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 телекомуникации»
по теме «Протоколы ARP/RARP»

Выполнил: студент группы ВТ-31 Макаров Д.С.
Проверил: Федотов Е.А.

Белгород 2020

Лабораторная работа №5

«Протоколы ARP/RARP»

Цель работы:изучить протоколы ARP/RARP.

Вариант 6

Содержание отчета

1. Краткие теоретические сведения.
2. Основные функции API, использованные в данной работе.
3. Разработка программы. Блок-схемы программы.
4. Анализ функционирования разработанных программ.
5. Выводы.
6. Тексты программ. Скриншоты программ.

Ход работы

1. **Краткие 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.

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. В данной лабораторной работе совместно с библиотекой Windows Sockets необходимо использовать библиотеку функций IP Helper.

1. **Основные функции API, использованные в данной работе.**

- `DWORD CreateIpNetEntry(PMIB_IPNETROW)`
- `DWORD CreateProxyArpEntry(DWORD,DWORD,DWORD)`
- `DWORD DeleteIpNetEntry(PMIB_IPNETROW)`
- `DWORD DeleteProxyArpEntry(DWORD,DWORD,DWORD)`
- `DWORD FlushIpNetTable(DWORD)`
- `DWORD SetIpNetEntry(PMIB_IPNETROW pArpEntry)`
- `DWORD SendARP(IPAddr,IPAddr,PVOID,PULONG)`
- `ULONG GetIpNetTable(PMIB_IPNETTABLE,PULONG,BOOL)`

2. **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

3. **Анализ функционирования разработанных программ.**

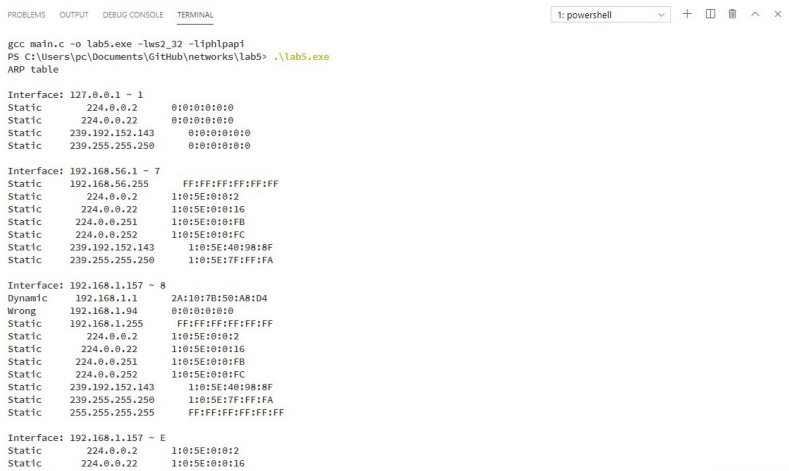

Рис. 1: Пример вывода ARP таблицы

4. **Выводы.**

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

5. **Тексты программ. Скриншоты программ.**

Тексты программ см. в приложении.

7. **Контрольные вопросы**

1. Какие задачи решает протокол ARP?

   Протокол ARP решает проблему преобразования IP-адреса в MAC-адрес.

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

   ARP таблица - таблица хранящаяя соответствия IP->MAC адресов. Необходима для уменьшения кол-ва запросов, если в таблице отсутствует запись только тогда будет выполнен запрос.

3. Типы записей ARP-таблицы.

   Статические и динамические. Статические создаются вручную администратором сети. Динамические создаются автоматически и имеют время жизни, которое продлевается если запись используется системой.

4. Опишите процесс преобразования ip-адреса в локальный.

   1. Просматривается ARP таблица на предмет нужного ip адреса.
   2. Если адрес отсутствует то формируется шировещательный ARP запрос.
   3. Каждый узел проверяет на соответствие свой ip адрес и адрес в пакете.
   4. Если адреса совпадают формируется ответный запрос на адрес отправителя широковещательного запроса.

5. Как может работать протокол ARP в глобальных сетях?

   В глобальных сетях администратору сети приходится вручную формировать ARP-таблицы, в которых он задает, соответствие IP-адреса адресу узла сети X.25, который имеет для протокола IP смысл локального адреса.

6. Что представляет собой протокол RARP?

   RARP - протокол обратный ARP. Выполняет задачу нахождения IP адреса по MAC адресу.

7. В каких целях может быть использован протокол RARP?

   В бездисковые машины не могут сохранить свой сконфигурированный IP адрес. RARP позволяет использовать уникальный для каждого устройства MAC адрес как идентификатор по которому можно будет определить IP адрес.

Приложение

Содержимое файла arpWrapper.h

```
#ifndef _ARP_WRAPPER_H
#define _ARP_WRAPPER_H

#include <stdlib.h>
#include <stdio.h>
#include <winsock.h>
#include <iphlpapi.h>

void print_arp_table(){
    int pos = 0;
    PMIB_IPNETTABLE IPARPTable = NULL;
    DWORD ActualSize = 0;
    GetIpNetTable(IPARPTable, &ActualSize, 1);
    IPARPTable = (PMIB_IPNETTABLE)malloc(ActualSize);
    if (GetIpNetTable(IPARPTable, &ActualSize, 1) != NO_ERROR) {
        if (IPARPTable)
            free(IPARPTable);
        return;
    }
    DWORD CurrentIndex;
    char Address[256];
    PMIB_IPADDRTABLE IPAddressTable = NULL;
    ActualSize = 0;
    GetIpAddrTable(IPAddressTable, &ActualSize, 1);
    IPAddressTable = (PMIB_IPADDRTABLE)malloc(ActualSize);
    GetIpAddrTable(IPAddressTable, &ActualSize, 1);
    CurrentIndex = -100;//индекс адаптера
    printf("ARP table\n");
    for (DWORD i = 0; i < IPARPTable->dwNumEntries; i++) {
        char if_str[255];
        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));
            }
            printf("\nInterface: %s ~ %X\n",Address,CurrentIndex);
        }
        switch (IPARPTable->table[i].dwType) {
            case 1:
                printf("Other      ");
            break;
            case 2:
                printf("Wrong      ");
            break;
            case 3:
                printf("Dynamic    ");
            break;
            case 4:
                printf("Static     ");
            break;
            default:
                printf("None       ");
```

```
        }
        IN_ADDR InAddress;
        InAddress.S_un.S_addr = IPARPTable->table[i].dwAddr;
        printf("%12s      ",inet_ntoa(InAddress));
        for (int k = 0; k < 6; k++){
            if (IPARPTable->table[i].bPhysAddr[k] < 0x10){
                printf("%X",IPARPTable->table[i].bPhysAddr[k]);
            } else {
                printf("%X",IPARPTable->table[i].bPhysAddr[k]);
            }
            if (k != 5) printf(":");
        }
        pos++;
        printf("\n");
    }
};

void getMac(const char *IP) {
    DWORD ActualSize = 0;
    PMIB_IPNETTABLE IPAddressTable = NULL;
    GetIpNetTable(IPAddressTable, &ActualSize, 1);
    IPAddressTable = (PMIB_IPNETTABLE)malloc(ActualSize);
    GetIpNetTable(IPAddressTable, &ActualSize, 1);
    DWORD InetAddress = inet_addr(IP);
    if (InetAddress == INADDR_NONE) {
        printf("Wrong IP adress\n");
        return;
    }
    char SearchFlag = 1;
    for (DWORD i = 0; i < IPAddressTable->dwNumEntries; i++) {
        if (InetAddress == IPAddressTable->table[i].dwAddr) {
            printf("MAC: \n");
            for (int k = 0; k < 6; k++){
                if (IPAddressTable->table[i].bPhysAddr[k] < 0x10){
                    printf("%X",IPAddressTable->table[i].bPhysAddr[k]);
                } else {
                    printf("%X",IPAddressTable->table[i].bPhysAddr[k]);
                }
                if (k != 5) printf(":");
            }
            printf(" on iface - %h\n",IPAddressTable->table[i].dwIndex);
            SearchFlag = 0;
        }
    }
    if (SearchFlag) printf("MAC doesn't exist\n");
};

DWORD addIpNetEntry(const char * ip, const char * mac, unsigned iface){
    DWORD InetAddress = inet_addr(ip);
    if (InetAddress == INADDR_NONE) {
        printf("Wrong IP adress\n");
        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:
```

```
        printf("ACCESS DENIED\n");
        break;
    case NO_ERROR:
        printf("Record created\n");
        break;
    default:
        printf("Record doesn't created\n");
    }
    return 0;
};

DWORD dropIpNetEntry(const char * ip, unsigned iface){
    DWORD InetAddress = inet_addr(ip);
    if (InetAddress == INADDR_NONE){
        printf("Wrong IP adress\n");
        return 1;
    }
    MIB_IPNETROW ARPRow;
    ARPRow.dwIndex = iface;
    ARPRow.dwAddr = InetAddress;
    switch (DeleteIpNetEntry(&ARPRow)) {
        case ERROR_ACCESS_DENIED:
        printf("ACCESS DENIED\n");
        break;
         case NO_ERROR:
        printf("Record created\n");
        break;
         default:
        printf("Record doesn't created\n");
    }
};

#endif
```

Содержимое файла main.c

```
#include <stdlib.h>
#include <stdio.h>
#include "arp_wrapper.h"

int main(){
    print_arp_table();
    getMac("239.192.152.143");
    return 0;
}
```

Содержимое файла makefile

```
all: build

build:
        gcc main.c -o lab5.exe -lws2_32 -liphlpapi
```