sendto() sent a different number of bytes than expected.\n");}
printf("Send successfully\n");
```

❖ Receive DHCP Offer packet

```
if ((recvMsgSize = recvfrom(sock, Buffer, MAX,0,(struct sockaddr *)&dhcpServAddr, &serAddrLen)) < 0)
printf("recvfrom() failed.\n");
```

❖ Send DHCP Request packet

```
if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)& dhcpServAddr,sizeof (dhcpServAddr))) != recvMsgSize)
{
      printf("sendto() sent a different number of bytes than expected.\n");
}
```

❖ Receive ACK packet

```
if ((recvMsgSize = recvfrom(sock, Buffer, MAX,0,(struct sockaddr *)&dhcpServAddr, &serAddrLen)) < 0)
{
printf("recvfrom() failed.\n");
}
memcpy(&ack,Buffer,sizeof(ack));
```

❖ Send Release packet

```
memset(&release,0,sizeof(release));
setCommonHeader(&release.packet);
memset(Buffer,0,sizeof(Buffer));
client_addr.s_addr = inet_addr(ClntIP);
memcpy(release.packet.clint_addr,&client_addr.s_addr,4);
strcpy(MsegType,"350107");
HexStrToByte(MsegType,release.MsegType,6);
HexStrToByte(server_identifier,release.server_identifier,12);
release.End = 0xff;
memcpy(Buffer,&release,sizeof(release));
dhcpServAddr.sin_addr.s_addr = inet_addr(ServIP);
if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)&dhcpServAddr, sizeof(dhcpServAddr))) != recvMsgSize)
{printf("sendto() sent a different number of bytes than expected.\n");}
setLocalIpAddr("0.0.0.0");
}
```

❖ Send Renew packet

```
strcpy(MsegType,"350103");
HexStrToByte(MsegType,request_renewal.MsegType,6);
HexStrToByte(parameter_list,request_renewal.parameter_list,30);
request_renewal.End = 0xff;
memcpy(Buffer,&request_renewal,sizeof(request_renewal));
dhcpServAddr.sin_addr.s_addr = inet_addr(ServIP);
if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)&dhcpServAddr,
sizeof(dhcpServAddr))) != recvMsgSize)
{printf("sendto() sent a different number of bytes than expected.\n");}
```

❖ Send Inform packet

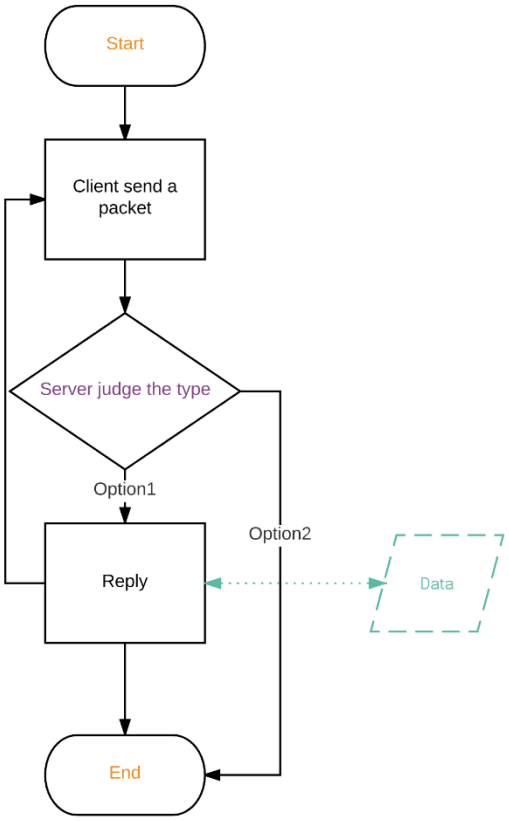
```
strcpy(MsegType,"350108");
HexStrToByte(MsegType,inform.MsegType,6);
HexStrToByte(parameter_list,inform.parameter_list,30);
inform.End = 0xff;
memcpy(Buffer,&inform,sizeof(inform));
printf("The ack server ip address is wrong? %s\n",inet_ntoa(* (struct in_addr
*)(ack.packet.server_addr)));
dhcpServAddr.sin_addr.s_addr = inet_addr(ServIP);
if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)&dhcpServAddr,
sizeof(dhcpServAddr))) != recvMsgSize)
{printf("sendto() sent a different number of bytes than expected.\n");}
```

❖ Receive NAK packet

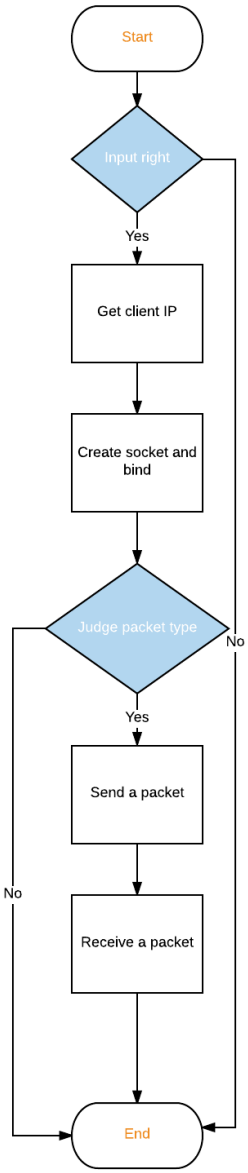
```
memset(&nak,0,sizeof(nak));
memset(Buffer,0,sizeof(Buffer));
if ((recvMsgSize = recvfrom(sock, Buffer, MAX,0,(struct sockaddr *)&dhcpServAddr,
&serAddrLen)) < 0)
{
printf("recvfrom() failed.\n");
}
setLocalIpAddr("0.0.0.0");
memcpy(&nak,Buffer,sizeof(nak));
ByteToHexStr(nak.MsegType,MsegType,3);
printf("Message type is: %s\n",MsegType);
argv[1] = "--default";
```

## 3.2 Relationship between Modules

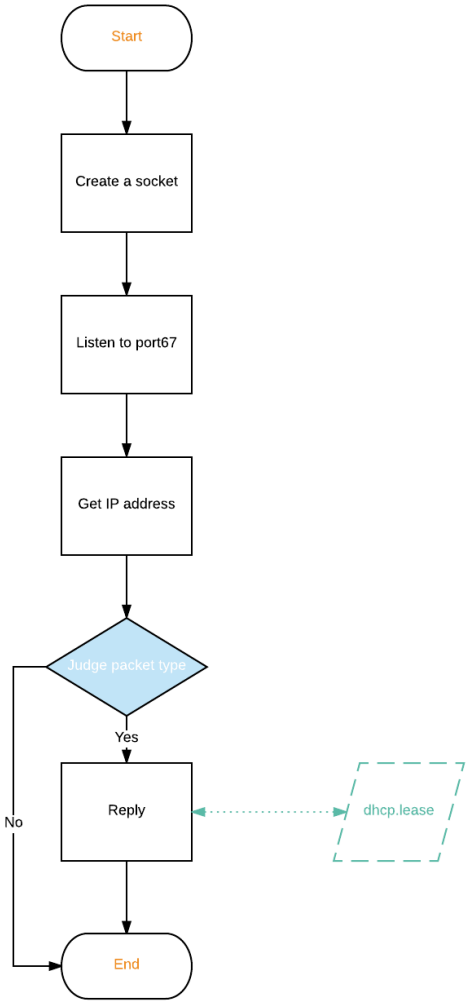
### 3.2.1 General structure



### 3.2.2 Client Send and Receive



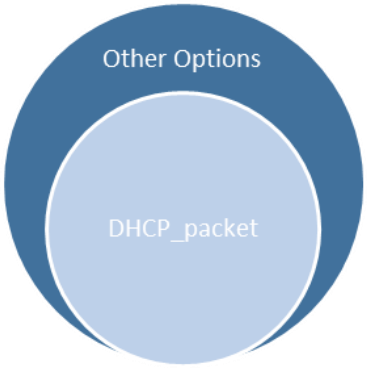
### 3.2.3 Server Reply




# 4. Detailed Design

## 4.1 Main Structures

Main structures are designed as follows, since the first 240 bits of dhcp packets are the same, we define a DHCP_packet structure to store the first 240 bits. And other dhcp packets are specialized by distinctive options.




### 4.1.1 Build fundamental packet of DHCP with parameters:

```
struct DHCP_packet {
    char op;
    char htype;
    char hlen;
    char hops;
```

Which satisfy the DHCP message format:

| OP (1) | HTYPE (1) | HLEN (1) | HOPS (1) |
|---|---|---|---|
| TRANSACTION ID (4) | | | |
| SECONDS (2) | | FLAGS (2) | |
| CLIENT IP ADDRESS (4) | | | |
| YOUR IP ADDRESS (4) | | | |
| SERVER IP ADDRESS (4) | | | |
| ROUTER IP ADDRESS (4) | | | |
| CLIENT HARDWARE ADDRESS (16) ⋮ | | | |
| SERVER HOST NAME (64) ⋮ | | | |
| BOOT FILE NAME (128) ⋮ | | | |
| OPTIONS (variable) ⋮ | | | |

### 4.1.2 Then build the struct of address acquisition process packet with their specific option:

```
struct DHCP_discover
{
    struct DHCP_packet packet;
    char MsegType[3];
```

```
struct DHCP_offer
{
    struct DHCP_packet packet;
    char MsegType[3];

   struct DHCP_request
   {
       struct DHCP_packet packet;
       char MsegType[3];

  struct DHCP_nak
  {
      struct DHCP_packet packet;
      char MsegType[3];
```

❖ And when input ./client --addraq: then the system will judge the order and send discover packet:

```
dhcpServAddr.sin_addr.s_addr = inet_addr("255.255.255.255");
if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)&dhcpServAddr,sizeof(dhcpServAddr))) != recvMsgSize)
```

❖ Then get the offer packet from server, then inverse resolution to get the IP from server,then client sent the packet of request to server to insure the IP address is used:

```
dhcpServAddr.sin_addr.s_addr = inet_addr("255.255.255.255");
if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)&dhcpServAddr,sizeof(dhcpServAddr))) != recvMsgSize)
```

❖ Then sever will replay the ack.

### 4.1.3 The structure of request_renewal:

```
struct DHCP_request_renewal
{
    struct DHCP_packet packet;
    char MsegType[3];
```

❖ When input ./client --sleep:

The client will sent the DHCP_request packet at 1/2 lease time and sever will replay an ack.
But if client renew a wrong IP address:

{
If the IP address is not equal to the address that the server assigned
then the server will replay a nack.
}

❖ Then client receive the nack and re-transmit the process of address acquisition.Repeat the process of ‘1’.

### 4.1.4 The structure of inform:

```
struct DHCP_inform
{
    struct DHCP_packet packet;
    char MsegType[3];
```

For client sent a packet to inform the address to server.

### 4.1.5 The struct of release:

```
struct DHCP_release
{
    struct DHCP_packet packet;
    char MsegType[3];
```

- When input ./client --release, the client will release the IP address that it get from the server then let server to know that the address has been released.

```
    dhcpServAddr.sin_addr.s_addr = inet_addr(serv_add);
if((sendto(sock,Buffer,recvMsgSize,0,(struct sockaddr *)&dhcpServAddr,sizeof(dhcpServAddr))) != recvMsgSize)
```

### 4.1.6 Address format exchange between hexadecimal to byte and hexadecimal to int.

### 4.1.7 Get client local IP address:

```
strcpy(ipAddr,inet_ntoa(( (struct sockaddr_in *)&ifr.ifr_addr )->sin_addr))
```

## 4.2 Other Structures

### 4.2.1 Socket

A socket can be treated the same as a standard file descriptor except that. It is created with the socket (). Additional calls are needed to connect and activate it. recv() and send() are also used as counterparts to read() and write().

### 4.2.2 Address Formats Conversion

- inet_aton(): IP address in numbers-and-dots notation (ASCII string)-> IP address structure in network byte order.
- inet_addr(): function is same with inet_aton().
- inet_ntoa(): IP address structure in network byte order ->IP address in numbers-and-dots notation (ASCII string).

### 4.2.3 Internet-specific socket address

```
struct sockaddr_in {
   unsigned short sin_family; /* address family (always AF_INET) */
   unsigned short sin_port; /* port num in network byte order */
   struct in_addr sin_addr; /* IP addr in network byte order */
   unsigned char sin_zero[8]; /* pad to sizeof(struct sockaddr) */
};
```

❖ Generic socket address

```
struct sockaddr {
   unsigned short sa_family; /* protocol family */
   char sa_data[14]; /* protocol-specific address,
                        up to 14 bytes. */
};
```

❖ Bind method

```
struct sockaddr_in serv;
/* fill in serv{}*/
bind (sockfd, (struct sockaddr *)&serv , sizeof(serv));
```

### 4.2.4 Broadcast

```
i=1;
struct ifreq if_eth1;
strcpy(if_eth1.ifr_name,"eth1");
socklen_t len = sizeof(i);
/*allow socket to broadcast ?*/
setsockopt(sock,SOL_SOCKET,SO_BROADCAST,&i,len);
/*set socket to interface eth1 */
if(setsockopt(sock,SOL_SOCKET,SO_BINDTODEVICE,(char *)&if_eth1,sizeof(if_eth1))<0)
{
      printf("bind socket to eth1 error\n");
}
```

### 4.2.5 Others

Change data format

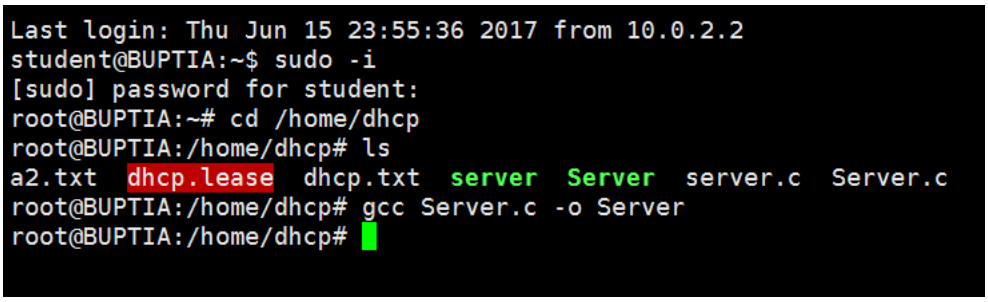
```
void HexByteToInt(const char* source, unsigned int* dest, int sourcelen)
{
    char middle[sourcelen];
    int i;
    for(i=0;i<sourcelen;i++)
    {
        middle[i] = source[sourcelen-1-i];
    }
    memcpy(dest,middle,sourcelen);
}
```

```
void IntToHexByte(unsigned int* source, char* dest, int destlen)
{
    char in[destlen];
    char out[destlen];
    memcpy(in,source,destlen);
    int i;
    for(i=0;i<destlen;i++)
    {
        out[i] = in[destlen-1-i];
    }
    out[destlen] = '\0';
    memcpy(dest,out,destlen);
}
```

# 5. Results

Results of this program are as follows:

- In the terminal of server, get into "root" and enter home/dhcp, where the documents are stored, and compile the source file Server.c.




- In the terminal of client, get into "root" and enter home/dhcp, where the documents are stored,

and compile the source file Client.c.

```
Last login: Thu Jun 15 23:55:07 2017
student@BUPTIA:~$ sudo -i
[sudo] password for student:
root@BUPTIA:~# cd /home/dhcp
root@BUPTIA:/home/dhcp# ls
Client  client.c  Client.c  clinet  clinet.c
root@BUPTIA:/home/dhcp# gcc Client.c -o Client
root@BUPTIA:/home/dhcp#
```

## 5.1 Get IP Address

- In the terminal of server, run the complied file Server, and enter a lease time, which is required as a number.

```
root@BUPTIA:/home/dhcp# ./Server 30
x is: 30bind successfully
i is: 4
get ip is: 192.168.0.110
Receive successfully
```

- In the terminal of client, run the complied file Client, and enter an instruction, which should be in the form "--instruction". Then there are displays on the screen to show the assigned IP address for the client. Message type 350102 means that the packet is a dhcp offer, 350105 means an dhcp ACK packet.

```
root@BUPTIA:/home/dhcp# ./Client --default
socket built successfully
bind failed.
eth1 IP is: 0.0.0.0
Buffer is:
Send successfully
tranaction_id is: 53A703DD
offer.packet.your_addr is: 192.168.0.110
Message type is: 350102
Message type is: 350105
lease_time is: 30
ie1 ip address will be: 192.168.0.110
```

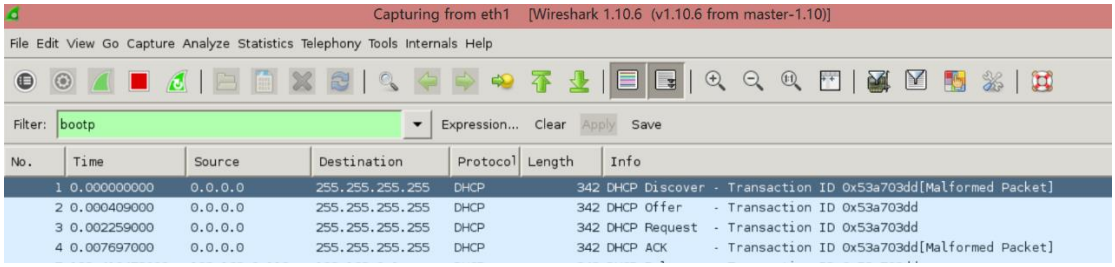
- In the terminal of client, use "ifconfig" instruction to check the client address.

```
root@BUPTIA:/home/dhcp# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:a9:67:a3
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fea9:67a3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:343 errors:0 dropped:0 overruns:0 frame:0
          TX packets:245 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:34074 (34.0 KB)  TX bytes:36354 (36.3 KB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:d3:fd:ae
          inet addr:192.168.0.110  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fed3:fdae/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:684 (684.0 B)  TX bytes:1332 (1.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

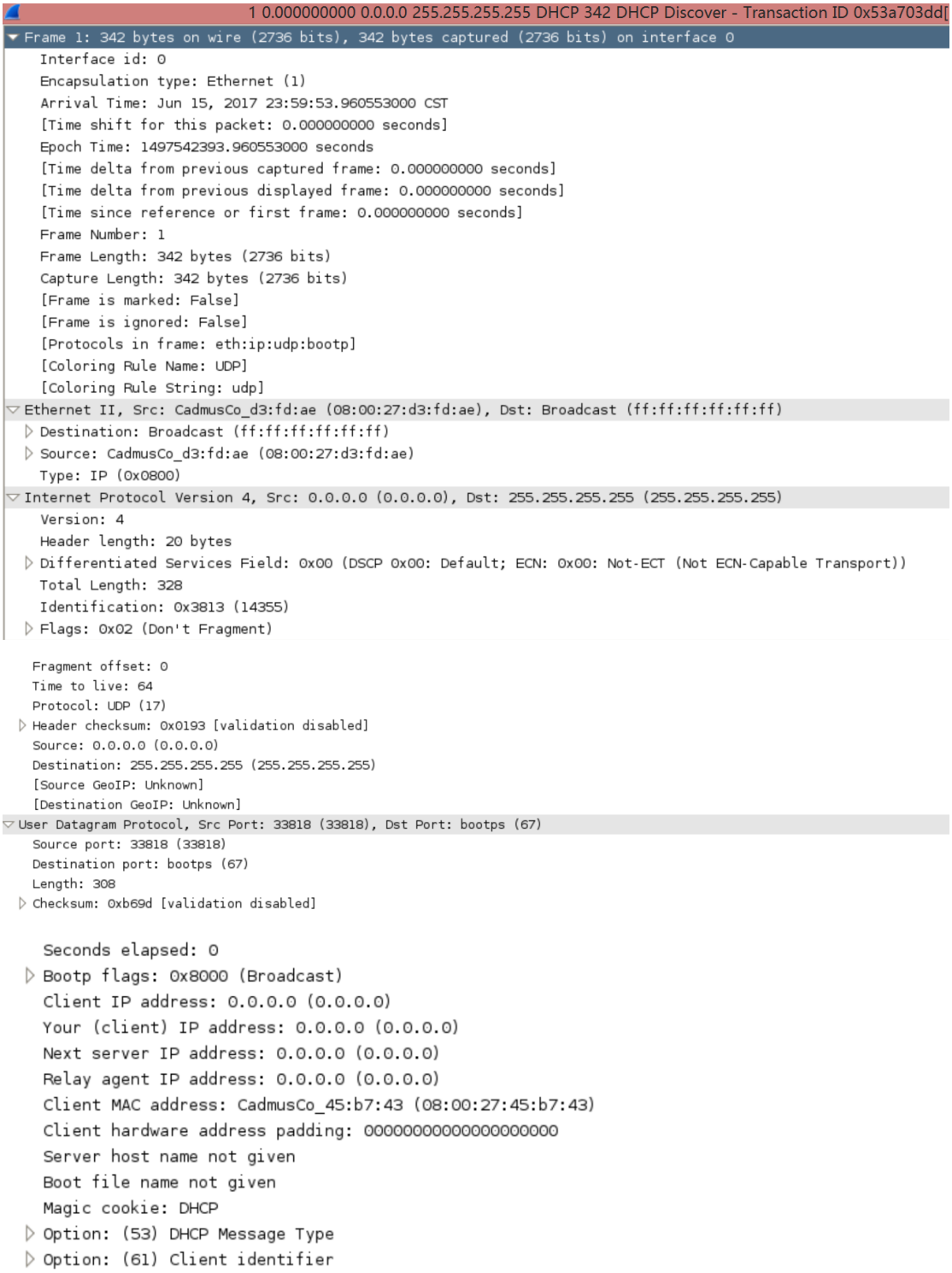
❖ In the terminal of server, open “dhcp.lease” file and there is the information of the assigned IP address.

```
student@BUPTIA:~$ sudo -i
[sudo] password for student:
root@BUPTIA:~# cd /home/dhcp
root@BUPTIA:/home/dhcp# ls
a2.txt  dhcp.lease  dhcp.txt  server  Server  server.c  Server.c
root@BUPTIA:/home/dhcp# vi dhcp.lease

192.168.0.110 08002745B743 30
```

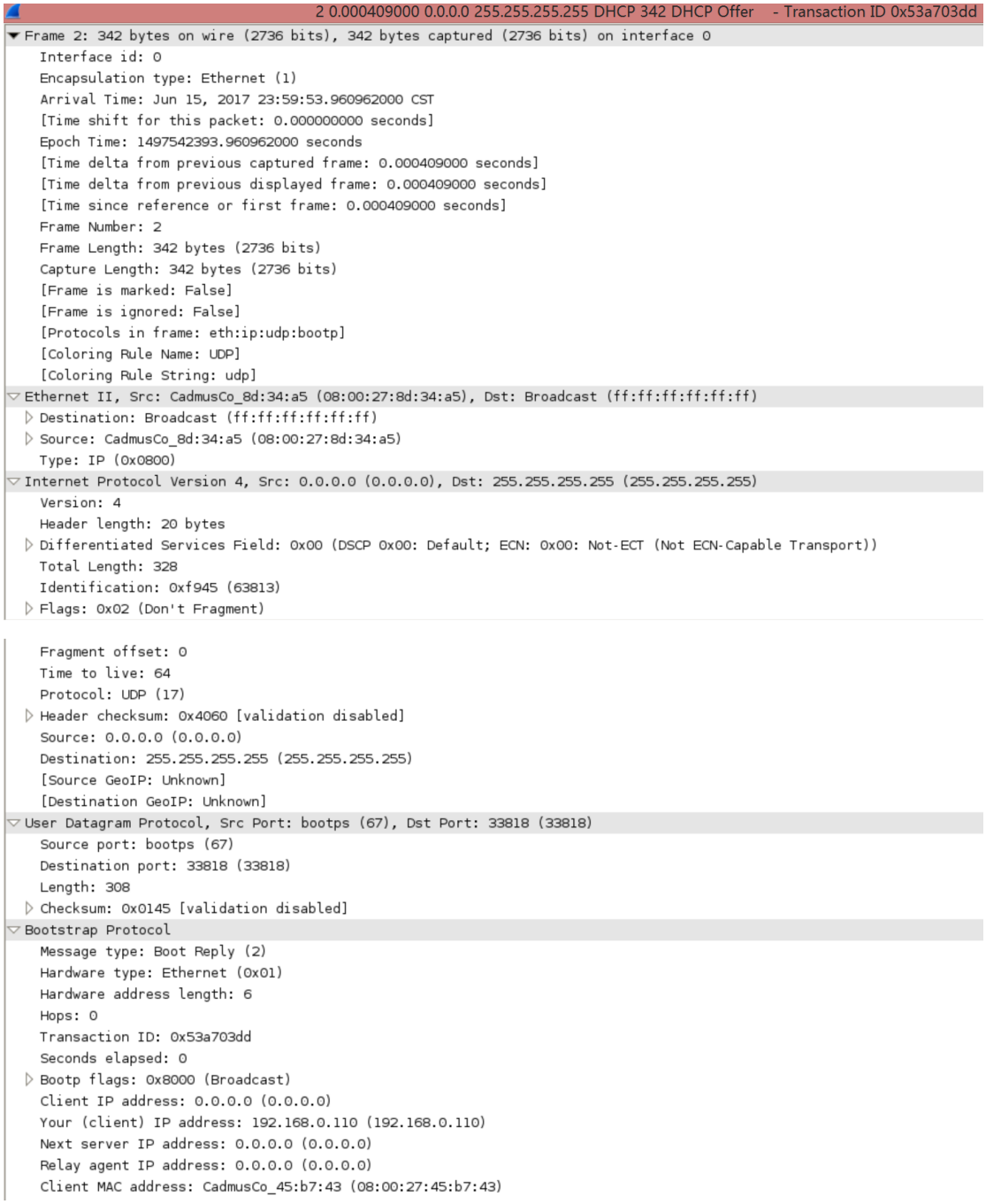
❖ In wireshark, there are four packets captured, which is the common broadcast communication process between the dhcp server and the dhcp client.




- Discover packet




- Offer packet


2 0.000409000 0.0.0.0 255.255.255.255 DHCP 342 DHCP Offer - Transaction ID 0x53a703dd
Frame 2: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
Interface id: 0
Encapsulation type: Ethernet (1)
Arrival Time: Jun 15, 2017 23:59:53.960962000 CST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1497542393.960962000 seconds
[Time delta from previous captured frame: 0.000409000 seconds]
[Time delta from previous displayed frame: 0.000409000 seconds]
[Time since reference or first frame: 0.000409000 seconds]
Frame Number: 2
Frame Length: 342 bytes (2736 bits)
Capture Length: 342 bytes (2736 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ip:udp:bootp]
[Coloring Rule Name: UDP]
[Coloring Rule String: udp]
Ethernet II, Src: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Source: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5)
Type: IP (0x0800)
Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 328
Identification: 0xf945 (63813)
Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 64
Protocol: UDP (17)
Header checksum: 0x4060 [validation disabled]
Source: 0.0.0.0 (0.0.0.0)
Destination: 255.255.255.255 (255.255.255.255)
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
User Datagram Protocol, Src Port: bootps (67), Dst Port: 33818 (33818)
Source port: bootps (67)
Destination port: 33818 (33818)
Length: 308
Checksum: 0x0145 [validation disabled]
Bootstrap Protocol
Message type: Boot Reply (2)
Hardware type: Ethernet (0x01)
Hardware address length: 6
Hops: 0
Transaction ID: 0x53a703dd
Seconds elapsed: 0
Bootp flags: 0x8000 (Broadcast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 192.168.0.110 (192.168.0.110)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: CadmusCo_45:b7:43 (08:00:27:45:b7:43)

```
Client hardware address padding: 00000000000000000000
Server host name not given
Boot file name not given
Magic cookie: DHCP
▽ Option: (53) DHCP Message Type
    Length: 1
    DHCP: Offer (2)
▽ Option: (54) DHCP Server Identifier
    Length: 4
    DHCP Server Identifier: 192.168.0.1 (192.168.0.1)
▽ Option: (1) Subnet Mask
    Length: 4
    Subnet Mask: 255.255.255.0 (255.255.255.0)
▽ Option: (3) Router
    Length: 4
    Router: 192.168.0.1 (192.168.0.1)
▽ Option: (6) Domain Name Server
    Length: 4
    Domain Name Server: 192.168.0.1 (192.168.0.1)
▽ Option: (0) Padding
    Padding: 00000000000000000000000000
▽ Option: (255) End
    Option End: 255
Padding
```

- Request packet

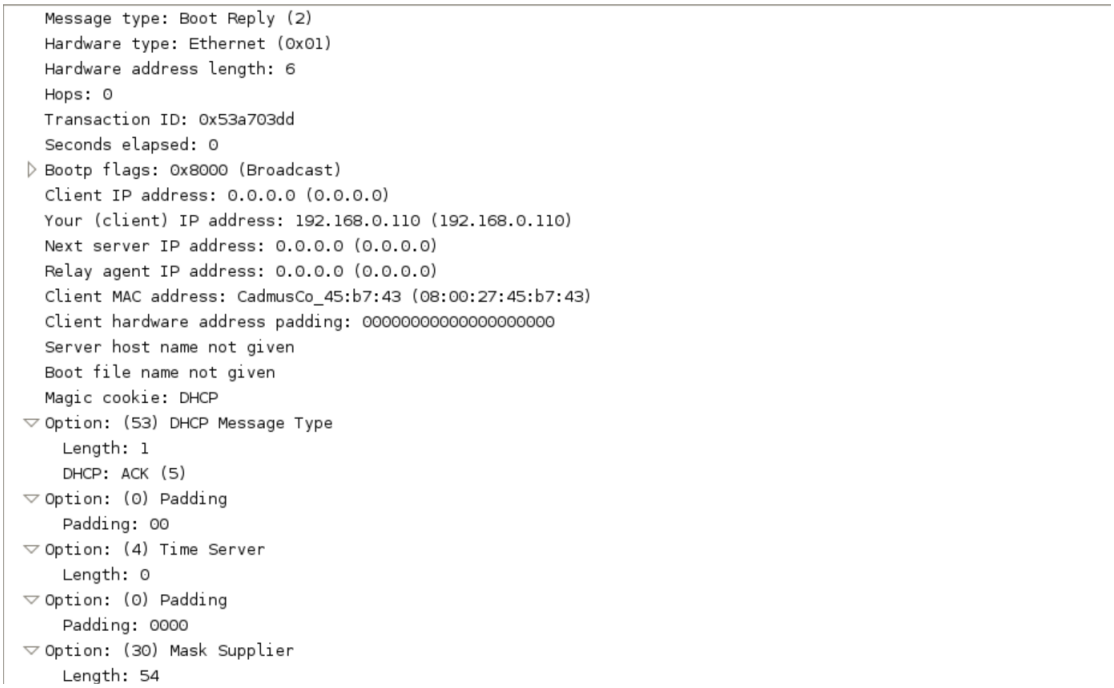
```
3 0.002259000 0.0.0.0 255.255.255.255 DHCP 342 DHCP Request  - Transaction ID 0x53a703dd
▽ Frame 3: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Jun 15, 2017 23:59:53.962812000 CST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1497542393.962812000 seconds
    [Time delta from previous captured frame: 0.001850000 seconds]
    [Time delta from previous displayed frame: 0.001850000 seconds]
    [Time since reference or first frame: 0.002259000 seconds]
    Frame Number: 3
    Frame Length: 342 bytes (2736 bits)
    Capture Length: 342 bytes (2736 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:udp:bootp]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
▽ Ethernet II, Src: CadmusCo_d3:fd:ae (08:00:27:d3:fd:ae), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Source: CadmusCo_d3:fd:ae (08:00:27:d3:fd:ae)
    Type: IP (0x0800)
▽ Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 328
    Identification: 0x3814 (14356)
  ▷ Flags: 0x02 (Don't Fragment)
```

```
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
  ▷ Header checksum: 0x0192 [validation disabled]
    Source: 0.0.0.0 (0.0.0.0)
    Destination: 255.255.255.255 (255.255.255.255)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▽ User Datagram Protocol, Src Port: 33818 (33818), Dst Port: bootps (67)
    Source port: 33818 (33818)
    Destination port: bootps (67)
    Length: 308
  ▷ Checksum: 0x1513 [validation disabled]
▽ Bootstrap Protocol
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x53a703dd
    Seconds elapsed: 0
  ▷ Bootp flags: 0x8000 (Broadcast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: CadmusCo_45:b7:43 (08:00:27:45:b7:43)
    Client hardware address padding: 00000000000000000000
```

```
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ▽ Option: (53) DHCP Message Type
      Length: 1
      DHCP: Request (3)
  ▽ Option: (50) Requested IP Address
      Length: 4
      Requested IP Address: 192.168.0.110 (192.168.0.110)
  ▽ Option: (55) Parameter Request List
      Length: 13
      Parameter Request List Item: (1) Subnet Mask
      Parameter Request List Item: (3) Router
      Parameter Request List Item: (6) Domain Name Server
      Parameter Request List Item: (15) Domain Name
      Parameter Request List Item: (31) Perform Router Discover
      Parameter Request List Item: (33) Static Route
      Parameter Request List Item: (43) Vendor-Specific Information
      Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
      Parameter Request List Item: (46) NetBIOS over TCP/IP Node Type
      Parameter Request List Item: (47) NetBIOS over TCP/IP Scope
      Parameter Request List Item: (121) Classless Static Route
      Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
      Parameter Request List Item: (252) Private/Proxy autodiscovery
  ▽ Option: (255) End
      Option End: 255
    Padding
```

- ACK packet

```
4 0.007697000 0.0.0.0 255.255.255.255 DHCP 342 DHCP ACK      - Transaction ID 0x53a703dd[Malformed Packet]
Frame 4: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Jun 15, 2017 23:59:53.968250000 CST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1497542393.968250000 seconds
    [Time delta from previous captured frame: 0.005438000 seconds]
    [Time delta from previous displayed frame: 0.005438000 seconds]
    [Time since reference or first frame: 0.007697000 seconds]
    Frame Number: 4
    Frame Length: 342 bytes (2736 bits)
    Capture Length: 342 bytes (2736 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:udp:bootp]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Ethernet II, Src: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▷ Source: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 328
    Identification: 0xf947 (63815)
  ▷ Flags: 0x02 (Don't Fragment)

    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
  ▷ Header checksum: 0x405e [validation disabled]
    Source: 0.0.0.0 (0.0.0.0)
    Destination: 255.255.255.255 (255.255.255.255)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
User Datagram Protocol, Src Port: bootps (67), Dst Port: 33818 (33818)
    Source port: bootps (67)
    Destination port: 33818 (33818)
    Length: 308
  ▷ Checksum: 0x0145 [validation disabled]
Bootstrap Protocol
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x53a703dd
    Seconds elapsed: 0
  ▷ Bootp flags: 0x8000 (Broadcast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 192.168.0.110 (192.168.0.110)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: CadmusCo_45:b7:43 (08:00:27:45:b7:43)
    Client hardware address padding: 00000000000000000000
```




## 5.2 Release IP Address

- In the terminal of client, run the complied file Client, and enter an instruction "--release". Use "ifconfig" instruction to check that there is no IP address for client.

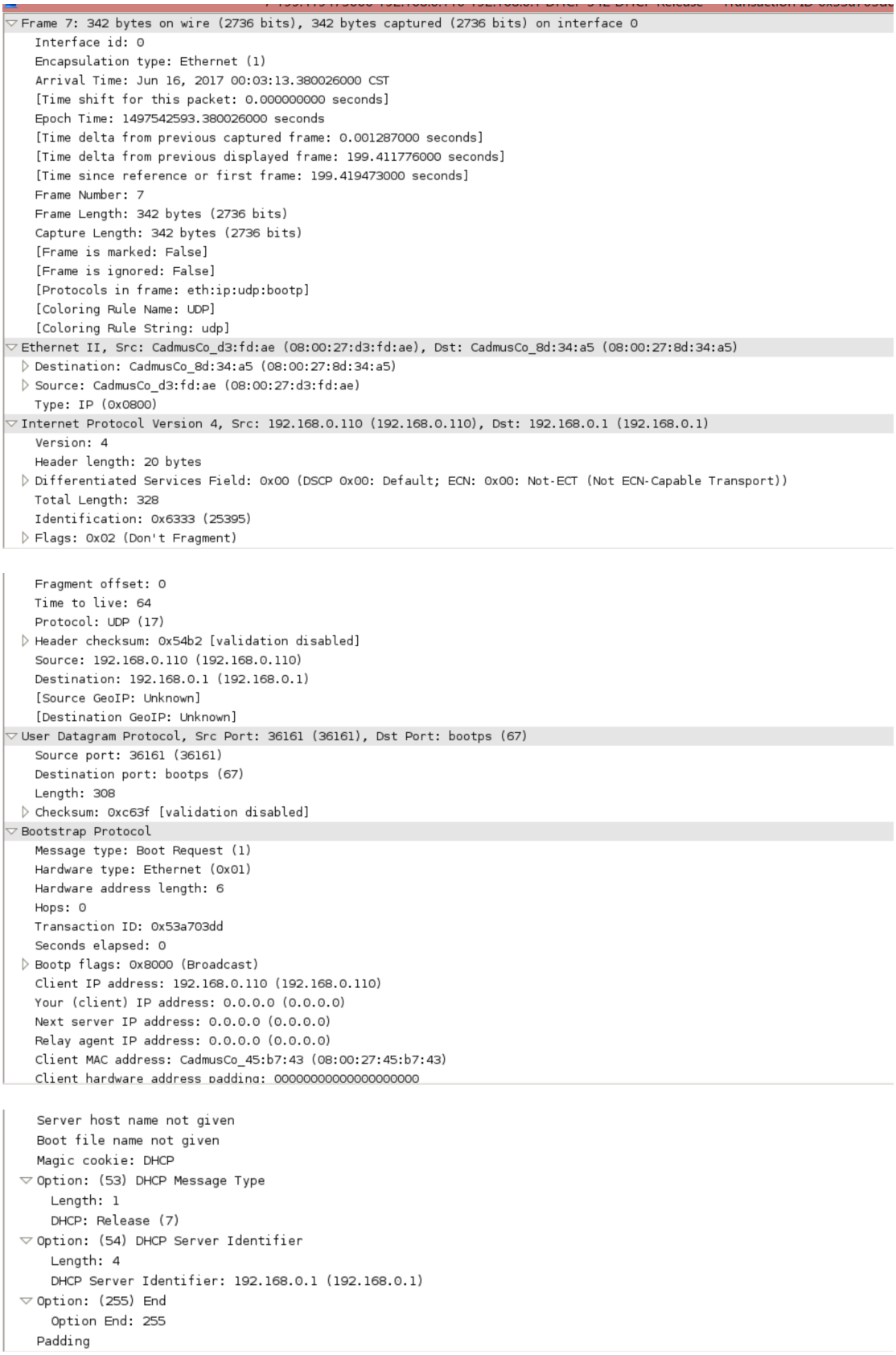
```
root@BUPTIA:/home/dhcp# ./Client --release
socket built successfully
bind failed.
root@BUPTIA:/home/dhcp# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:a9:67:a3
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fea9:67a3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:510 errors:0 dropped:0 overruns:0 frame:0
          TX packets:338 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:48374 (48.3 KB)  TX bytes:48554 (48.5 KB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:d3:fd:ae
          inet6 addr: fe80::a00:27ff:fed3:fdae/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:744 (744.0 B)  TX bytes:1734 (1.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- In the terminal of server, open “dhcp.lease” file and there is no information of the assigned IP address.

```
root@BUPTIA:/home/dhcp# vi dhcp.lease
```

- In wireshark, there is one packet captured, which is a release packet. A unicast communication is achieved.

| | | | | | |
|---|---|---|---|---|---|
| 7 | 199.419473000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 DHCP Release - Transaction ID 0x53a703dd |



```
▽ Frame 7: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Jun 16, 2017 00:03:13.380026000 CST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1497542593.380026000 seconds
    [Time delta from previous captured frame: 0.001287000 seconds]
    [Time delta from previous displayed frame: 199.411776000 seconds]
    [Time since reference or first frame: 199.419473000 seconds]
    Frame Number: 7
    Frame Length: 342 bytes (2736 bits)
    Capture Length: 342 bytes (2736 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:udp:bootp]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
▽ Ethernet II, Src: CadmusCo_d3:fd:ae (08:00:27:d3:fd:ae), Dst: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5)
  ▷ Destination: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5)
  ▷ Source: CadmusCo_d3:fd:ae (08:00:27:d3:fd:ae)
    Type: IP (0x0800)
▽ Internet Protocol Version 4, Src: 192.168.0.110 (192.168.0.110), Dst: 192.168.0.1 (192.168.0.1)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 328
    Identification: 0x6333 (25395)
  ▷ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
  ▷ Header checksum: 0x54b2 [validation disabled]
    Source: 192.168.0.110 (192.168.0.110)
    Destination: 192.168.0.1 (192.168.0.1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▽ User Datagram Protocol, Src Port: 36161 (36161), Dst Port: bootps (67)
    Source port: 36161 (36161)
    Destination port: bootps (67)
    Length: 308
  ▷ Checksum: 0xc63f [validation disabled]
▽ Bootstrap Protocol
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x53a703dd
    Seconds elapsed: 0
  ▷ Bootp flags: 0x8000 (Broadcast)
    Client IP address: 192.168.0.110 (192.168.0.110)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: CadmusCo_45:b7:43 (08:00:27:45:b7:43)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ▽ Option: (53) DHCP Message Type
      Length: 1
      DHCP: Release (7)
  ▽ Option: (54) DHCP Server Identifier
      Length: 4
      DHCP Server Identifier: 192.168.0.1 (192.168.0.1)
  ▽ Option: (255) End
      Option End: 255
    Padding
```

## 5.3 Automatically Renew

- In the terminal of client, run the complied file Client, and enter an instruction "—default sleep" to automatically run the renewal program. The sleep time is set as 50% of the lease time entered in the server terminal which is 30s.

```
root@BUPTIA:/home/dhcp# ./Client --default sleep
socket built successfully
bind failed.
eth1 IP is: 0.0.0.0
Buffer is:
Send successfully
tranaction_id is: 53A703DD
offer.packet.your_addr is: 192.168.0.110
Message type is: 350102
Message type is: 350105
lease_time is: 30
ie1 ip address will be: 192.168.0.110
Message type is: 350105
lease_time is: 30
Message type is: 350105
lease_time is: 30
Message type is: 350105
lease_time is: 30
Message type is: 350105
lease_time is: 30
```

- In the wireshark, there are request and ACK pairs which are captured every 15 seconds.

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | 244.366983000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Discover - Transaction ID 0x53a703dd[Malformed Packet] |
| 9 | 244.367376000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Offer - Transaction ID 0x53a703dd |
| 10 | 244.368997000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 11 | 244.369347000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 14 | 259.387980000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 15 | 259.388301000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 18 | 274.392279000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 19 | 274.392499000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 20 | 289.394326000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 21 | 289.394576000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 24 | 304.397188000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 25 | 304.397407000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |

## 5.4 Incorrect Renewal

- In the terminal of client, kill the Client process to let it run again.

```
root@BUPTIA:/home/dhcp# sudo lsof -i
COMMAND   PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
dhclient  592    root    6u  IPv4   9814      0t0  UDP *:bootpc
dhclient  592    root   20u  IPv4   9565      0t0  UDP *:35443
dhclient  592    root   21u  IPv6   9566      0t0  UDP *:54852
sshd      973    root    3u  IPv4  10441      0t0  TCP *:ssh (LISTEN)
sshd      973    root    4u  IPv6  10443      0t0  TCP *:ssh (LISTEN)
sshd     1262    root    3u  IPv4  11098      0t0  TCP 10.0.2.15:ssh->10.0.2.2:49529 (ESTABLISHED)
sshd     1310 student    3u  IPv4  11098      0t0  TCP 10.0.2.15:ssh->10.0.2.2:49529 (ESTABLISHED)
sshd     1310 student    9u  IPv6  11291      0t0  TCP localhost:6010 (LISTEN)
sshd     1310 student   10u  IPv4  11292      0t0  TCP localhost:6010 (LISTEN)
sshd     1327    root    3u  IPv4  11321      0t0  TCP 10.0.2.15:ssh->10.0.2.2:49532 (ESTABLISHED)
sshd     1375 student    3u  IPv4  11321      0t0  TCP 10.0.2.15:ssh->10.0.2.2:49532 (ESTABLISHED)
sshd     1375 student    9u  IPv6  11471      0t0  TCP localhost:6011 (LISTEN)
sshd     1375 student   10u  IPv4  11472      0t0  TCP localhost:6011 (LISTEN)
Client   1421    root    3u  IPv4  11573      0t0  UDP *:43353
root@BUPTIA:/home/dhcp# sudo kill -s 9 1421
[1]+  Killed                  ./Client --default sleep
root@BUPTIA:/home/dhcp#
```

❖ In the terminal of client, run the client program by enter the instruction "—renewal wrong IP", where here is 127.0.0.1.

```
root@BUPTIA:/home/dhcp# ./Client --renewal 127.0.0.1
socket built successfully
bind failed.
Message type is: 350106
eth1 IP is: 192.168.0.110
Buffer is:
Send successfully
tranaction_id is: 53A703DD
offer.packet.your_addr is: 192.168.0.111
Message type is: 350102
Message type is: 350105
lease_time is: 30
ie1 ip address will be: 192.168.0.111
```

❖ In wireshark, there are request packet and NAK. The process of asking IP address is failed, and the program automatically broadcast to find a new IP address.

- NAK packet

```
28 416.127894000   192.168.0.110   192.168.0.1       DHCP          342 DHCP Request  - Transaction ID 0x53a703dd
29 416.128250000   192.168.0.1     192.168.0.110     DHCP          342 DHCP NAK      - Transaction ID 0x53a703dd
30 416.140550000   0.0.0.0         255.255.255.255   DHCP          342 DHCP Discover - Transaction ID 0x53a703dd[Malformed Packet]
31 416.140937000   0.0.0.0         255.255.255.255   DHCP          342 DHCP Offer    - Transaction ID 0x53a703dd
32 416.144064000   0.0.0.0         255.255.255.255   DHCP          342 DHCP Request  - Transaction ID 0x53a703dd
33 416.144500000   0.0.0.0         255.255.255.255   DHCP          342 DHCP ACK      - Transaction ID 0x53a703dd[Malformed Packet]
```

```
29 416.128250000 192.168.0.1 192.168.0.110 DHCP 342 DHCP NAK      - Transaction ID 0x53a703dd
Frame 29: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Jun 16, 2017 00:06:50.088803000 CST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1497542810.088803000 seconds
    [Time delta from previous captured frame: 0.000356000 seconds]
    [Time delta from previous displayed frame: 0.000356000 seconds]
    [Time since reference or first frame: 416.128250000 seconds]
    Frame Number: 29
    Frame Length: 342 bytes (2736 bits)
    Capture Length: 342 bytes (2736 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:udp:bootp]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Ethernet II, Src: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5), Dst: CadmusCo_d3:fd:ae (08:00:27:d3:fd:ae)
  Destination: CadmusCo_d3:fd:ae (08:00:27:d3:fd:ae)
  Source: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.110 (192.168.0.110)
    Version: 4
    Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 328
    Identification: 0x87fd (34813)
  Flags: 0x02 (Don't Fragment)

    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
  Header checksum: 0x2fe8 [validation disabled]
    Source: 192.168.0.1 (192.168.0.1)
    Destination: 192.168.0.110 (192.168.0.110)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
User Datagram Protocol, Src Port: bootps (67), Dst Port: 42069 (42069)
    Source port: bootps (67)
    Destination port: 42069 (42069)
    Length: 308
  Checksum: 0x8305 [validation disabled]
Bootstrap Protocol
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x53a703dd
    Seconds elapsed: 0
  Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: CadmusCo_45:b7:43 (08:00:27:45:b7:43)
    Client hardware address padding: 00000000000000000000

    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  Option: (53) DHCP Message Type
      Length: 1
      DHCP: NAK (6)
  Option: (54) DHCP Server Identifier
      Length: 4
      DHCP Server Identifier: 192.168.0.1 (192.168.0.1)
  Option: (255) End
      Option End: 255
    Padding
```

## 5.5 Inform

❖ In the terminal of client, run the client program by enter the instruction "—inform".

```
root@BUPTIA:/home/dhcp# ./Client --inform
socket built successfully
bind failed.
Message type is: 350105
```

❖ In the wireshark, there is a inform packet and an ACK packet captured.

- Inform packet

```
39 449.966410000   192.168.0.111   192.168.0.1     DHCP          342 DHCP Inform   - Transaction ID 0x53a703dd
40 449.966762000   192.168.0.1     192.168.0.111   DHCP          342 DHCP ACK      - Transaction ID 0x53a703dd
```

```
39 449.966410000 192.168.0.111 192.168.0.1 DHCP 342 DHCP Inform  - Transaction ID 0x53a703dd
▼ Frame 39: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Jun 16, 2017 00:07:23.926963000 CST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1497542843.926963000 seconds
    [Time delta from previous captured frame: 0.001145000 seconds]
    [Time delta from previous displayed frame: 33.821910000 seconds]
    [Time since reference or first frame: 449.966410000 seconds]
    Frame Number: 39
    Frame Length: 342 bytes (2736 bits)
    Capture Length: 342 bytes (2736 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:udp:bootp]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
▽ Ethernet II, Src: CadmusCo_d3:fd:ae (08:00:27:d3:fd:ae), Dst: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5)
  ▷ Destination: CadmusCo_8d:34:a5 (08:00:27:8d:34:a5)
  ▷ Source: CadmusCo_d3:fd:ae (08:00:27:d3:fd:ae)
    Type: IP (0x0800)
▽ Internet Protocol Version 4, Src: 192.168.0.111 (192.168.0.111), Dst: 192.168.0.1 (192.168.0.1)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 328
    Identification: 0x3798 (14232)
  ▷ Flags: 0x02 (Don't Fragment)
```

```
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
  ▷ Header checksum: 0x804c [validation disabled]
    Source: 192.168.0.111 (192.168.0.111)
    Destination: 192.168.0.1 (192.168.0.1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▽ User Datagram Protocol, Src Port: 59460 (59460), Dst Port: bootps (67)
    Source port: 59460 (59460)
    Destination port: bootps (67)
    Length: 308
  ▷ Checksum: 0x0403 [validation disabled]
▽ Bootstrap Protocol
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x53a703dd
    Seconds elapsed: 0
  ▷ Bootp flags: 0x0000 (Unicast)
    Client IP address: 192.168.0.111 (192.168.0.111)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: CadmusCo_45:b7:43 (08:00:27:45:b7:43)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ▽ Option: (53) DHCP Message Type
      Length: 1
      DHCP: Inform (8)
  ▽ Option: (55) Parameter Request List
      Length: 13
      Parameter Request List Item: (1) Subnet Mask
      Parameter Request List Item: (3) Router
      Parameter Request List Item: (6) Domain Name Server
      Parameter Request List Item: (15) Domain Name
      Parameter Request List Item: (31) Perform Router Discover
      Parameter Request List Item: (33) Static Route
      Parameter Request List Item: (43) Vendor-Specific Information
      Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
      Parameter Request List Item: (46) NetBIOS over TCP/IP Node Type
      Parameter Request List Item: (47) NetBIOS over TCP/IP Scope
      Parameter Request List Item: (121) Classless Static Route
      Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
      Parameter Request List Item: (252) Private/Proxy autodiscovery
  ▽ Option: (255) End
      Option End: 255
    Padding
```

## 5.6 Turn Down the Server

- Turn down the Server

```
-------------DHCP_offer------------
choose is: 1
get ip is: 192.168.0.111
the address is wrong? 192.168.0.1
Receive successfully
iplease.clientmac is :08002745B743
iplease.lease_time is :30
Receive successfully
is is is: 192.168.0.111
^Z
[1]+  Stopped                 ./Server 30
root@BUPTIA:/home/dhcp#
```

❖ In the terminal of client, run the client program by enter the instruction "—serverdown".

```
root@BUPTIA:/home/dhcp# ./Client --serverdown
socket built successfully
bind failed.
Buffer is:
Send successfully
root@BUPTIA:/home/dhcp#
```

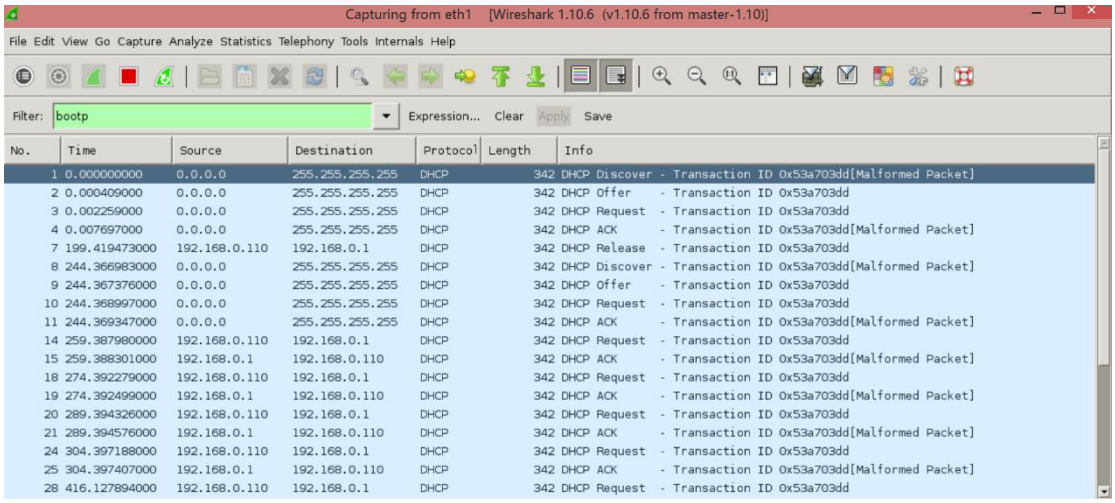
❖ In the wireshark, there are three packets captured. The second request packet is captured when 50% of lease time is passed, and the discover packet is captured when 7/8 of the lease time is passed.

| | | | | | | |
|---|---|---|---|---|---|---|
| 43 | 538.907929000 | 192.168.0.111 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 46 | 544.916253000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 47 | 546.930499000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Discover - Transaction ID 0x53a703dd[Malformed Packet] |

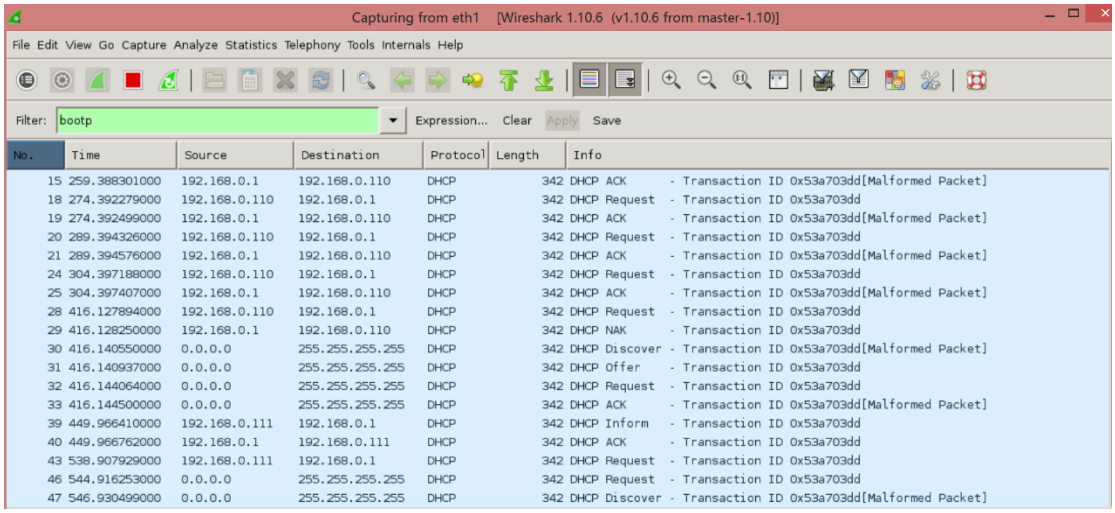
❖ Results in the terminal of server are shown as follows.

```
root@BUPTIA:/home/dhcp# ./Server 30
x is: 30bind successfully
i is: 4
get ip is: 192.168.0.110
Receive successfully
------------DHCP_offer------------
choose is: 0
get ip is: 192.168.0.110
the address is wrong? 192.168.0.1
Receive successfully
iplease.clientmac is :08002745B743
iplease.lease_time is :30
Receive successfully
which IP should release? 1IP is released, which is: 192.168.0.110
file descriper: 1
remove success
Receive successfully
------------DHCP_offer------------
choose is: 0
get ip is: 192.168.0.110
the address is wrong? 192.168.0.1
Receive successfully
iplease.clientmac is :08002745B743
iplease.lease_time is :30
Receive successfully
Receive successfully
Receive successfully
Receive successfully
Receive successfully
May be the address problem
Receive successfully
```

❖ Results in the wireshark are shown as follows.




| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Discover - Transaction ID 0x53a703dd[Malformed Packet] |
| 2 | 0.000409000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Offer - Transaction ID 0x53a703dd |
| 3 | 0.002259000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 4 | 0.007697000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 7 | 199.419473000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Release - Transaction ID 0x53a703dd |
| 8 | 244.366983000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Discover - Transaction ID 0x53a703dd[Malformed Packet] |
| 9 | 244.367376000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Offer - Transaction ID 0x53a703dd |
| 10 | 244.368997000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 11 | 244.369347000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 14 | 259.387980000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 15 | 259.388301000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 18 | 274.392279000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 19 | 274.392499000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 20 | 289.394326000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 21 | 289.394576000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 24 | 304.397188000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 25 | 304.397407000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 28 | 416.127894000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |




| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 15 | 259.388301000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 18 | 274.392279000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 19 | 274.392499000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 20 | 289.394326000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 21 | 289.394576000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 24 | 304.397188000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 25 | 304.397407000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 28 | 416.127894000 | 192.168.0.110 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 29 | 416.128250000 | 192.168.0.1 | 192.168.0.110 | DHCP | 342 | DHCP NAK - Transaction ID 0x53a703dd |
| 30 | 416.140550000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Discover - Transaction ID 0x53a703dd[Malformed Packet] |
| 31 | 416.140937000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Offer - Transaction ID 0x53a703dd |
| 32 | 416.144064000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 33 | 416.144500000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd[Malformed Packet] |
| 39 | 449.966410000 | 192.168.0.111 | 192.168.0.1 | DHCP | 342 | DHCP Inform - Transaction ID 0x53a703dd |
| 40 | 449.966762000 | 192.168.0.1 | 192.168.0.111 | DHCP | 342 | DHCP ACK - Transaction ID 0x53a703dd |
| 43 | 538.907929000 | 192.168.0.111 | 192.168.0.1 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 46 | 544.916253000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Request - Transaction ID 0x53a703dd |
| 47 | 546.930499000 | 0.0.0.0 | 255.255.255.255 | DHCP | 342 | DHCP Discover - Transaction ID 0x53a703dd[Malformed Packet] |