Федеральное агентство связи

Сибирский Государственный Университет Телекоммуникаций и Информатики

Кафедра ТС и ВС

Лабораторная работа № 3 **“ Разработка простейшего сервера DHCP”**

По дисциплине: **Сетевое программное обеспечение**

**Выполнил:** Тиванов Данил Евгеньевич

Группа: ИА-832

**Проверил:** Лошкарев Александр Васильевич

**Задание:**Разработать DHCP сервер, выполняющий регистрацию запросов от клиентов, формирование предложения сетевых настроек, отправки настроек клиенту и получения подтверждения принятия настроек.

DCHP сервер должен выполнять следующий процесс выдачи сетевых адресов:

* Обнаружение DHCP;
* Предложение DHCP;
* Запрос DHCP;
* Подтверждение DHCP.

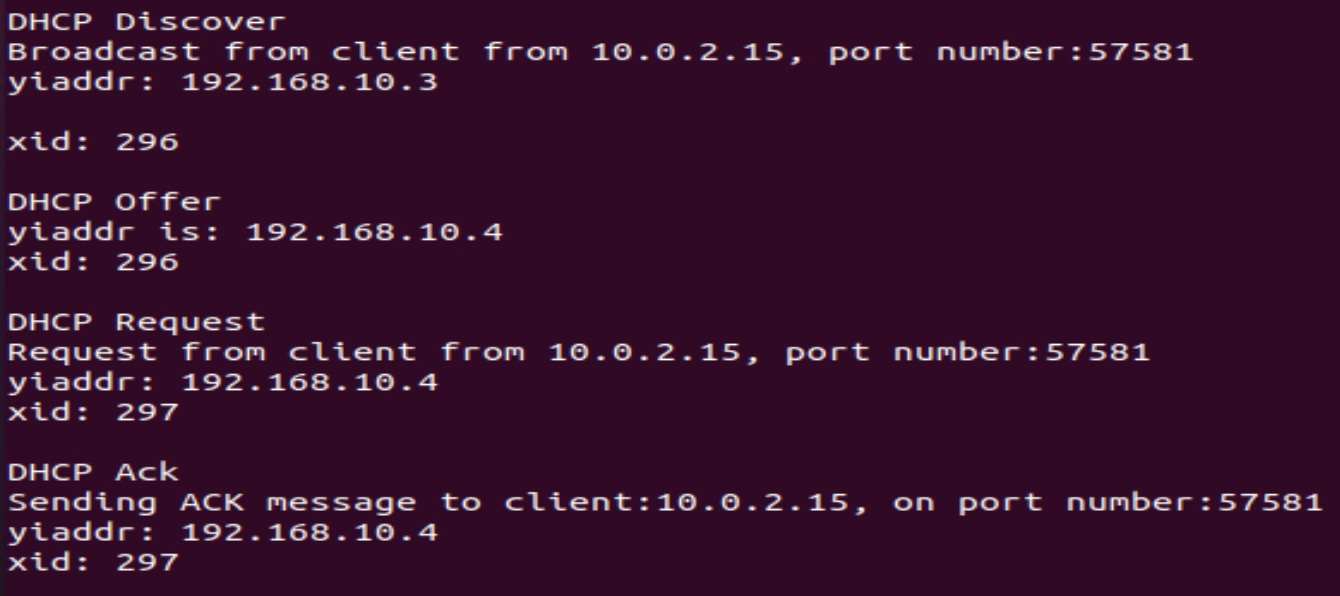
Язык разработки - С/С++

Для сетевого взаимодействия используем советы Беркли.

**Результаты:**

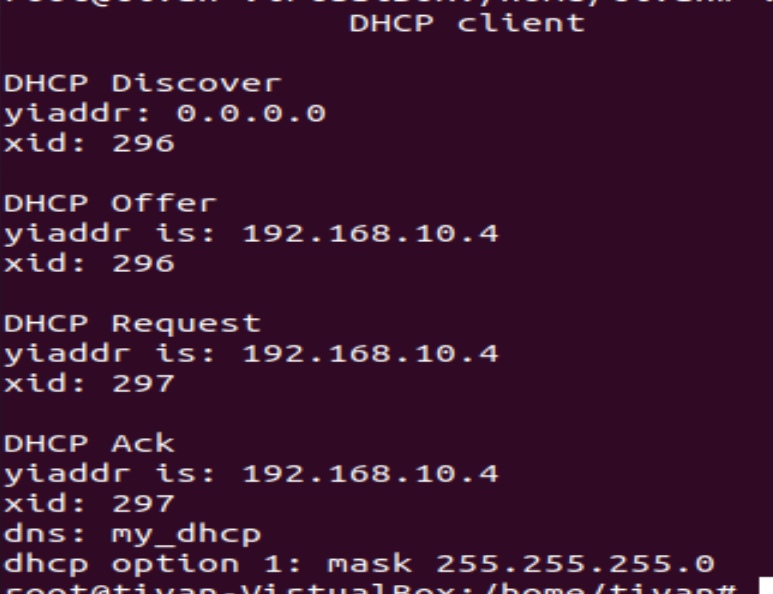
* Клиент отправляет запрос с номером транзакции 310 (Discover)
* Сервер получает сообщение от клиента и предлагает IP-адрес клиенту (Offer);

server попытка 1

  
• Сервер получает сообщение от клиента и предлагает IP-адрес клиенту (Offer);

• Сервер подтверждает (Ack).

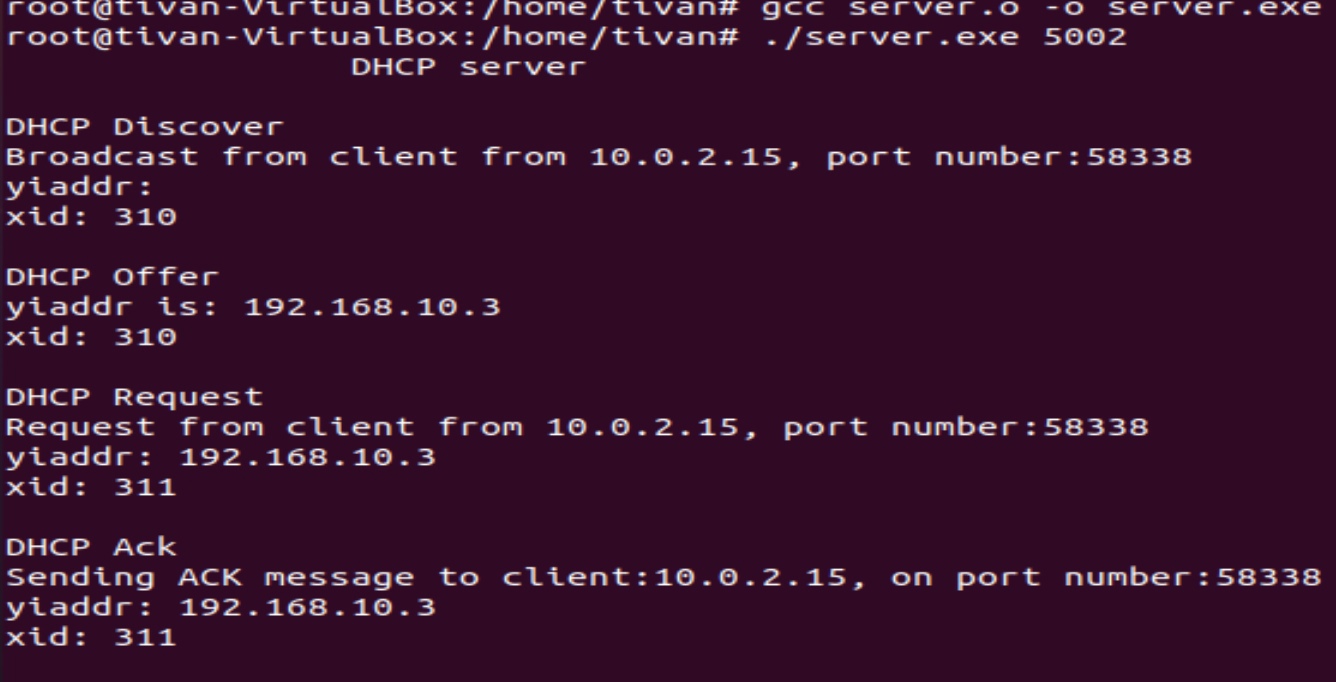
client попытка 1

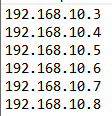


client попытка 2



server попытка 2.



свободные IP берутся из файла  


В случае если IP закончатся, сервер выведет на экран

**Код программы:**

server.c

#include<stdio.h>

#include<ctype.h>

#include<stdlib.h>

#include<string.h>

#include<unistd.h>

#include<signal.h>

#include<sys/wait.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<errno.h>

#include<sys/file.h>

#include<sys/msg.h>

#include<sys/ipc.h>

#include<time.h>

#define BUFLEN 512

void **die**(char \*s){

perror(s);

exit(1);

}

struct dhcpmessage

{

uint8\_t op;

uint8\_t htype;

uint8\_t hlen;

uint8\_t hops;

uint32\_t xid;

uint16\_t secs;

uint16\_t flags;

uint32\_t ciaddr;

uint32\_t yiaddr;

uint32\_t siaddr;

uint32\_t giaddr;

char chaddr[16];

char sname[64];

char file[128];

char magic[4];

char opt[3];

};

int **main**(int argc, char \*argv[]){

struct dhcpmessage dhcpmsg;

struct **sockaddr\_in** sa\_server, si\_other;

int s, i = 1, slen = sizeof(si\_other), recv\_len, xid = 0;

char cyiaddr[BUFLEN] = "";

char fileGetIPAddr[50], string[50];

FILE \*fileStream;

if((s = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP)) == -1)

die("socket");

memset((char \*) &sa\_server, 0, sizeof(sa\_server));

sa\_server.sin\_family = AF\_INET;

sa\_server.sin\_port = htons(atoi(argv[1]));

sa\_server.sin\_addr.s\_addr = htonl(INADDR\_ANY);

if(bind(s, (struct sockaddr \*)&sa\_server, sizeof(sa\_server)) == -1)

die("bind");

printf("\t\tDHCP server\n");

while(1){

//DHCP discover

printf("\nDHCP Discover\n");

if((recv\_len = recvfrom(s, &dhcpmsg, sizeof(dhcpmsg), 0, (struct sockaddr \*) &si\_other, &slen)) == -1)

die("recvfrom()");

printf("Broadcast from client from %s, port number:%d\n", inet\_ntoa(si\_other.sin\_addr), ntohs(si\_other.sin\_port));

printf("yiaddr: %s\n", cyiaddr);

printf("xid: %d\n", dhcpmsg.xid);

//DHCP offer

printf("\nDHCP Offer");

fileStream = fopen("IPs.txt", "r"); //IP addresses from file

for(int j = 0; j < i; ++j, fgets(fileGetIPAddr, 50, fileStream));

strcpy(cyiaddr, fileGetIPAddr);

printf("\nyiaddr is: %s", cyiaddr);

printf("xid: %d\n", dhcpmsg.xid);

inet\_aton(cyiaddr, &dhcpmsg.yiaddr);

strcpy(dhcpmsg.sname, "my\_dhcp");

dhcpmsg.opt[1] = 1; // mask 255.255.255.0

if(sendto(s, &dhcpmsg, sizeof(dhcpmsg), 0, (struct sockaddr \*) &si\_other, slen) == -1)

die("sendto()");

//DHCP request

printf("\nDHCP Request\n");

if((recv\_len = recvfrom(s, &dhcpmsg, sizeof(dhcpmsg), 0, (struct sockaddr \*) &si\_other, &slen)) == -1)

die("recvfrom()");

printf("Request from client from %s, port number:%d\n", inet\_ntoa(si\_other.sin\_addr), ntohs(si\_other.sin\_port));

printf("yiaddr: %s", cyiaddr);

printf("xid: %d\n", dhcpmsg.xid);

//DHCP ack

printf("\nDHCP Ack\n");

printf("Sending ACK message to client:%s, on port number:%d\n", inet\_ntoa(si\_other.sin\_addr), ntohs(si\_other.sin\_port));

printf("yiaddr: %s", cyiaddr);

printf("xid: %d\n", dhcpmsg.xid);

dhcpmsg.opt[1] = 1; // mask 255.255.255.0

if(sendto(s, &dhcpmsg, sizeof(dhcpmsg), 0, (struct sockaddr \*) &si\_other, slen) == -1)

die("sendto()");

i++;

if(i == 7){

printf("Run out of IPs..\n");

break;

}

printf("\n\n");

}

fclose(fileStream);

close(s);

return 0;

}

client.c  
  
#include<stdio.h>

#include<ctype.h>

#include<stdlib.h>

#include<string.h>

#include<unistd.h>

#include<signal.h>

#include<sys/wait.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<errno.h>

#include<sys/file.h>

#include<sys/msg.h>

#include<sys/ipc.h>

#include<time.h>

#define SERVER "10.0.2.15"

#define BUFLEN 512

void **die**(char \*s){

perror(s);

exit(1);

}

struct dhcpmessage

{

uint8\_t op;

uint8\_t htype;

uint8\_t hlen;

uint8\_t hops;

uint32\_t xid;

uint16\_t secs;

uint16\_t flags;

uint32\_t ciaddr;

uint32\_t yiaddr;

uint32\_t siaddr;

uint32\_t giaddr;

char chaddr[16];

char sname[64];

char file[128];

char opt[3];

};

int **main**(int argc, char \*argv[]){

struct **sockaddr\_in** sa;

struct dhcpmessage dhcpmsg;

dhcpmsg.op = 1;

dhcpmsg.htype = 1;

dhcpmsg.hlen = 6;

dhcpmsg.hops = 0;

dhcpmsg.xid = htonl(1000);

dhcpmsg.secs = htons(0);

dhcpmsg.flags = htons(0x8000);

dhcpmsg.chaddr[0] = 0x00;

dhcpmsg.chaddr[1] = 0x1A;

dhcpmsg.chaddr[2] = 0x80;

dhcpmsg.chaddr[3] = 0x80;

dhcpmsg.chaddr[4] = 0x2C;

dhcpmsg.chaddr[5] = 0xC3;

dhcpmsg.opt[0]=53;

dhcpmsg.opt[1]=1;

dhcpmsg.opt[2]=1;

int s, slen = sizeof(sa), recv\_len, xid;

char yiaddr[BUFLEN] = "0.0.0.0";

srand(time(0));

printf("\t\tDHCP client\n");

if((s = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP)) == -1){

die("socket");

}

memset((char \*) &sa, 0, sizeof(sa));

sa.sin\_family = AF\_INET;

sa.sin\_port = htons(atoi(argv[1]));

if (inet\_aton(SERVER , &sa.sin\_addr) == 0) {

fprintf(stderr, "inet\_aton() failed\n");

exit(1);

}

//DHCP discover

printf("\nDHCP Discover\n");

printf("yiaddr: %s\n", yiaddr);

dhcpmsg.xid = rand() % 900 + 100;

dhcpmsg.yiaddr = htonl( "0.0.0.0" );

printf("xid: %d\n", dhcpmsg.xid);

if(sendto(s, &dhcpmsg, sizeof(dhcpmsg), 0, (struct sockaddr \*) &sa, slen) == -1)

die("sendto()");

//DHCP offer

if((recv\_len = recvfrom(s, &dhcpmsg, sizeof(dhcpmsg), 0, (struct sockaddr \*) &sa, &slen)) == -1)

die("recvfrom()");

printf("\nDHCP Offer\n");

struct **in\_addr** ip\_addr;

ip\_addr.s\_addr = dhcpmsg.yiaddr;

printf("yiaddr is: %s\n", inet\_ntoa(ip\_addr));

printf("xid: %d\n", dhcpmsg.xid);

//DHCP request

printf("\nDHCP Request\n");

ip\_addr.s\_addr = dhcpmsg.yiaddr;

printf("yiaddr is: %s\n", inet\_ntoa(ip\_addr));

printf("xid: %d\n", ++dhcpmsg.xid);

//Send to DHCP server

if(sendto(s, &dhcpmsg, sizeof(dhcpmsg), 0, (struct sockaddr \*) &sa, slen) == -1)

die("sendto()");

//DHCP ack

if((recv\_len = recvfrom(s, &dhcpmsg, sizeof(dhcpmsg), 0, (struct sockaddr \*) &sa, &slen)) == -1)

die("recvfrom()");

printf("\nDHCP Ack\n");

ip\_addr.s\_addr = dhcpmsg.yiaddr;

printf("yiaddr is: %s\n", inet\_ntoa(ip\_addr));

printf("xid: %d\n", dhcpmsg.xid);

printf("dns: %s\n", dhcpmsg.sname);

printf("dhcp option 1: mask 255.255.255.0\n");

close(s);

return 0;

}