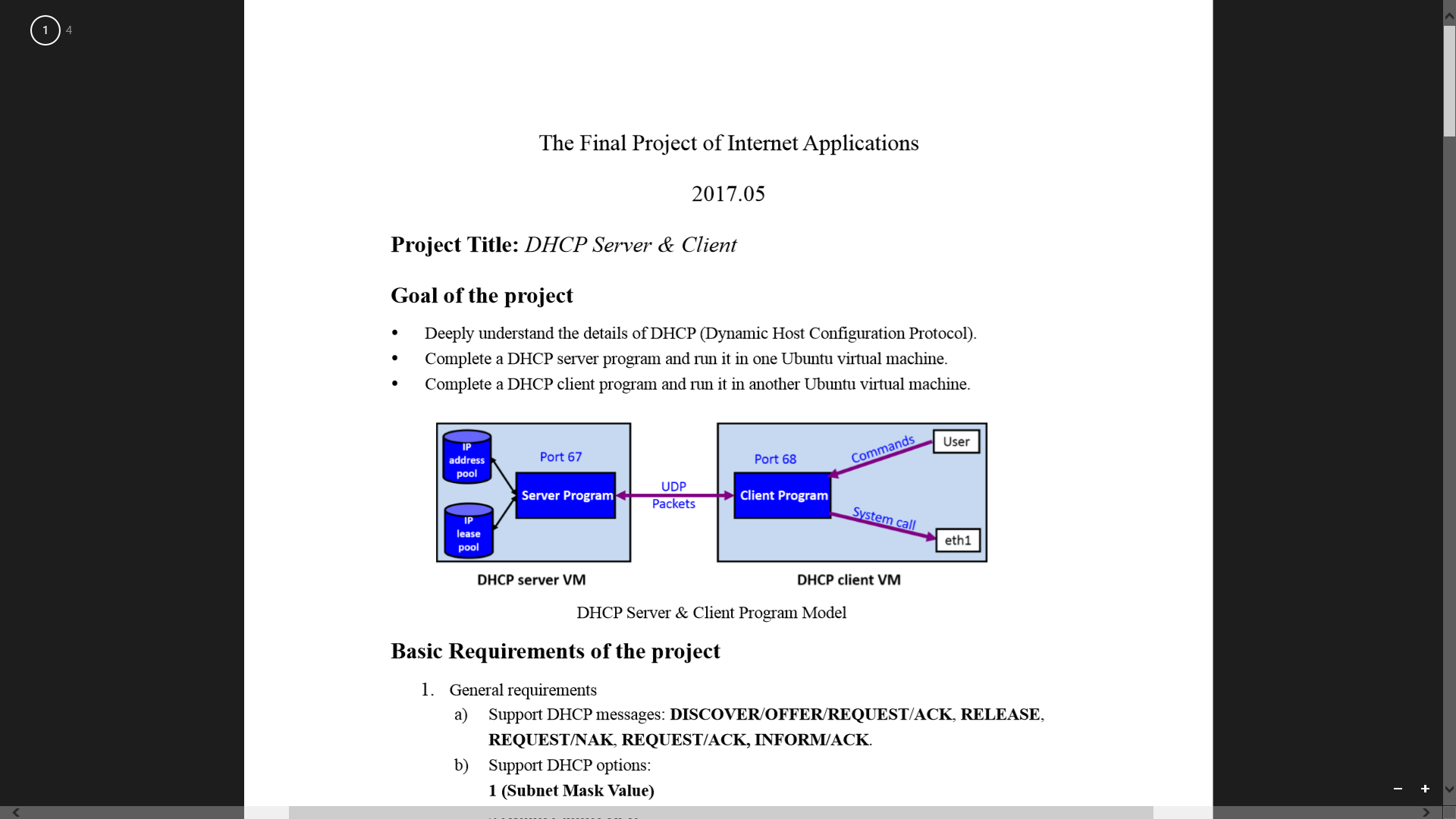
**1. Overview**



**1.1 Goal of the Project**

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

c) Complete a DHCP client program and run it in another Ubuntu virtual machine.

**1.2 General Requirements**

a) Support DHCP messages: DISCOVER/OFFER/REQUEST/ACK, RELEASE, REQUEST/NAK, REQUEST/ACK, INFORM/ACK.

b) Support DHCP options 1,3,6,51,53,54,55,58,59,60,255

c) Four messages during address acquisition can be delivered on broadcast packets.

d) Support 5 DHCP Procedures:

release + address acquisition

successful lease renew

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

failed lease renew (server down)

inform

**2. Requirements Analysis**

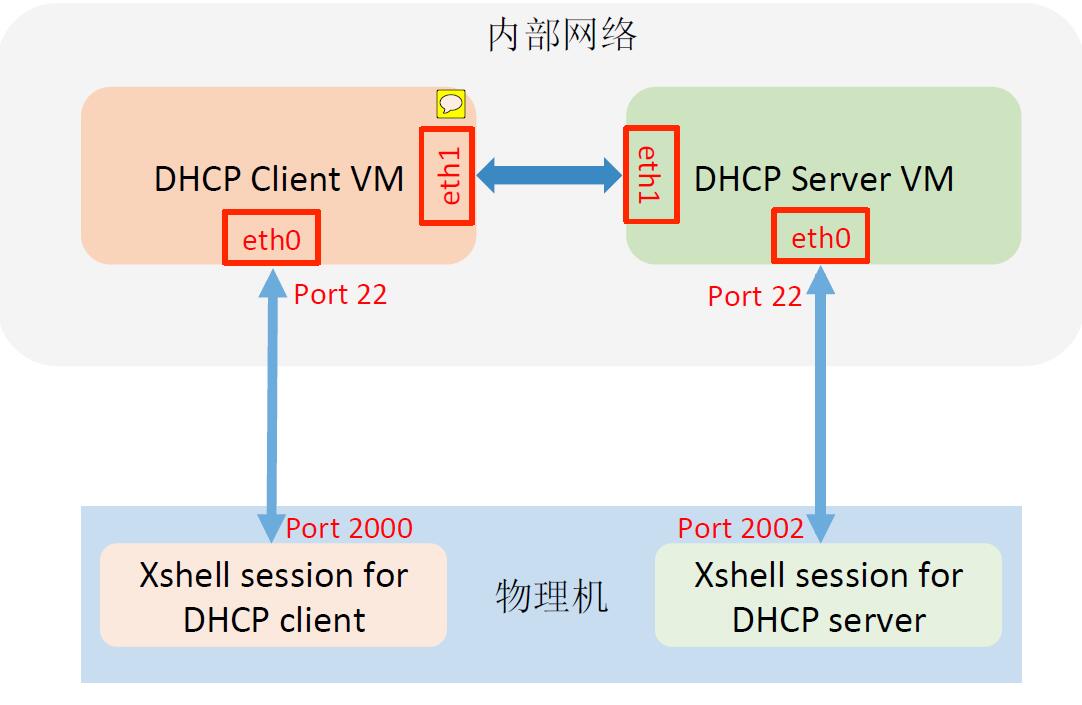
**2.1 Development environment description**

Operating System: Linux

Programming Language: C

Configuration:

1) Add port forwarding rules to satisfy the following diagram:



2) Set IP address on eth1 of server: 192.168.0.1, netmask: 255.255.255.0

Set IP address on eth1 of client: 0.0.0.0, netmask: 0.0.0.0

3) Add iptables rules to set source address to 0.0.0.0 when broadcast required.

4) Bind the port of server socket address to 67, bind the port of client socket address to 68.

**2.2 Detailed function requirements**

**1) Client**

|  |  |  |
| --- | --- | --- |
| **No.** | **Title** | **Description** |
| 1-1 | UDP connection | For sending and receiving packets |
| 1-2 | Data structure of DHCP packets | Construct a general DHCP packet structure at the beginning |
| 2-1 | Address Acquisition-1 | Send DHCPDISCOVER and receive OFFER |
| 2-2 | Address Acquisition-2 | Send DHCP REQUEST and receive ACK |
| 3-1 | Manual Renewal Success | Send DHCPREQUEST and receive ACK |
| 3-2 | Manual Renewal Failure (reject by server) | Send DHCPREQUEST and receive NAK |
| 4-1 | Automated Renewal Success | Send DHCPREQUEST when t1 expires automatically, and receive ACK |
| 4-2 | Automated Renewal Failure (server down) | Send DHCPREQUEST (unicast) when t1 expires automatically, and receive nothing, send DHCPREQUEST (broadcast) when t2 expires, send DHCPDISCOVER when t expires |
| 5-1 | Release IP address | Send DHCPRELEASE |
| 6-1 | Inform | Send DHCPINFORM and receive ACK |
| 7-1 | Interactive | Users can select functions to choose what to do. |

**2) Server**

|  |  |  |
| --- | --- | --- |
| **No.** | **Title** | **Description** |
| 1-1 | UDP connection | For sending and receiving packets |
| 1-2 | Data structure of DHCP packets | Construct a general DHCP packet structure at the beginning |
| 2-1-1 | Respond to address acquisition-1 | Receive DHCPDISCOVER and send OFFER. |
| 2-1-2 | Choose IP address from address pool | Select randomly from dhcp.config file and save the chosen one in DHCPOFFER |
| 2-2 | Respond to address acquisition-2 | Receive DHCPREQUEST and send ACK |
| 3-1 | Respond to renew request -success | Receive DHCPREQUEST and send ACK |
| 3-2 | Respond to renew request -failure | Receive DHCPREQUEST and send NAK |
| 4-1 | Respond to inform | Receive DHCPINFORM and send ACK |
| 5-1 | Respond to release address | Receive DHCPRELEASE and update dhcp.lease file |

**3. Preliminary Design**

**3.1 Data Structure Design**

Every DHCP message are stored in a struct(dhcp\_packet) showed below, all options are stored in a unsigned char array:

struct dhcp\_packet {

u\_int8\_t op; /\* 0: Message opcode/type \*/

u\_int8\_t htype; /\* 1: Hardware addr type (net/if\_types.h) \*/

u\_int8\_t hlen; /\* 2: Hardware addr length \*/

u\_int8\_t hops; /\* 3: Number of relay agent hops from client \*/

u\_int32\_t xid; /\* 4: Transaction ID \*/

u\_int16\_t secs; /\* 8: Seconds since client started looking \*/

u\_int16\_t flags; /\* 10: Flag bits \*/

struct in\_addr ciaddr; /\* 12: Client IP address (if already in use) \*/

struct in\_addr yiaddr; /\* 16: Client IP address \*/

struct in\_addr siaddr; /\* 18: IP address of next server to talk to \*/

struct in\_addr giaddr; /\* 20: DHCP relay agent IP address \*/

unsigned char chaddr [DHCP\_CHADDR\_LEN]; /\* 24: Client hardware address \*/

char sname [DHCP\_SNAME\_LEN]; /\* 40: Server name \*/

char file [DHCP\_FILE\_LEN]; /\* 104: Boot filename \*/

uint32\_t magic\_cookie;

unsigned char padding[64];

unsigned char options[DHCP\_MAX\_OPTION\_LEN];

};

Different DHCP messages share the same struct but vary in the content. So we declare different set packet function to get each packet in dhcp.h.

**3.2 Module Decomposition**

**3.2.1 Client**

a) UDP connection for sending and receiving packets

b) Address Acquisition

Send DHCP DISCOVER and receive OFFER: void getIPAddrClt()

Send DHCP REQUEST and receive ACK: sendReq()

c) Renew

Send DHCP REQUEST and receive ACK/NAK: void sendReq(struct in\_addr curIP, unsigned char\* option50, u\_int32\_t leaseTime,int optcast)

Send DHCP REQUEST when t1 expires and no packet sent manually: void timerHandler(int sig)

d) Release IP address

Send DHCP RELEASE and receive ACK: void releaseAddr(char \*curIPChar)

e) Inform

Send DHCP INFORM and receive ACK: void informClt()

f) Select Function

choose what to do: void selectFunc();

**3.2.2 Server**

a) UDP connection for sending and receiving packets

b) Respond to address acquisition

Receive DHCP DISCOVER and send OFFER: void getIPAddrSvr()

Receive DHCP REQUEST and send ACK: void setRenew(struct dhcp\_packet echoBuffer,unsigned char option51[4])

c) Respond to renew request

Receive DHCP REQUEST and send ACK/NAK: void setRenew(struct dhcp\_packet echoBuffer,unsigned char option51[4])

d) Respond to inform

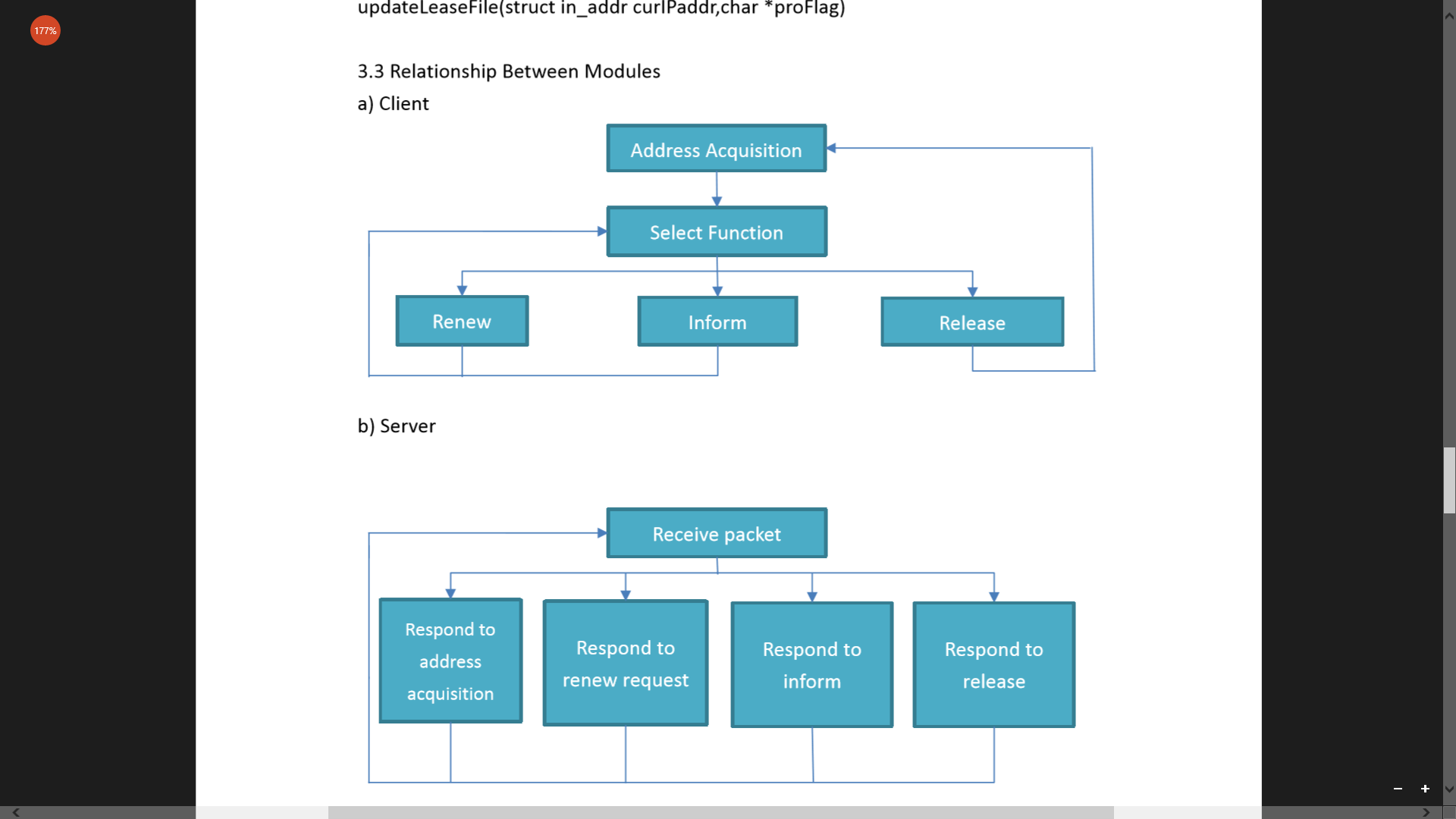
Receive DHCP INFORM and send ACK: void informSvr()

e) Respond to release address

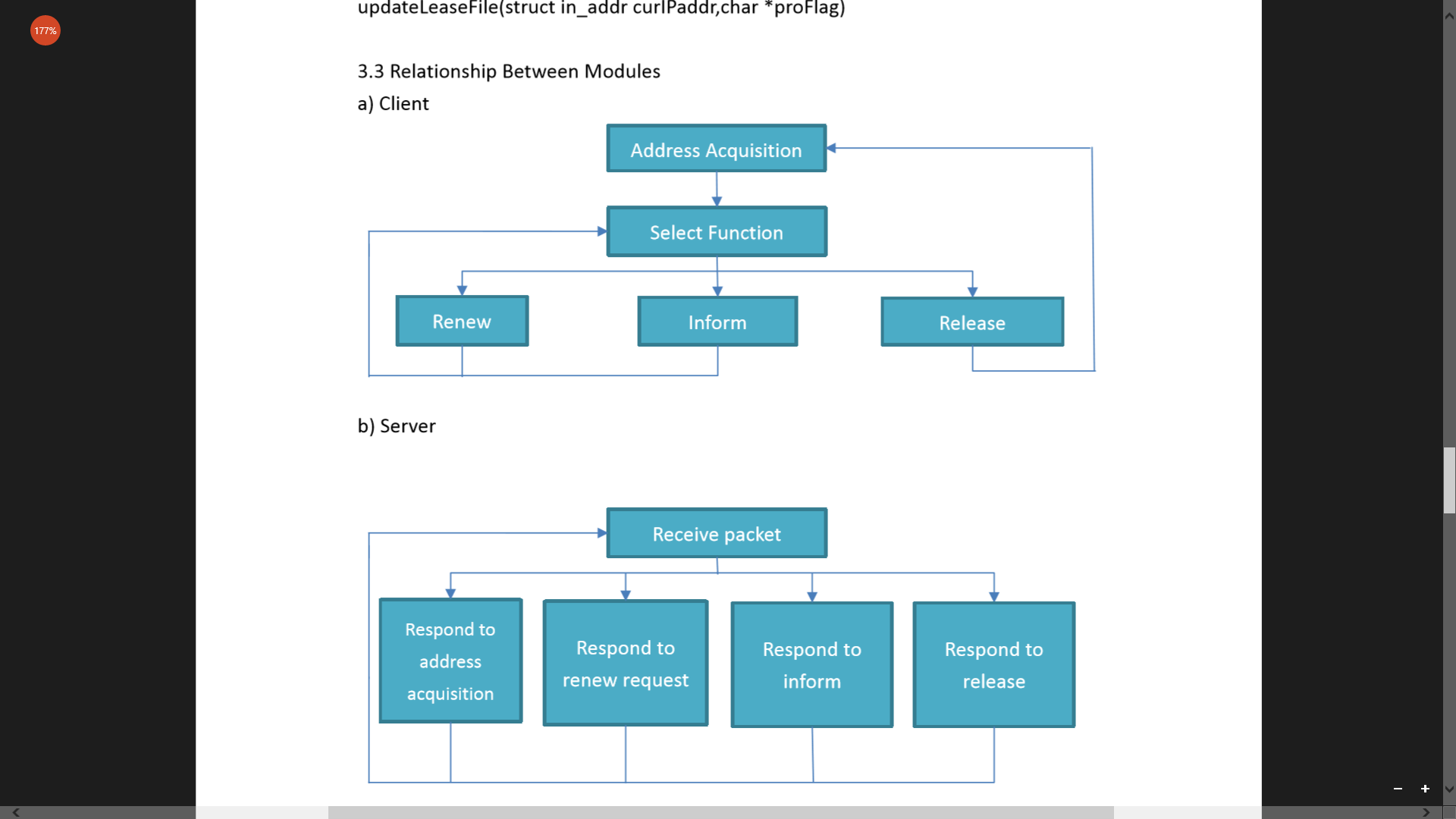
Receive DHCP RELEASE and send ACK: releaseAddr(): void updateLeaseFile(struct in\_addr curIPaddr,char \*proFlag)

**3.3 Relationship Between Modules**

a) Client



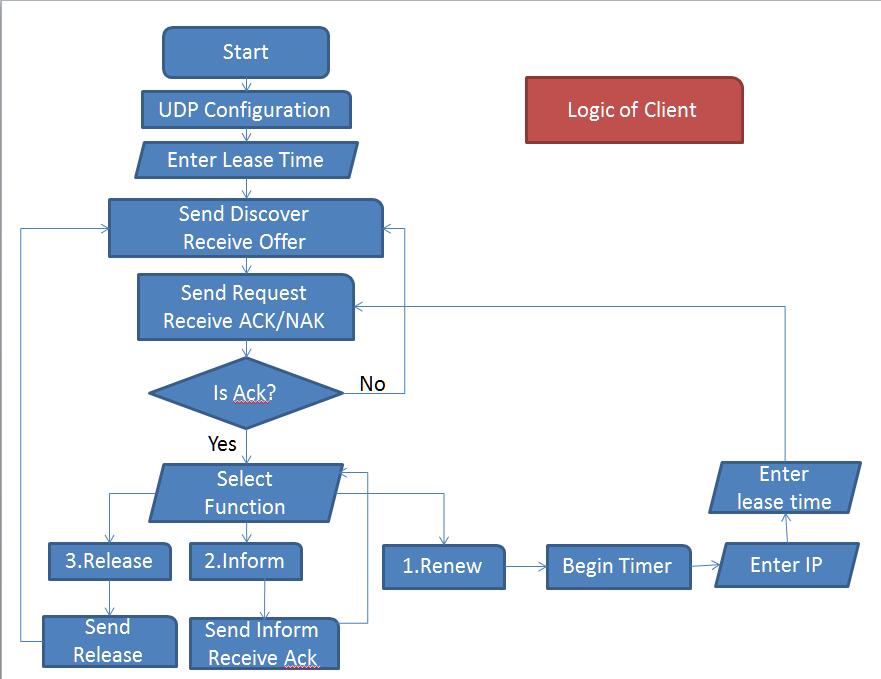
b) Server



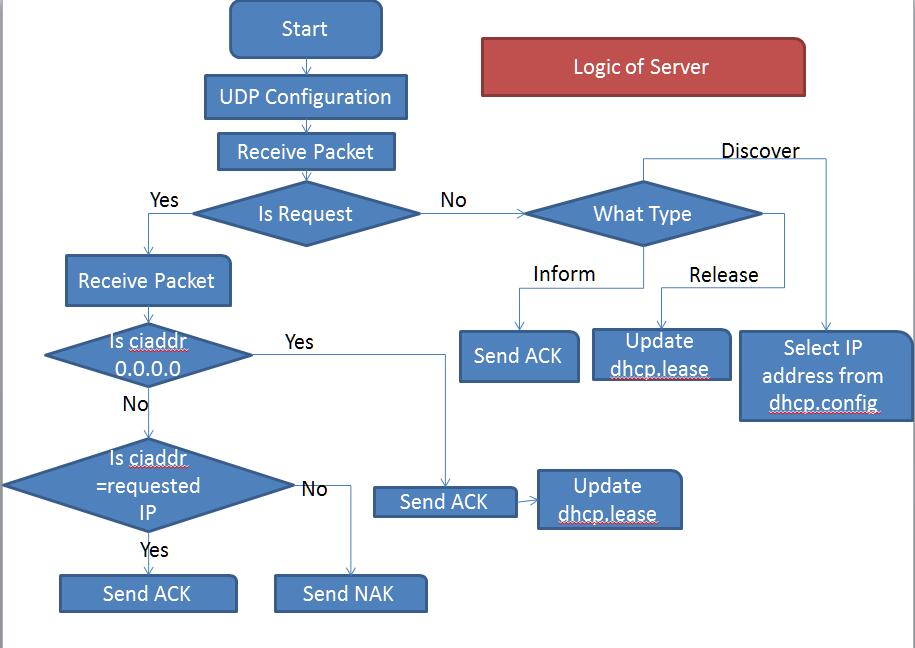
**4. Detailed Design**

**4.1 Process diagrams**

Logic of Client:



Logic of Server:

****

**4.2 Send and receive packets using UDP**

**1) Create socket:**

sock = socket(PF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

**2) Bind eth1 and set acceptance of broadcast**

setsockopt(sock,SOL\_SOCKET,SO\_BROADCAST,&i,len);

setsockopt(sock,SOL\_SOCKET,SO\_BINDTODEVICE,"eth1",sizeof(if\_eth1));

**3) Bind client socket address**

bind(sock, (struct sockaddr \*)&echoClntAddr,sizeof(echoClntAddr));

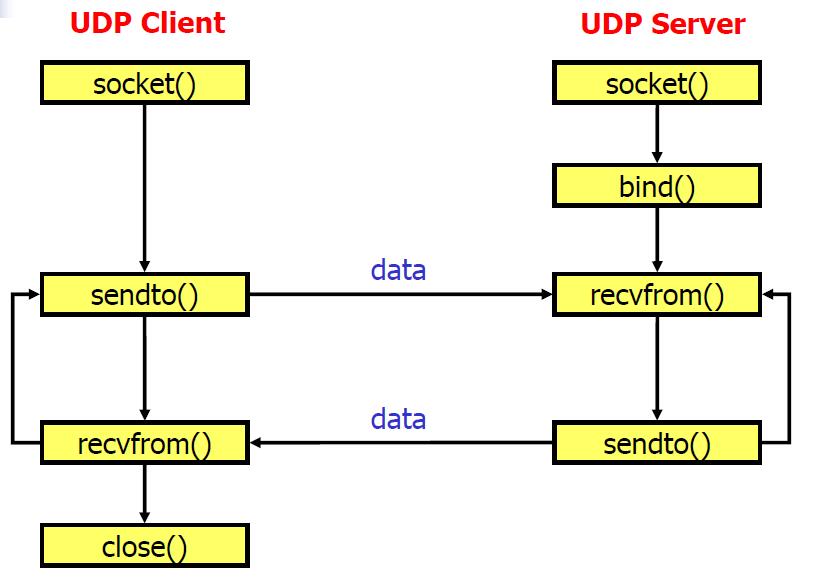
**4) Send DHCP packets using UDP**

sendto(sock, &tempPkt, sizeof(tempPkt), 0,(struct sockaddr \*)&echoServAddr, sizeof(echoServAddr));

**5) Receive DHCP packets using UDP**

recvfrom(sock, &replyPkt, ECHOMAX, (struct sockaddr \*)&fromAddr, &fromAddrLen);

**6) Close**



**4.3 Global variables/reference description**

**1) Head files and defined values**

#include "dhcp.h"

dhcp.h: head file, which contains the structure of all DHCP packets defined by us, and all packets generation (value assignment) functions.

#define ECHOMAX 2500

ECHOMAX: the maximum bytes that can be received at one time

#define IPPOOLSIZE 10

Size of IP address pool

**2) Used for UDP communication**

int sock;

struct sockaddr\_in echoServAddr;

struct sockaddr\_in echoClntAddr;

struct sockaddr\_in echoFromAddr;

unsigned short echoServPort ;

unsigned short echoClntPort ;

struct dhcp\_packet tempPkt;

be used to store temporary data of packets to be sent

struct dhcp\_packet replyPkt:

be used to store temporary data of packets to be received

**3) Used for DHCP requirements**

Struct in\_addr curIP:

Current IP address that client has

struct in\_addr setEthIP:

IP address entered by user in client side when renewing, used to judge whether the user enters a correct IP address

u\_int32\_t leaseTime:

Required lease time that can be accessed globally

long t1, t2, t3:

current time+0.5t, current time+0.875t, current time+t

unsigned char\* option50:

Requested IP address that can be accessed globally

unsigned char option51[4]:

store lease time in DHCP packet option field

int optcast:

A flag to set whether broadcast or unicast when sending DHCPREQUEST

**4.4 Module design**

**4.4.1 Client module**

**1) Interactive function selection**

Void selectFunc();

Operation: There is three functions to choose after the address acquisition process: Renew, Inform and release. Identify the user input and call on each function. In the first option—renew, print the instructions to let users input requested IP address and lease time, then scan the input information. In the meanwhile, set a timer(call on timeHandler(int sig)) to check if t1 expires.

**2) Address release**

void releaseAddr(char \*curIPChar);

Parameter: pass current IP address to this function, which will be released after execution.

Operation: Get DHCPRELEASE packet and change its field “ciaddr” to current IP address. Send this packet to server, after receiving DHCPACK packet, call the system command to unbind IPv4 address of eth1, and print success log.

**3) IP address acquisition**

void getIPAddrClt();

Operation: Set lease time that has been entered by user. Create DHCPDISCOVER packet and sent it to server. Receive DHCPOFFER packet and save the IP address into option50 (requested IP address) from “yiaddr” field of offer. Call sendReq() function to continue the process of sending DHCPREQUEST and receiving DHCPACK.

**4) Renew (shared by auto mode and manual mode)**

void sendReq(struct in\_addr curIP, unsigned char\* option50, u\_int32\_t leaseTime,int optcast);

Parameter: struct in\_addr curIP -- users’ current IP address, ciaddr of the DHCP REQUEST packet.

unsigned char\* option50 -- requested IP address, read from DHCP OFFER and stores in option 50 requested IP of DHCP REQUEST packet.

u\_int32\_t leaseTime -- lease time set by the user, send to the server alone.

int optcast -- 1 for unicast and 0 for broadcast.

Operation: Form different DHCP REQUEST packet for all process that need to send request with different parameters. Send request to server and receive ACK or NAK, calls on different function for different type of respond. When the renew request is sent, if the server is down and no DHCP ACK packet is received, it will sleep until t2 expires and send another DHCP REQUEST packet by broadcast. If no ACK is received, sleep until t expires and begin address acquisition process.

**5) Timer (auto mode)**

void timerHandler(int sig);

Operation: Check if t1 expires when the user is entering required information. Send DHCP REQUEST packet by calling on SendReq() if time expires .

**6) Inform function**

void informClt();

Operation: Create DHCPINFORM packet and send it to server. Print success log after receiving DHCPACK and step to function selection mode.

**4.4.2 Server module**

**1) IP address acquisition**

void getIPAddrSvr();

Operation: Randomly choose an IP address from our address pool (dhcp.config), create DHCPOFFER wrapping the chosen IP address in “yiaddr” and send to client. Receive DHCPREQUEST and call on our setRenew() function, which passes on the DHCPREQUEST packet and lease time entered by user.

**2) Renew process**

void setRenew(struct dhcp\_packet echoBuffer,unsigned char option51[4]);

Parameter: struct dhcp\_packet echoBuffer -- the DHCP packet received.

unsigned char option51[4] -- lease time received in another packet.

Operation: Send DHCP ACK packet to different REQUEST packet. If the request is of the address acquisition, the ciaddr and yiaddr of DHCPACK packet is randomly chosen from the dhcp.config. If the request is of the renew process, it will check if the requested IP is correct. If correct, send DHCPACK and the ciaddr, yiaddr is the same with the ciaddr in the received REQUEST packet, otherwise send NAK. Option 51 (lease time) of the ACK packet is from the parameter passing in.

**3) Inform process**

void informSvr();

Operation: This function is entered after server receives a DHCPINFORM packet. It creates DHCPACK according to the “ciaddr” from DHCPINFORM and sends it to client.

**4) Lease file update**

void updateLeaseFile(struct in\_addr curIPaddr,char \*proFlag);

Parameters: curIPaddr -- the IP address that server wants to modify status or details.

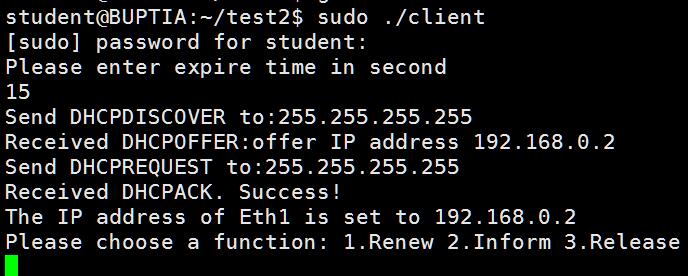
proFlag -- whether the IP address that passed in is on use or not, 1 for on use and 0 for not on use

Operation: Open the “dhcp.lease” file, store every line of the file into a variable: char \*temp[10]. Separate each element using strtok method according to delimiter “;”. Search if the required address has already been in this file. If it does, change the relative status in this variable, which means changing 0 to 1 in address acquisition, 1 to 0 in address release. If it doesn’t, add a new line to the variable. Write back to the file with the entire updated data.

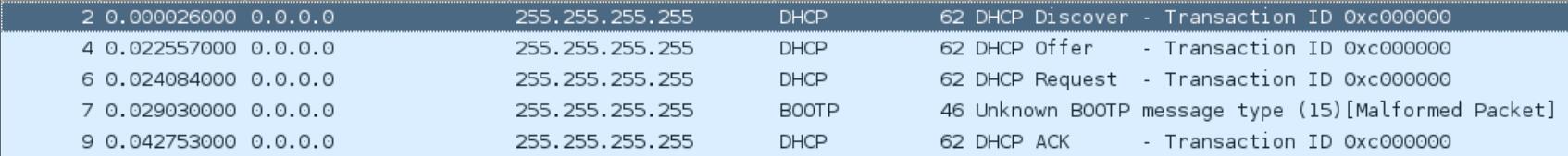
**5. Results**

**1) Address acquisition**

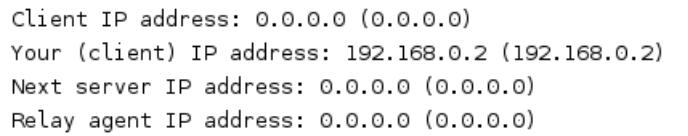
Run the system, it requires put in a lease time. After entering it, it completes the address acquisition process and entered selection mode:



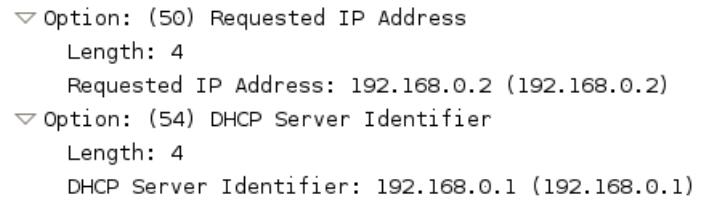
Wireshark packets capturing:



Emphasis points in DHCPOFFER:

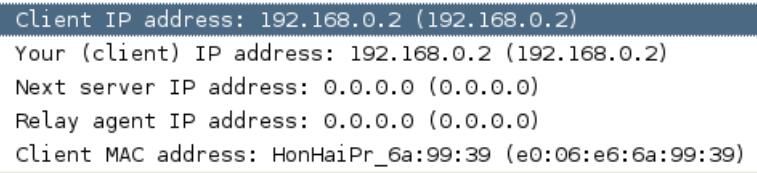


Emphasis points in DHCPREQUEST:



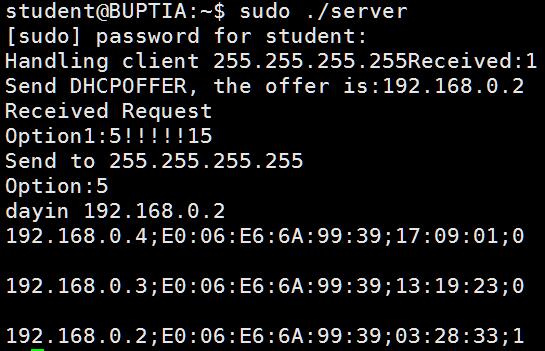
Emphasis points in DHCPACK:

(where lease time 15s is entered by user before)



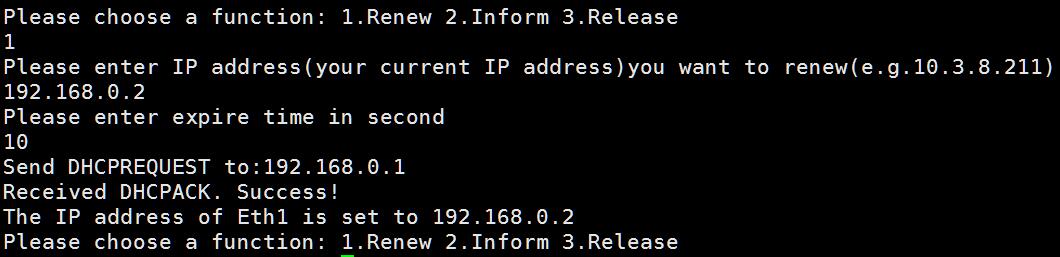


On server side, it prints some important information and also the new contents of “dhcp.lease” file.

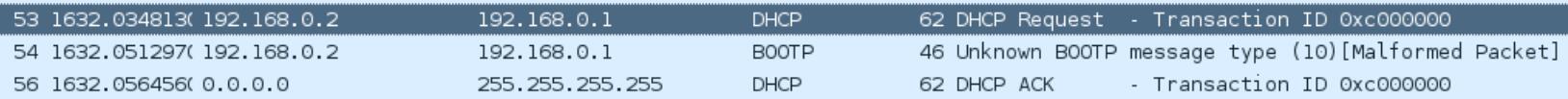


**2) Manual renewal success**

After I choose function 1 for renewal, if I entered a correct IP address which is consistent with current using address, it would receive a DHCPACK packet and enter the function selection mode again.



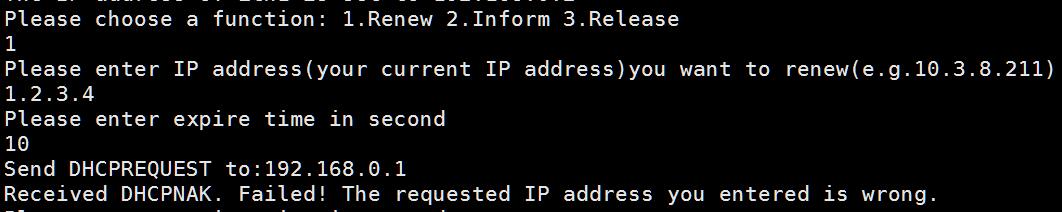
In Wireshark,



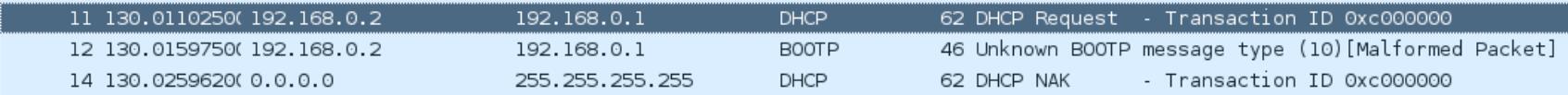


**3) Manual renewal failure**

After I choose function 1 for renewal, if I entered a wrong IP address which is not consistent with current using address, it would receive a DHCPNAK packet and prompt you to try again.

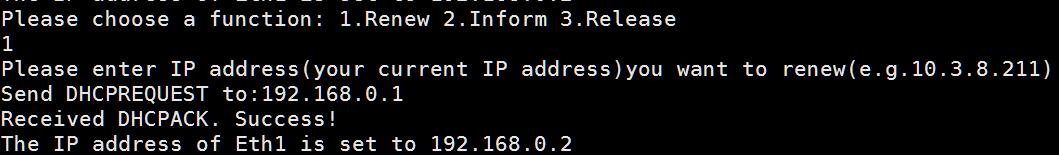


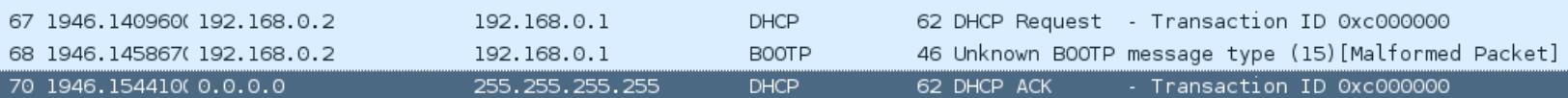
In Wireshark:



**4) Automated renewal success**

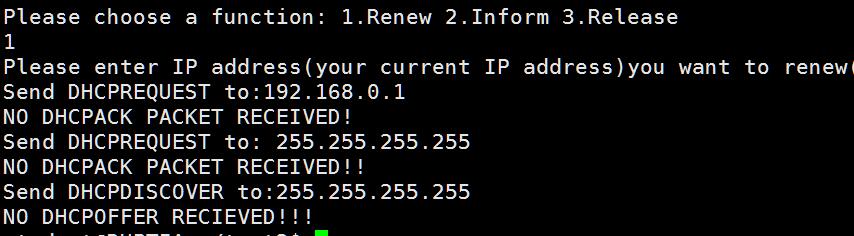
If you entered renewal mode but waited until t1 (0.5 of the lease time that has been entered at first), it would automatically send a DHCPREQUEST to server. If the server isn’t down, it would receive a DHCPACK and finish renewal process.



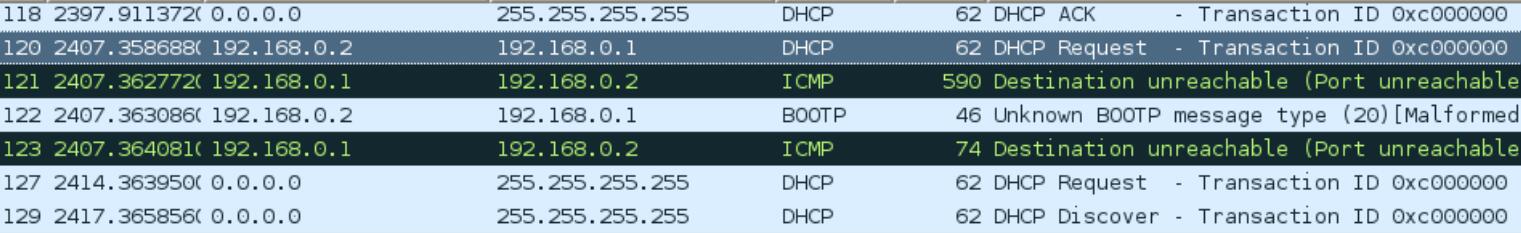


**5) Renewal failure when server is down**

If you entered renewal mode but waited until t1 (0.5 of the lease time that has been entered at first), it would automatically send a DHCPREQUEST to server by unicast. If the server is down, it would receive nothing and send another DHCPREQUEST by broadcast until t2 reaches (0.875 of the lease time). If there’s no response until t3 reaches (lease time ends), it would send a DHCPDISCOVER and enter address acquisition mode.

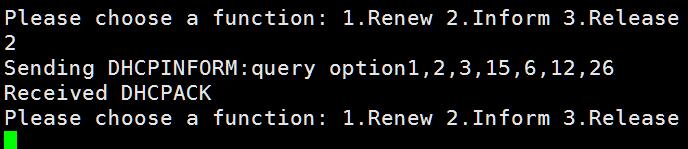


In Wireshark, we can see if we set the lease time as 20s, duration between ACK and 1st Request is 10s, duration between 1st Request and 2nd Request is 7.5s, and duration between 2nd Request and final Discover is 2.5s.

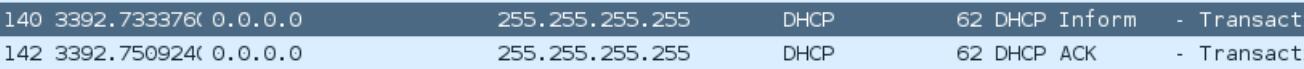


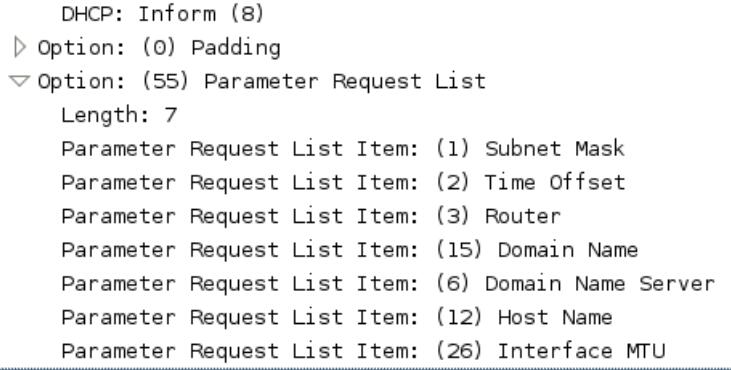
**6) Inform process**

After selecting function 2, client sends a DHCPINFORM to server and receives a DHCPACK



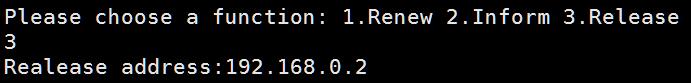
In Wireshark, we can see Option 55 as the parameter request list:



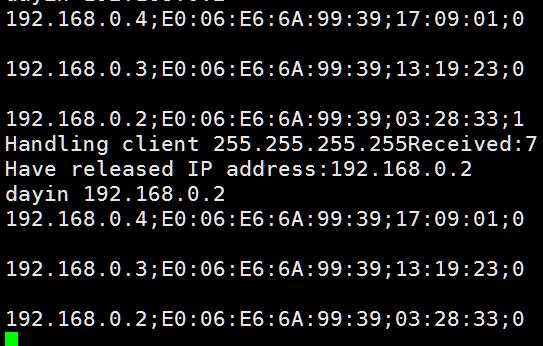


**7) Release process**

After selecting function 3, client sends a DHCPRELEASE and prints the released address.



On server side, we can see “dhcp.lease” file has been rewritten, and the status of related IP address record has been changed from 1 (used) to 0 (unused).



In Wireshark, we can see the details about DHCPRELEASE packet.

