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 №6**

дисциплина «Сети ЭВМ и телекоммуникации»

тема: «Протоколы DHCP и DNS»

Выполнил: студент группы ВТ-31

Ковалёв И. Д.

Проверил: Федотов Е. А.

Белгород 2020

**Цель работы**: изучить протоколы DHCP, DNS и составить программы согласно заданию.

**Теоретические сведения**

**Протокол DHCP**

DHCP (англ. Dynamic Host Configuration Protocol — протокол динамической конфигурации узла) - это сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер». Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP, и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок. Протокол DHCP используется в большинстве сетей TCP/IP.

**Распределение IP-адресов**

1. Ручное распределение. При этом способе сетевой администратор сопоставляет аппаратному адресу (для Ethernet сетей это MAC-адрес) каждого клиентского компьютера определённый IP-адрес.

2. Автоматическое распределение. При данном способе каждому компьютеру на постоянное использование выделяется произвольный свободный IP-адрес из определённого администратором диапазона.

3. Динамическое распределение. Этот способ аналогичен автоматическому распределению, за исключением того, что адрес выдаётся компьютеру не на постоянное пользование, а на определённый срок.

Помимо IP-адреса, DHCP также может сообщать клиенту дополнительные параметры, необходимые для нормальной работы в сети. Эти параметры называются опциями DHCP. Список стандартных опций можно найти в RFC 2132.

Пример процесса получения IP-адреса клиентом от сервера DHCP:

1. Обнаружение DHCP.

Вначале клиент выполняет широковещательный запрос по всей физической сети с целью обнаружить доступные DHCP-серверы. Он отправляет сообщение типа DHCPDISCOVER, при этом в качестве IP-адреса источника указывается 0.0.0.0 (так как компьютер ещё не имеет собственного IP-адреса), а в качестве адреса назначения - широковещательный адрес 255.255.255.255. Клиент заполняет несколько полей сообщения начальными значениями.

2. Предложение DHCP.

Получив сообщение от клиента, сервер определяет требуемую конфигурацию клиента в соответствии с указанными сетевым администратором настройками. Пусть в данном случае DHCP-сервер согласен с запрошенным клиентом адресом 192.168.1.100. Сервер отправляет ему ответ (DHCPOFFER), в котором предлагает конфигурацию. Предлагаемый клиенту IP-адрес указывается в поле yiaddr. Прочие параметры указываются в виде опций в соответствующем поле. Далее это сообщение DHCP-сервер отправляет хосту, пославшему DHCPDISCOVER, на его MAC, при определенных обстоятельствах сообщение может распространяться как широковещательная рассылка. Клиент может получить несколько различных предложений DHCP от разных серверов; из них он должен выбрать то, которое его «устраивает».

3. Запрос DHCP.

Выбрав одну из конфигураций, предложенных DHCP-серверами, клиент отправляет запрос DHCP (DHCPREQUEST). Он рассылается широковещательно; при этом к опциям, указанным клиентом в сообщении DHCPDISCOVER, добавляется специальная опция – идентификатор сервера - указывающая адрес DHCP-сервера, выбранного клиентом (в данном случае - 192.168.1.1).

4. Подтверждение DHCP.

Наконец, сервер подтверждает запрос и направляет это подтверждение (DHCPACK) клиенту. После этого клиент должен настроить свой сетевой интерфейс, используя предоставленные опции.

**Протокол DNS**

В стеке TCP/IP применяется доменная система имен, которая имеет иерархическую древовидную структуру. Иерархия доменных имен аналогична иерархии имен файлов, принятой в файловых системах. В отличие от имен файлов запись доменного имени начинается с самой младшей составляющей, и заканчивается самой старшей. Составные части доменного имени отделяются друг от друга точкой. Разделение имени на части позволяет разделить административную ответственность за назначение уникальных имен между различными людьми или организациями в пределах своего уровня иерархии. Разделение административной ответственности позволяет решить проблему образования уникальных имен без взаимных консультаций между организациями, отвечающими за имена одного уровня иерархии. Поэтому должна существовать одна организация, отвечающая за назначение имен верхнего уровня иерархии. Совокупность имен, у которых несколько старших составных частей совпадают, образуют домен (domain) имен.

В настоящее время используется масштабируемая служба для разрешения имен — система доменных имен (Domain Name System, DNS). Служба DNS использует в своей работе протокол типа «клиент-сервер». В нем определены DNS-серверы и DNS-клиенты. DNS-серверы поддерживают распределенную базу отображений, а DNS-клиенты обращаются к серверам с запросами о разрешении доменного имени в IP-адрес.

Для каждого домена имен создается свой DNS-сервер. Обычно сервер домена хранит только имена, которые заканчиваются на следующем ниже уровне иерархии по сравнению с именем домена. Именно при такой организации службы DNS нагрузка по разрешению имен распределяется более-менее равномерно между всеми DNS- серверами сети. Каждый DNS-сервер кроме таблицы отображений имен содержит ссылки на DNS-серверы своих поддоменов. Эти ссылки связывают отдельные DNS-серверы в единую службу DNS. Ссылки представляют собой IP-адреса соответствующих серверов. Для обслуживания корневого домена выделено несколько дублирующих друг друга DNS-серверов, IP-адреса которых являются широко известными (их можно узнать в InterNIC).

**Задание к работе**

1. Разработать программу, позволяющую по доменному имени узнать IP-адрес узла, а по IP-адресу узнать доменное имя узла.

2. Разработать программу, которая выводит информацию о DHCP–сервере и позволяет сбросить и запросить IP-адрес у этого сервера.

3. Программы должны быть написаны на языке программирования Pascal или C.

**Основные API-функции, использованные в процессе разработки программы:**

INT getnameinfo(SOCKADDR \*pSockaddr, socklen\_t SockaddrLength, PCHAR pNodeBuffer, DWORD NodeBufferSize, PCHAR pServiceBuffer, DWORD ServiceBufferSize, INT Flags);

**Назначение:** позволяет получить IP-адрес по доменному имени.

INT getaddrinfo(PCSTR pNodeName, PCSTR pServiceName, const ADDRINFOA \*pHints, PADDRINFOA \*ppResult)

**Назначение:** позволяет получить доменное имя по IP-адресу.

WORD GetInterfaceInfo(PIP\_INTERFACE\_INFO pIfTable, PULONG dwOutBufLen)

**Назначение:** Получает список сетевых адаптеров.

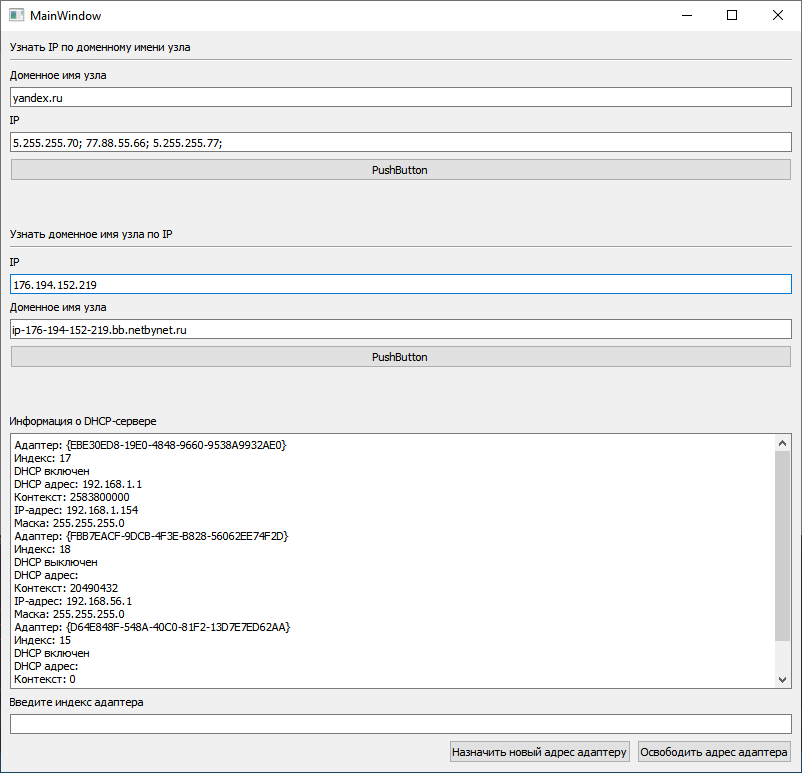
DWORD IpReleaseAddress( PIP\_ADAPTER\_INDEX\_MAP AdapterInfo)

**Назначение:** «отпускает» IP-адрес, выданный DHCP.

DWORD IpRenewAddress( PIP\_ADAPTER\_INDEX\_MAP AdapterInfo)

**Назначение**: возобновляет использование адреса, полученного от DHCP.

**Скриншоты разработанной программы**



**Вывод:** В ходе лабораторной работы были изучены протоколы DHCP, DNS и составлена программа согласно заданию.

DNS используется для получения IP-адреса по имени хоста. Данный протокол важен для функционирования интернета в целом, так как для соединения с узлом необходима информация об его IP-адресе, но для человека проще и удобнее запомнить буквенные адреса, чем последовательность чисел из IP.

DHCP – протокол, отвечающий за автоматическую выдачу сетевым устройствам IP-адреса и другие параметры, необходимые для работы в сети. Работает на модели «клиент-сервер».

**Контрольные вопросы**

1. Что представляет собой протокол DHCP?
2. Способы распределения IP-адресов
3. Опишите процесс получения клиентом IP-адреса от DHCP-сервера
4. В чём состоит суть доменной системы имён?
5. Типы доменных имён.
6. Как происходит управление доменами?
7. Состав службы DNS.
8. Опишите процесс разрешения доменного имени по двум существующим схемам.

1 . DHCP – протокол, позволяющий ПК автоматически получать IP-адрес и другие параметры, которые нужны для работы в сетях TCP/IP. Работает по модели «клиент-сервер». Для автоматической настройки клиент на этапе конфигурации обращается к серверу DHCP, и получает от него необходимые параметры. При этом сетевой администратор имеет возможность задать диапазон адресов, которые будут распределены сервером среди ПК, что позволяет избежать длительной ручной настройки компьютеров. Используется в большинстве сетей TCP/IP.

2 . Существует три вида распределения IP-адресов:

Ручное – администратор самостоятельно назначает адрес каждой машине в сети.

Автоматическое – каждый компьютер получает неиспользуемый в текущий момент времени IP-адрес автоматически.

Динамическое – адреса выдаются автоматически на некоторое время, а после окончания этого срока, адрес освобождается.

3. Клиент посылает запрос DHCPDISCOVER, который содержит физический адрес машины и уникальный идентификатор транзакции. DHCP сервер, получив это сообщение, отсылает DHCPOFFER с предлагаемой конфигурацией. Клиент, выбрав устраивающую его конфигурацию из предложенных, отправляет DHCPREQUEST, который содержит идентификатор выбранного клиентом конфига, на что сервер отвечает подтверждением DHCPACK.

4. Доменная система имен позволяет преобразовывать доменное имя в IP-адрес. Доменная система имен является иерархической и состоит из полного пути от узла к корню дерева доменных имен.

5. Доменные имена могут быть краткими,, относительными и полными.

Краткое имя – имя конечного узла сети.

Относительное имя – имя, начинающееся с какого-нибудь уровня иерархии.

Полное имя – содержит весь путь от узла к корню.

6. Корневой домен управляется центральными органами интернета: IANA, InterNIC. Данные организации распределяют домены верхнего уровня, в т.ч. национальные.

Древообразная структура доменной системы имен дает возможность делегировать управление доменами более низких уровней другим организациям.

7. Служба DNS состоит из серверов и клиентов. Сервера поддерживают базу отображений имен и адресов, а клиенты обращаются к серверам для разрешения имени в адрес.

Каждый сервер при этом полагается в разрешении имен на известные ему имена, а также на вышестоящие серверы.

8 . В итеративной схеме разрешения DNS-имен клиент обращается к серверам DNS последовательным образом, начиная от корневого. Такая схема на данный момент используется крайне редко.

Так же, существует рекурсивная схема, согласно которой клиент, обращаясь к локальному серверу DNS, требует ответ, который сервер, в случае, если данный сервер знает ответ, отправляет его клиенту. Иначе, локальный сервер обращается за необходимой информацией к вышестоящему DNS-серверу и так далее, пока ответ не будет получен.

**Листинг программы**

Mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QDebug>

#include <windows.h>

#include <winsock2.h>

#include <iphlpapi.h>

#include <stdio.h>

#include <ws2tcpip.h>

QT\_BEGIN\_NAMESPACE

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

QT\_END\_NAMESPACE

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

{

Q\_OBJECT

*public*:

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

~***MainWindow***();

QString **getIp**(QString name);

QString **getName**(QString ip);

void **viewIpInfo**();

void **changeIp**(int index, bool toRenew);

*private* slots:

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

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

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

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

*private*:

Ui::MainWindow \*ui;

QString text;

WSADATA wsaData;

*struct* addrinfo \*result = NULL;

*struct* addrinfo \*ptr = NULL;

*struct* addrinfo hints;

*struct* sockaddr\_in \*sockaddr\_ipv4;

*struct* sockaddr\_in saGNI;

LPSOCKADDR sockaddr\_ip;

char hostname[NI\_MAXHOST];

char servInfo[NI\_MAXSERV];

char ip[NI\_MAXHOST];

};

#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*);

text = "";

viewIpInfo();

*if* (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

*return*;

}

ZeroMemory(&hints, *sizeof*(hints));

hints.ai\_family = AF\_UNSPEC;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

}

MainWindow::~MainWindow(){

WSACleanup();

*delete* ui;

}

void MainWindow::viewIpInfo(){

PIP\_ADAPTER\_INFO AdapterInfo, Adapter;

ULONG ValueAdapterInfo;

int Error;

ValueAdapterInfo = 0;

Error = GetAdaptersInfo(NULL, &ValueAdapterInfo);

*if* ((Error != 0) && (Error != ERROR\_BUFFER\_OVERFLOW)){

*return*;

}

*if* ((AdapterInfo = (PIP\_ADAPTER\_INFO)GlobalAlloc(GPTR, ValueAdapterInfo)) == NULL){

*return*;

}

*if* (GetAdaptersInfo(AdapterInfo, &ValueAdapterInfo) != 0){

*return*;

}

Adapter = AdapterInfo;

*while* (Adapter){

text += "Адаптер: " + QString::fromStdString(Adapter->AdapterName) + "\n";

text += "Индекс: " + QString::number(Adapter->Index) + "\n";

*if* (Adapter->DhcpEnabled) text += "DHCP включен\n"; *else* text += "DHCP выключен\n";

text += "DHCP адрес: " + QString::fromStdString(Adapter->DhcpServer.IpAddress.String) + "\n";

PIP\_ADDR\_STRING Address = &(Adapter->IpAddressList);

*while* (Address){

text += "Контекст: " + QString::number(Address->Context) + "\n";

text += "IP-адрес: " + QString::fromStdString(Address->IpAddress.String) + "\n";

text += "Маска: " + QString::fromStdString(Address->IpMask.String) + "\n";

Address = Address->Next;

}

Adapter = Adapter->Next;

}

ui->textEdit->setText(text);

}

void MainWindow::changeIp(int index, bool toRenew){

DWORD InterfaceInfoSize = 0;

PIP\_INTERFACE\_INFO InterfaceInfo;

*if* (GetInterfaceInfo(NULL, &InterfaceInfoSize) != ERROR\_INSUFFICIENT\_BUFFER){

*return*;

}

*if* ((InterfaceInfo = (PIP\_INTERFACE\_INFO)GlobalAlloc(GPTR, InterfaceInfoSize)) == NULL){

*return*;

}

*if* (GetInterfaceInfo(InterfaceInfo, &InterfaceInfoSize) != 0){

*return*;

}

*for* (int i = 0; i < InterfaceInfo->NumAdapters; ++i) {

*if* (index == InterfaceInfo->Adapter[i].Index){

*if* (toRenew) IpRenewAddress(&InterfaceInfo->Adapter[i]);

*else* IpReleaseAddress(&InterfaceInfo->Adapter[i]);

}

}

}

QString MainWindow::getIp(QString name){

QString res = "";

getaddrinfo(name.toStdString().c\_str(), 0, &hints, &result);

*for* (ptr = result; ptr != NULL; ptr = ptr->ai\_next) {

*if* (ptr->ai\_family == AF\_INET) {

sockaddr\_ipv4 = (*struct* sockaddr\_in \*) ptr->ai\_addr;

res += QString::fromStdString(inet\_ntoa(sockaddr\_ipv4->sin\_addr)) + "; ";

}

}

*return* res;

}

QString MainWindow::getName(QString ip){

saGNI.sin\_family = AF\_INET;

saGNI.sin\_addr.s\_addr = inet\_addr(ip.toStdString().c\_str());

saGNI.sin\_port = htons(27015);

getnameinfo((*struct* sockaddr \*) &saGNI, *sizeof*(*struct* sockaddr), hostname, NI\_MAXHOST, servInfo, NI\_MAXSERV, NI\_NUMERICSERV);

*return* QString(hostname);

}

void MainWindow::on\_dToIpGet\_clicked(){

ui->dToIpResult->setText(getIp(ui->dToIpEdit->text()));

}

void MainWindow::on\_ipToDGet\_clicked(){

ui->ipToDResult->setText(getName(ui->ipToDEdit->text()));

}

void MainWindow::on\_renewButton\_clicked(){

changeIp(ui->adapter->text().toInt(), *true*);

}

void MainWindow::on\_releaseButton\_clicked(){

changeIp(ui->adapter->text().toInt(), *false*);

}