**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Дисциплина: Сети ЭВМ и телекоммуникации**

**Тема: Протоколы ARP/RARP**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: ст. пр. ПО и ВТАС

Федотов Евгений Александрович

**Белгород 2020**

**Цель работы:** изучить принципы и характеристику протоколов ARP/RARP.

**Краткие теоретические сведения**

**ARP** (**Address Resolution Protocol** - протокол определения адреса) - протокол канального уровня, предназначенный для определения MACадреса (адреса канального уровня) по известному IP-адресу (адресу сетевого уровня). Наибольшее распространение этот протокол получил благодаря распространению сетей IP, построенных поверх Ethernet, поскольку практически в 100 % случаев при таком сочетании используется протокол ARP.

Протокол ARP работает различным образом в зависимости от того, какой протокол канального уровня работает в данной сети - протокол локальной сети (Ethernet, Token Ring, FDDI) с возможностью широковещательного доступа одновременно ко всем узлам сети, или же протокол глобальной сети (X.25, frame relay), как правило не поддерживающий широковещательный доступ. Функциональность протокола ARP сводится к решению двух задач. Одна часть протокола определяет физические адреса при посылке дейтаграммы, другая отвечает на запросы устройств в сети. Протокол ARP предполагает, что каждое устройство «знает» как свой IP -адрес, так и свой физический адрес.

Для того чтобы уменьшить количество посылаемых запросов ARP, каждое устройство в сети, использующее протокол ARP, должно иметь специальную буферную память. В ней хранятся пары адресов (IP-адрес, физический адрес) устройств в сети. Всякий раз, когда устройство получает ARP-ответ, оно сохраняет в буферной памяти соответствующую пару. Если адрес есть в списке пар, то нет необходимости посылать ARP-запрос. Эта буферная память называется ARP-таблицей.

В ARP-таблице могут содержаться как **статические**, так и **динамические** записи. Динамические записи добавляются и удаляются автоматически, статические заносятся вручную. Так как большинство устройств в сети поддерживает динамическое разрешение адресов, то администратору, как правило, нет необходимости собственноручно указывать записи протокола ARP в таблице адресов.

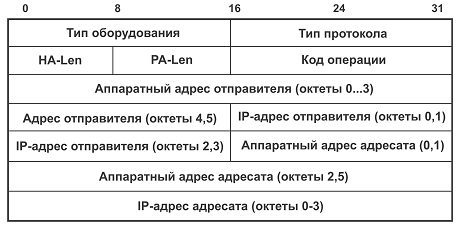
Кроме того, ARP-таблица всегда содержит запись с физическим широковещательным адресом (OxFFFFFFFFFFFF) для локальной сети. Эта запись позволяет устройству принимать широковещательные ARP-запросы. Каждая запись в ARP-таблице имеет свое время жизни, например для операционной системы Microsoft Windows 2000 оно составляет 10 минут.

При добавлении записи для нее активируется таймер. Если запись не востребована в первые две минуты, она удаляется. Если используется — будет существовать на протяжении 10 минут. В некоторых реализациях протокола ARP новый таймер устанавливается после каждого обращения к записи в ARP -таблице.

Сообщения протокола ARP при передаче по сети инкапсулируются в поле данных кадра. Они не содержат IP-заголовка. В отличие от сообщений большинства протоколов, сообщения ARP не имеют фиксированного формата заголовка. Это объясняется тем, что протокол был разработан таким образом, чтобы он был применим для разрешения адресов в различных сетях. Фактически протокол способен работать с произвольными физическими адресами и сетевыми протоколами.

Узел, которому нужно выполнить отображение IP-адреса на локальный адрес, формирует ARP запрос, вкладывает его в кадр протокола канального уровня, указывая в нем известный IP-адрес, и рассылает запрос широковещательно. Все узлы локальной сети получают ARP запрос и сравнивают указанный там IP-адрес с собственным. В случае их совпадения узел формирует ARP-ответ, в котором указывает свой IP-адрес и свой локальный адрес и отправляет его уже направленно, так как в ARP запросе отправитель указывает свой локальный адрес. ARP-запросы и ответы используют один и тот же формат пакета. Так как локальные адреса могут в различных типах сетей иметь различную длину, то формат пакета протокола ARP зависит от типа сети. На показана структура запросов и ответов ARP и RARP.

**Протокол RARP** - это протокол, решающий обратную задачу - нахождение IP-адреса по известному локальному адресу. Он называется реверсивный ARP - RARP (Reverse Address Resolution Protocol) и используется при старте бездисковых станций, не знающих в начальный момент своего IP-адреса, но знающих адрес своего сетевого адаптера. Reverse ARP (или обратное разрешение) работает аналогично протоколу ARP за исключением того, что в его задачи входит определение физического адреса по известному адресу сетевого уровня. Этот протокол требует наличия в сети сервера RARP, подключенного к тому же сегменту сети, что и интерфейс маршрутизатора. Наиболее часто протокол reverse ARP используется для запуска бездисковых рабочих станций.



Структура запросов и ответов ARP и RARP

**Разработка программы.**

В было разработано консольное приложение на языке Си которое может выполнять следующие действия:

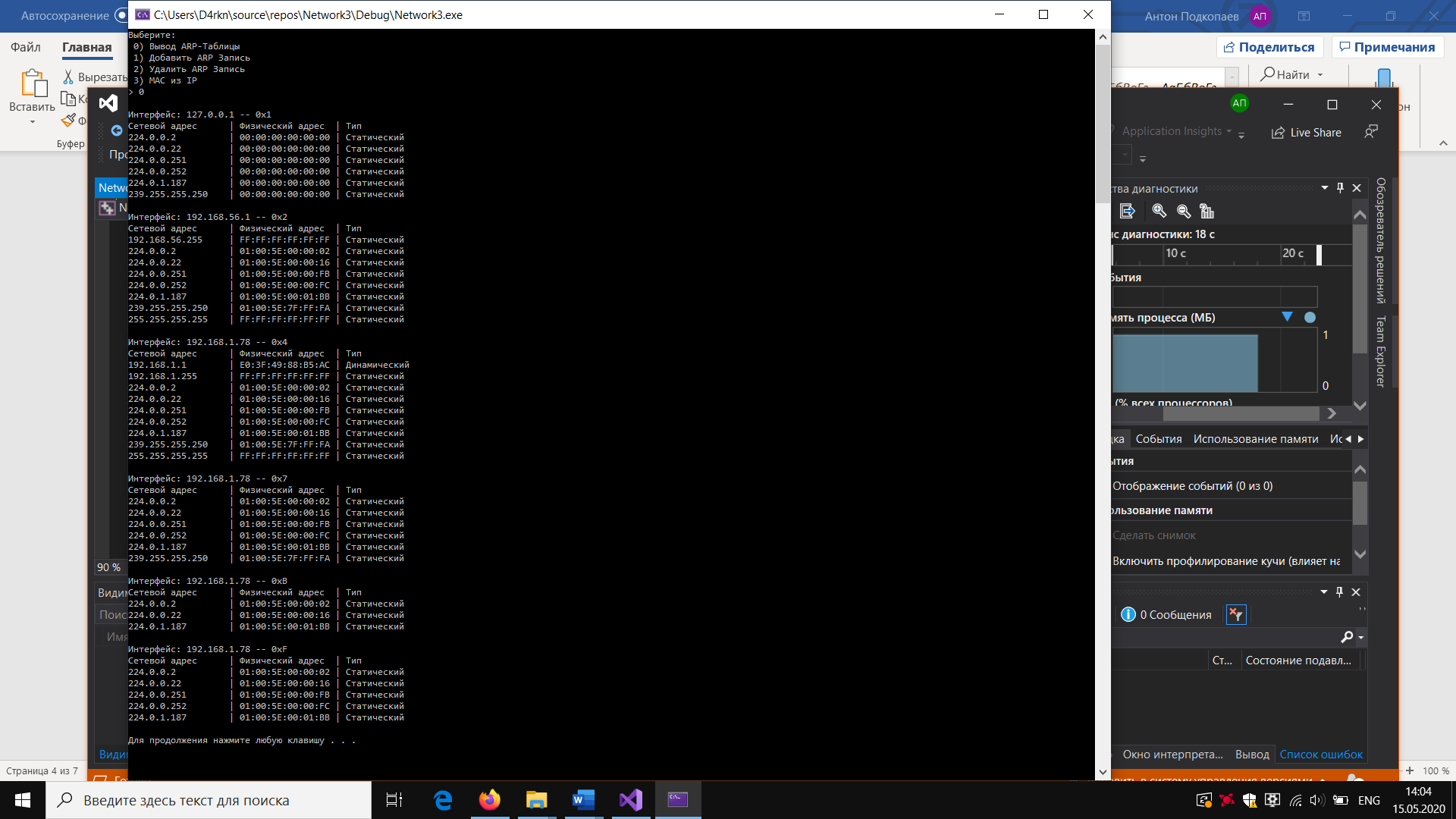
- вывод ARP-таблицы

- добавление записи в ARP-таблицу

- удаление записи из ARP-таблицы

- получение MAC-адреса по IP-адресу

**Результат работы программы**

**Изображение выглядит как снимок экрана, компьютер, монитор, ноутбук

Автоматически созданное описание**

**Выводы.**

В данной лабораторной работе была реализована программа для взаимодействия с ARP таблицей при помощи библиотек Winsock и IP Helper.

*Приложение*

#include <cstdlib>

#include <iostream>

#include <winsock.h>

#include <iphlpapi.h>

#define BUFFER\_SIZE 256

#pragma comment (lib, "iphlpapi.lib")

#pragma comment (lib, "wsock32.lib")

using namespace std;

unsigned long showARPTable() {

char type[BUFFER\_SIZE], address[BUFFER\_SIZE];

unsigned long realSize = 0;

PMIB\_IPADDRTABLE IPAddrTable = NULL;

PMIB\_IPNETTABLE IPARPTable = NULL;

GetIpNetTable(IPARPTable, &realSize, true);

IPARPTable = (PMIB\_IPNETTABLE)malloc(realSize);

if (GetIpNetTable(IPARPTable, &realSize, true) != NO\_ERROR) {

cout << "Ошибка при получении таблицы!\n" << endl;

if (IPARPTable) free(IPARPTable);

return 1;

}

realSize = 0;

GetIpAddrTable(IPAddrTable, &realSize, true);

IPAddrTable = (PMIB\_IPADDRTABLE)malloc(realSize);

GetIpAddrTable(IPAddrTable, &realSize, true);

unsigned long index = -100;

for (unsigned long i = 0; i < IPARPTable->dwNumEntries; i++) {

if (IPARPTable->table[i].dwIndex != index) {

index = IPARPTable->table[i].dwIndex;

IN\_ADDR inaddr;

for (unsigned long j = 0; j < IPAddrTable->dwNumEntries; j++) {

if (index != IPAddrTable->table[j].dwIndex) continue;

inaddr.S\_un.S\_addr = IPAddrTable->table[j].dwAddr;

strcpy\_s(address, inet\_ntoa(inaddr));

}

printf("\nИнтерфейс: %s -- 0x%X\nСетевой адрес | Физический адрес | Тип\n", address, index);

}

If (IPARPTable->table[i].dwType == 1) strcpy\_s(type, "Другой");

else if (IPARPTable->table[i].dwType == 2) strcpy\_s(type, "Неверный");

else if (IPARPTable->table[i].dwType == 3) strcpy\_s(type, "Динамический");

else if (IPARPTable->table[i].dwType == 4) strcpy\_s(type, "Статический");

else strcpy\_s(type, "");

IN\_ADDR inaddr;

inaddr.S\_un.S\_addr = IPARPTable->table[i].dwAddr;

printf("%-18s | %02X:%02X:%02X:%02X:%02X:%02X | %-11s\n",

inet\_ntoa(inaddr),

IPARPTable->table[i].bPhysAddr[0],

IPARPTable->table[i].bPhysAddr[1],

IPARPTable->table[i].bPhysAddr[2],

IPARPTable->table[i].bPhysAddr[3],

IPARPTable->table[i].bPhysAddr[4],

IPARPTable->table[i].bPhysAddr[5],

type);

}

free(IPARPTable);

puts("");

return 0;

}

unsigned long addARPRecord() {

char address[255], macaddr[255], interfaceindex[255];

MIB\_IPNETROW ARPRow;

cout << "Введите интерфейс : " ; cin >> interfaceindex;

cout << "Введите сетевой адрес : " ; cin >> address;

cout << "Введите mac : " ; cin >> macaddr;

unsigned long inetaddr = inet\_addr(address);

if (inetaddr == INADDR\_NONE) {

cout << "Неверный сетевой адрес!\n";

return 1;

}

sscanf\_s(interfaceindex, "%x", &(ARPRow.dwIndex));

ARPRow.dwPhysAddrLen = 6;

sscanf\_s(macaddr, "%hx:%hx:%hx:%hx:%hx:%hx",

&ARPRow.bPhysAddr[0],

&ARPRow.bPhysAddr[1],

&ARPRow.bPhysAddr[2],

&ARPRow.bPhysAddr[3],

&ARPRow.bPhysAddr[4],

&ARPRow.bPhysAddr[5]);

ARPRow.dwAddr = inetaddr;

ARPRow.dwType = MIB\_IPNET\_TYPE\_STATIC;

int err = CreateIpNetEntry(&ARPRow);

if (err == ERROR\_ACCESS\_DENIED) {

cout << "Доступ запрещен, запись не добавлена\n";

return 1;

}

else if (err == NO\_ERROR) cout << "Ок!\n";

else {

cout << "Что-то идет не так, запись не была добавлена\n";

return 1;

}

return 0;

}

unsigned long deleteARPRecord() {

MIB\_IPNETROW ARPRow;

char address[255], interfaceindex[255];

cout << "Введите интерфейс : "; cin >> interfaceindex;

cout << "Введите сетевой адрес : "; cin >> address;

unsigned long inetaddr = inet\_addr(address);

if (inetaddr == INADDR\_NONE) {

cout << "Wrong network address!" << endl;

return 1;

}

sscanf\_s(interfaceindex, "%x", &(ARPRow.dwIndex));

ARPRow.dwAddr = inetaddr;

if (DeleteIpNetEntry(&ARPRow) == ERROR\_ACCESS\_DENIED) {

cout << "Доступ запрещен, запись не добавлена\n";

return 1;

}

else if (CreateIpNetEntry(&ARPRow) == NO\_ERROR) cout << "Ок!\n";

else {

cout << "Что-то идет не так, запись не была добавлена\n";

return 1;

}

return 0;

}

void getIpFromMac() {

char address[255];

unsigned long realSize = 0, inetaddr;

PMIB\_IPNETTABLE IPAddrTable = NULL;

bool found = false;

GetIpNetTable(IPAddrTable, &realSize, true);

IPAddrTable = (PMIB\_IPNETTABLE)malloc(realSize);

GetIpNetTable(IPAddrTable, &realSize, true);

cout << "Введите сетевой адрес: "; cin >> address;

inetaddr = inet\_addr(address);

if (inetaddr == INADDR\_NONE) {

cout << "Неверный сетевой адрес!\n";

return;

}

for (unsigned long i = 0; i < IPAddrTable->dwNumEntries; i++)

if (inetaddr == IPAddrTable->table[i].dwAddr) {

printf("Mac: %02X-%02X-%02X-%02X-%02X-%02X на интерфейсе с индексом %x\n",

IPAddrTable->table[i].bPhysAddr[0],

IPAddrTable->table[i].bPhysAddr[1],

IPAddrTable->table[i].bPhysAddr[2],

IPAddrTable->table[i].bPhysAddr[3],

IPAddrTable->table[i].bPhysAddr[4],

IPAddrTable->table[i].bPhysAddr[5],

IPAddrTable->table[i].dwIndex);

found = true;

}

if (!found) cout << "Не найден!\n";

}

int main() {

int choice = 0;

setlocale(LC\_ALL, "Russian");

cout << "Выберите:\n 0) Вывод ARP-Таблицы\n 1) Добавить ARP Запись\n 2) Удалить ARP Запись\n 3) MAC из IP\n> "; cin >> choice;

if (choice == 0) showARPTable();

else if (choice == 1) addARPRecord();

else if (choice == 2) deleteARPRecord();

else if (choice == 3) getIpFromMac();

system("pause");

return 0;

}