МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им. В.Г.Шухова)

**Лабораторная работа №5**

дисциплина «Сети ЭВМ и телекоммуникации»

тема: «Протоколы ARP/RARP»

Выполнил: студент группы ВТ-31

Ковалёв И. Д.

Проверил: Федотов Е. А.

Белгород 2020

**Цель работы:** изучить протоколы ARP/RARP

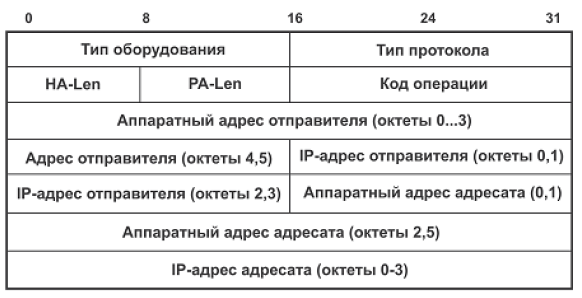
**Теоретические сведения**

**ARP (Address Resolution Protocol** - **протокол определения адреса)** - протокол канального уровня, предназначенный для определения MAC-адреса (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. Эта буферная память называется ARP-таблицей.

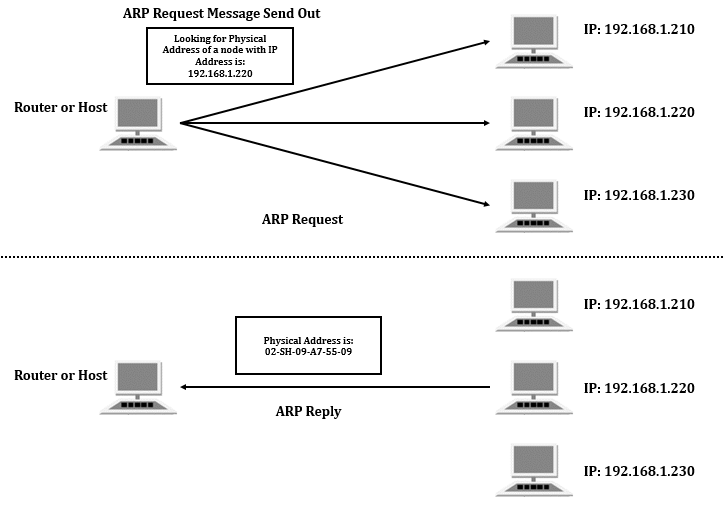
В ARP-таблице могут содержаться как статические, так и динамические записи. Динамические записи добавляются и удаляются автоматически, статические заносятся вручную. Так как большинство устройств в сети поддерживает динамическое разрешение адресов, то администратору, как правило, нет необходимости собственноручно 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. В некоторых реализациях протокола ARP новый таймер устанавливается после каждого обращения к записи в ARP –таблице.

**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:



Принцип работы ARP:

****

**Задание к работе**

1. Программно реализовать вывод ARP-таблицы.

2. Программно реализовать добавление записи в ARP-таблицу.

3. Программно реализовать удаление записи из ARP-таблицы.

4. Программно реализовать получение MAC-адреса по IP-адресу.

**Основные API-функции, использованные в данной работе:**

DWORD CreateIpNetEntry(PMIB\_IPNETROW pArpEntry)

Назначение: Создаёт запись протокола в таблице ARP на локальном компьютере.

DWORD CreateProxyArpEntry(DWORD dwAddress, DWORD dwMask, DWORD dwIfIndex)

Назначение: Создаёт запись PRAP на локальном компьютере.

DWORD DeleteIpNetEntry(PMIB\_IPNETROW pArpEntry)

Назначение: Удаляет запись PRAP на локальном компьютере.

ULONG GetIpNetTable(PMIB\_IPNETTABLE IpNetTable, PULONG SizePointer, BOOL Order)

Назначение: Извлекает таблицу сопоставления IPv4 с физическим адресом.

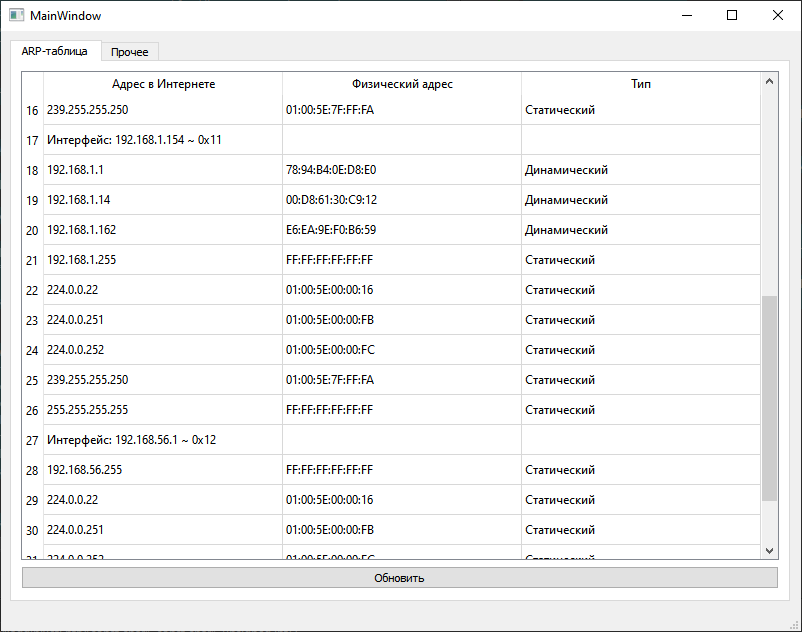
DWORD SetIpNetEntry(PMIB\_IPNETROW pArpEntry);

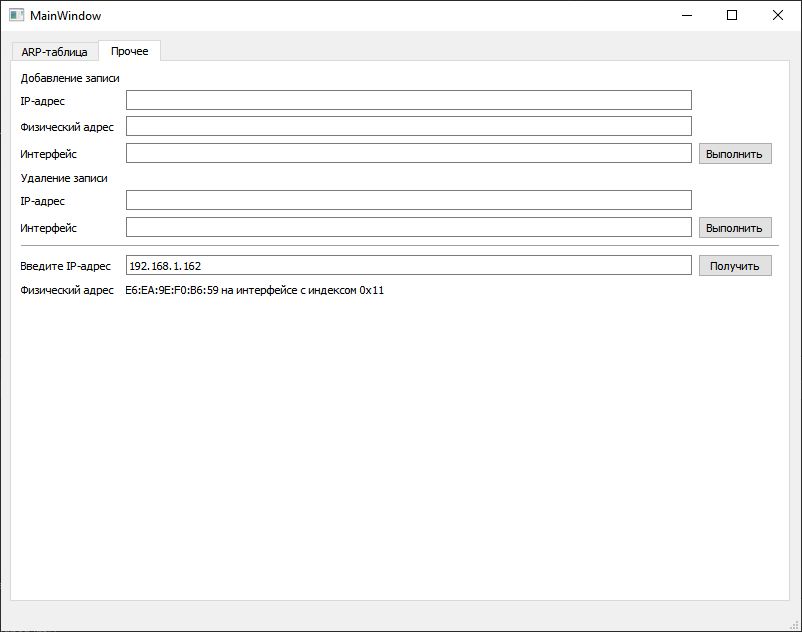
Назначение: Изменяет существующую запись в таблице ARP на локальном компьютер.

WORD SendARP(IPAddr DestIP, IPAddr SrcIP, PVOID pMacAddr, PULONG PhyAddrLen)

Назначение: Оправляет запрос для получения физ. адреса, соответствующего указанному IPv4-адресу назначения.

**Скриншоты разработанной программы**





**Вывод:** В ходе лабораторной работы были изучены протоколы ARP и RARP. Были использованы библиотеки Win Sockets и IPHelper. ARP-протокол предназначен для определения MAC-адреса по известному IP-адресу. Для того чтобы уменьшить количество посылаемых запросов ARP, каждое устройство в сети, использующее протокол ARP, имеет специальную буферную память (ARP-таблицу). В ней хранятся пары адресов (IP -адрес, физический адрес) устройств в сети.

**Контрольные вопросы**

1. Какие задачи решает протокол ARP?

ARP решает задачи по определению физического адреса по уже известному ip-адресу, а так же отвечает на запросы от остальных устройств в сети.

1. Что такое ARP-таблица? Почему она является необходимым элементом?

Это таблица, которая хранит в себе соответствие ip-адресов физическим. Нужна для уменьшения посылаемых в сеть запросов.

1. Типы записей ARP-таблицы.

Статические (Запись была внесена администратором сети).

Динамические (Запись была получена автоматически).

1. Опишите процесс преобразования ip-адреса в локальный.

Просматривается ARP-таблица. Если в ней не было найдено соответствия, узел должен сформировать ARP-запрос, в котором будет указан необходимый ip-адрес, и разослать его всем в сеть. В случае, если в сети найдется узел с таким ip-адресом, этот узел формирует ответ, в котором укажет свой физический адрес.

1. Что представляет собой протокол RARP?

Этот протокол выполняет обратную задачу: по физическому адресу определяется ip-адрес.

1. В каких целях может быть использован протокол RARP?

Данный протокол может использоваться, например, если в сети есть устройства, которые не имеют ПЗУ. В теории, производители оборудования задают уникальный физический адрес каждому выпущенному устройству, и в таком случае из этого физического адреса можно получить ip-адрес.

**Листинг программы**

Mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QMessageBox>

#include <cstdlib>

#include <winsock.h>

#include <iphlpapi.h>

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

void **getArpTable**();

void **getMac**(*const* char \* IP);

DWORD **addIpNetEntry**(*const* char \* ip, *const* char \* mac, unsigned iface);

DWORD **dropIpNetEntry**(*const* char \* ip, unsigned iface);

*private* slots:

void **on\_refreshArpTable\_clicked**();

void **on\_getMacFromIp\_clicked**();

void **on\_addEntry\_clicked**();

void **on\_deleteEntry\_clicked**();

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**Mainwindow.cpp:**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent):QMainWindow(parent), ui(*new* Ui::MainWindow){

ui->setupUi(*this*);

ui->arpTable->setColumnCount(3);

ui->arpTable->setHorizontalHeaderLabels({"Адрес в Интернете", "Физический адрес", "Тип"});

ui->arpTable->horizontalHeader()->setSectionResizeMode(QHeaderView::*Stretch*);

ui->arpTable->setRowCount(1);

ui->tabWidget->setTabText(0, "ARP-таблица");

getArpTable();

ui->tabWidget->setTabText(1, "Прочее");

}

MainWindow::~***MainWindow***(){

*delete* ui;

}

void MainWindow::**getArpTable**(){

int pos = 0;

PMIB\_IPNETTABLE IPARPTable = NULL;

DWORD ActualSize = 0;

GetIpNetTable(IPARPTable, &ActualSize, *true*);

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

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

*if* (IPARPTable)

free(IPARPTable);

*return*;

}

DWORD CurrentIndex;

char Address[256];

PMIB\_IPADDRTABLE IPAddressTable = NULL;

ActualSize = 0;

GetIpAddrTable(IPAddressTable, &ActualSize, *true*);

IPAddressTable = (PMIB\_IPADDRTABLE)malloc(ActualSize);

GetIpAddrTable(IPAddressTable, &ActualSize, *true*);

ui->arpTable->setRowCount(ActualSize);

CurrentIndex = -100;*//индекс* *адаптера*

*for* (DWORD i = 0; i < IPARPTable->dwNumEntries; i++) {

QString tmp;

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

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

IN\_ADDR InAddress;

*for* (DWORD j = 0; j < IPAddressTable->dwNumEntries; j++) {

*if* (CurrentIndex != IPAddressTable->table[j].dwIndex)

*continue*;

InAddress.S\_un.S\_addr = IPAddressTable->table[j].dwAddr;

strcpy(Address, inet\_ntoa(InAddress));

}

ui->arpTable->setItem(pos++, 0, *new* QTableWidgetItem(QString("Интерфейс: %1 ~ 0x%2")

.arg(Address)

.arg(CurrentIndex, 0, 16)

));

}

QString type = "";

*switch* (IPARPTable->table[i].dwType) {

*case* 1:

type = "Другой";

*break*;

*case* 2:

type = "Неправильный";

*break*;

*case* 3:

type = "Динамический";

*break*;

*case* 4:

type = "Статический";

*break*;

*default*:

type = "";

}

ui->arpTable->setItem(pos, 2, *new* QTableWidgetItem(type));

type = "";

IN\_ADDR InAddress;

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

ui->arpTable->setItem(pos, 0, *new* QTableWidgetItem(QString(inet\_ntoa(InAddress))));

QString macStr = "";

*for* (int k = 0; k < 6; k++){

*if* (IPARPTable->table[i].bPhysAddr[k] < 0x10){

macStr += QString("0%1").arg(IPARPTable->table[i].bPhysAddr[k], 0, 16).toUpper();

} *else* {

macStr += QString("%1").arg(IPARPTable->table[i].bPhysAddr[k], 0, 16).toUpper();

}

*if* (k != 5) macStr += ":";

}

ui->arpTable->setItem(pos, 1, *new* QTableWidgetItem(macStr));

macStr = "";

pos++;

}

ui->arpTable->setRowCount(pos);

free(IPARPTable);

}

void MainWindow::**getMac**(*const* char \*IP) {

DWORD ActualSize = 0;

PMIB\_IPNETTABLE IPAddressTable = NULL;

GetIpNetTable(IPAddressTable, &ActualSize, *true*);

IPAddressTable = (PMIB\_IPNETTABLE)malloc(ActualSize);

GetIpNetTable(IPAddressTable, &ActualSize, *true*);

DWORD InetAddress = inet\_addr(IP);

*if* (InetAddress == INADDR\_NONE) {

ui->macAddress->setText("Неправильный адрес!");

*return*;

}

bool SearchFlag = *true*;

*for* (DWORD i = 0; i < IPAddressTable->dwNumEntries; i++) {

*if* (InetAddress == IPAddressTable->table[i].dwAddr) {

QString macStr = "";

*for* (int k = 0; k < 6; k++){

*if* (IPAddressTable->table[i].bPhysAddr[k] < 0x10){

macStr += QString("0%1").arg(IPAddressTable->table[i].bPhysAddr[k], 0, 16).toUpper();

} *else* {

macStr += QString("%1").arg(IPAddressTable->table[i].bPhysAddr[k], 0, 16).toUpper();

}

*if* (k != 5) macStr += ":";

}

macStr += QString(" на интерфейсе с индексом 0x%1").arg(IPAddressTable->table[i].dwIndex, 0, 16);

ui->macAddress->setText(macStr);

SearchFlag = *false*;

}

}

*if* (SearchFlag) ui->macAddress->setText("Соответствий не найдено!");

}

DWORD MainWindow::**addIpNetEntry**(*const* char \* ip, *const* char \* mac, unsigned iface){

DWORD InetAddress = inet\_addr(ip);

*if* (InetAddress == INADDR\_NONE) {

QMessageBox::warning(*this*, "Ошибка!", "Введен неверный IP!");

*return* 1;

}

MIB\_IPNETROW ARPRow;

ARPRow.dwIndex = iface;

ARPRow.dwPhysAddrLen = 6;

ARPRow.dwAddr = InetAddress;

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

*switch* (CreateIpNetEntry(&ARPRow)) {

*case* ERROR\_ACCESS\_DENIED:

QMessageBox::warning(*this*, "Ошибка", "Запись не добавлена, доступ запрещен!");

*break*;

*case* NO\_ERROR:

QMessageBox::warning(*this*, "", "Запись добавлена!");

*break*;

*default*:

QMessageBox::warning(*this*, "Ошибка", "Запись не добавлена!");

}

*return* 0;

}

DWORD MainWindow::**dropIpNetEntry**(*const* char \* ip, unsigned iface){

DWORD InetAddress = inet\_addr(ip);

*if* (InetAddress == INADDR\_NONE){

QMessageBox::warning(*this*, "Ошибка!", "Введен неверный IP!");

*return* 1;

}

MIB\_IPNETROW ARPRow;

ARPRow.dwIndex = iface;

ARPRow.dwAddr = InetAddress;

*switch* (DeleteIpNetEntry(&ARPRow)) {

*case* ERROR\_ACCESS\_DENIED:

QMessageBox::warning(*this*, "Ошибка", "Запись не добавлена, доступ запрещен!");

*break*;

*case* NO\_ERROR:

QMessageBox::warning(*this*, "", "Запись добавлена!");

*break*;

*default*:

QMessageBox::warning(*this*, "Ошибка", "Запись не добавлена!");

}

*return* 0;

}

void MainWindow::**on\_refreshArpTable\_clicked**(){

ui->arpTable->clearContents();

getArpTable();

}

void MainWindow::**on\_getMacFromIp\_clicked**(){

QString IP = ui->ipAdress->text();

*if* (IP == "") *return*;

getMac(IP.toStdString().c\_str());

}

void MainWindow::**on\_addEntry\_clicked**(){

addIpNetEntry(

ui->ipEdit->text().toStdString().c\_str(),

ui->macEdit->text().toStdString().c\_str(),

ui->ifaceEdit->text().toUInt(0, 16)

);

}

void MainWindow::**on\_deleteEntry\_clicked**(){

dropIpNetEntry(

ui->ipDrop->text().toStdString().c\_str(),

ui->ifaceDrop->text().toUInt(0, 16) );}