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)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №6

по дисциплине: Компьютерные сети
тема: «Протоколы DHCP и DNS»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
Рубцов Константин Анатольевич

Белгород 2025 г.

Лабораторная работа №6
Протоколы DHCP и DNS
Вариант 6

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

Краткие 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

Протокол DHCP предоставляет три способа распределения IPадресов:

1. *Ручное распределение*. При этом способе сетевой администратор сопоставляет аппаратному адресу (для Ethernet сетей это MACадрес) каждого клиентского компьютера определённый IP-адрес. Фактически, данный способ распределения адресов отличается от ручной настройки каждого компьютера лишь тем, что сведения об адресах хранятся централизованно (на сервере DHCP), и потому их проще изменять при необходимости.

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. Это называется арендой адреса. По истечении срока аренды IP-адрес вновь считается свободным, и клиент обязан запросить новый. Кроме того, клиент сам может отказаться от полученного адреса.

Некоторые реализации службы DHCP способны автоматически обновлять записи DNS, соответствующие клиентским компьютерам, при выделении им новых адресов. Это производится при помощи протокола обновления DNS, описанного в RFC 2136.

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.

Некоторыми из наиболее часто используемых опций являются:

- IP-адрес маршрутизатора по умолчанию
- маска подсети
- адреса серверов DNS
- имя домена DNS.

Протокол DHCP является клиент-серверным, то есть в его работе участвуют клиент DHCP и сервер DHCP. Передача данных производится при помощи протокола UDP, при этом сервер принимает сообщения от клиентов на порт 67 и отправляет сообщения клиентам на порт 68.

Все сообщения протокола DHCP разбиваются на поля, каждое из которых содержит определённую информацию. Все поля, кроме последнего (поля опций DHCP), имеют фиксированную длину.

Рассмотрим пример процесса получения IP-адреса клиентом от сервера DHCP. Предположим, клиент ещё не имеет собственного IP-адреса, но ему известен его предыдущий адрес — 192.168.1.100. Процесс состоит из четырёх этапов.

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 IPадреса 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:

   - В поле **xid** помещается уникальный *идентификатор транзакции*, который позволяет отличать данный процесс получения IP-адреса от других, протекающих в то же время.
   - В поле **chaddr** помещается аппаратный адрес (MAC-адрес) клиента.
   - В поле опций указывается последний известный клиенту IP-адрес. В данном примере это 192.168.1.100. Это необязательно и может быть проигнорировано сервером.

   Сообщение DHCPDISCOVER может быть распространено за пределы локальной физической сети при помощи специально настроенных агентов ретрансляции DHCP, перенаправляющих поступающие от клиентов сообщения DHCP серверам в других подсетях.

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 (такие, как адреса маршрутизаторов и DNS-серверов) 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.

Если один домен входит в другой домен как его составная часть, то такой домен могут называть поддоменом (subdomain). Обычно поддомен называют по имени той его старшей составляющей, которая отличает его от других поддоменов. Имя поддомену назначает администратор вышестоящего домена. Хорошей аналогией домена является каталог файловой системы.

По аналогии с файловой системой в доменной системе имен различают краткие имена, относительные имена и полные доменные имена. Краткое имя - это имя конечного узла сети. Относительное имя - это составное имя, начинающееся с некоторого уровня иерархии, но не самого верхнего. Например, wwwl.zil — это относительное имя. Полное доменное имя (fully qualified domain name, FQDN) включает составляющие всех уровней иерархии, начиная от краткого имени и кончая корневой точкой.

Корневой домен управляется центральными органами Интернета: IANA и InterNIC. Домены верхнего уровня назначаются для каждой страны, а также на организационной основе. Имена этих доменов должны следовать международному стандарту ISO 3166. Для обозначения стран используются трехбуквенные и двухбуквенные аббревиатуры, например, ru (Россия), uk (Великобритания), fin (Финляндия), us (Соединенные Штаты), а для различных типов организаций - следующие обозначения: com - коммерческие организации; edu - образовательные организации; gov - правительственные организации; org - некоммерческие организации; net - организации поддержки сетей.

Каждый домен администрируется отдельной организацией, которая обычно разбивает свой домен на поддомены и передает функции администрирования этих поддоменов другим организациям. Чтобы получить доменное имя, необходимо зарегистрироваться в какой-либо организации, которой организация InterNIC делегировала свои полномочия по распределению имен доменов. В России такой организацией является РосНИИРОС, которая отвечает за делегирование имен поддоменов в домене ru.

Доменная система имен реализована в Интернете, но она может работать и как автономная система имен в любой крупной корпоративной сети, которая также использует стек TCP/IP, но не связана с Интернетом.

Соответствие между доменными именами и IP-адресами может устанавливаться как средствами локального хоста, так и средствами централизованной службы. На раннем этапе развития Интернета на каждом хосте вручную создавался текстовый файл с известным именем hosts.txt. Этот файл состоял из некоторого количества строк, каждая из которых содержала одну пару «IP-адрес — доменное имя».

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

Служба DNS использует текстовые файлы, которые администратор подготавливает вручную. Однако служба DNS опирается на иерархию доменов, и каждый сервер службы DNS хранит только часть имен сети, а не все имена. При росте количества узлов в сети проблема масштабирования решается созданием новых доменов и поддоменов имен и добавлением в службу DNS новых серверов.

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).

Процедура разрешения DNS-имени во многом аналогична процедуре поиска файловой системой адреса файла по его символьному имени. Для определения IP-адреса по доменному имени необходимо просмотреть все DNS-серверы, обслуживающие цепочку поддоменов, входящих в имя хоста, начиная с корневого домена. Существенным отличием является то, что файловая система расположена на одном компьютере, а служба DNS по своей природе является распределенной.

Существует две основные схемы разрешения DNS-имен. В первом варианте работу по поиску IP-адреса координирует DNS-клиент, последовательно обращаясь к DNS-серверам, начиная с корневого. Такая схема взаимодействия называется нерекурсивной, или итеративной. Так как эта схема загружает клиента достаточно сложной работой, то она применяется редко. Во втором варианте DNS-клиент запрашивает локальный DNS-сервер, обслуживающий поддомен, к которому принадлежит имя клиента. Если локальный DNS-сервер знает ответ, то он сразу же возвращает его клиенту. Это может соответствовать случаю, когда запрошенное имя входит в тот же поддомен, что и имя клиента, а также случаю, когда сервер уже узнавал данное соответствие для другого клиента и сохранил его в своем кэше. Если локальный сервер не знает ответ, то он выполняет итеративные запросы к корневому серверу и к нижним по иерархии. Получив ответ, он передает его клиенту. Такая схема называется косвенной, или рекурсивной.

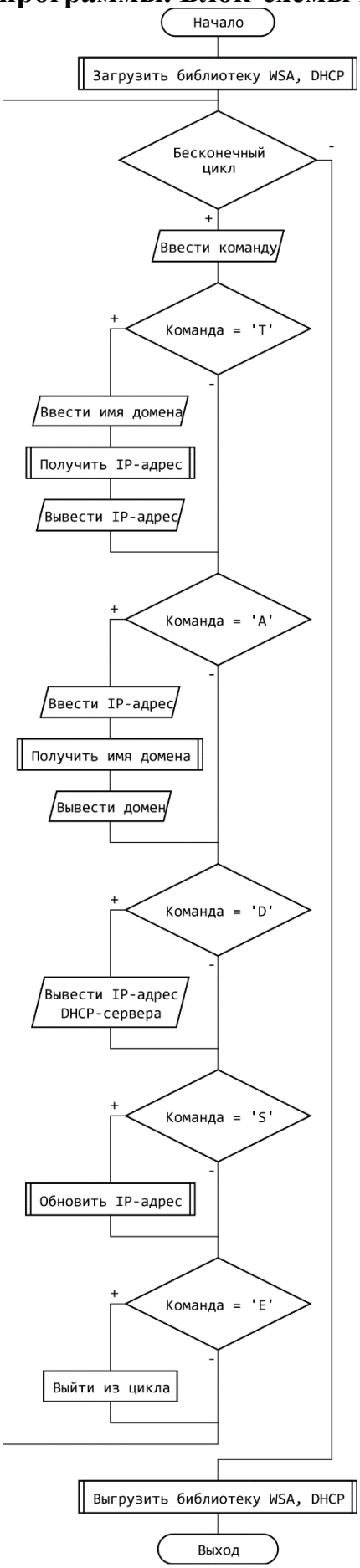
Для ускорения поиска IP-адресов DNS-серверы широко применяют процедуру кэширования проходящих через них ответов. Чтобы служба DNS могла оперативно отрабатывать изменения, происходящие в сети, ответы кэшируются на определенное время — обычно от нескольких часов до нескольких дней.

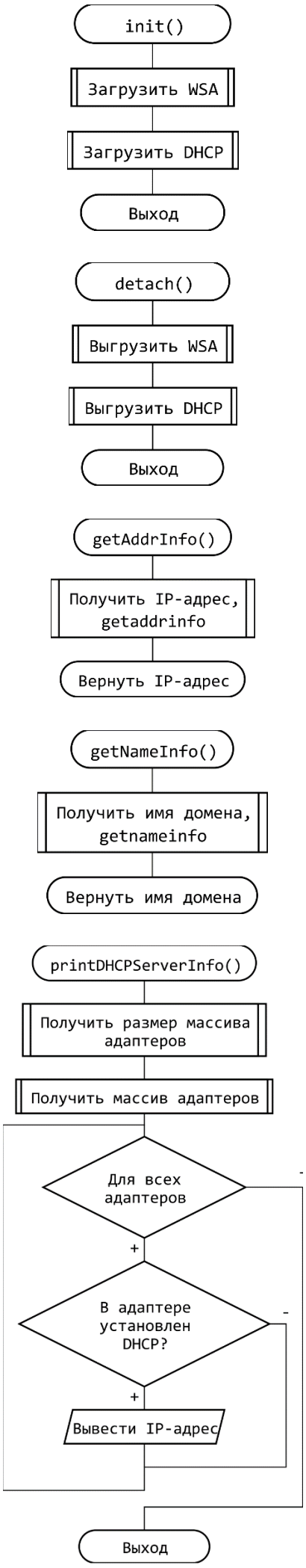
Используемые функции

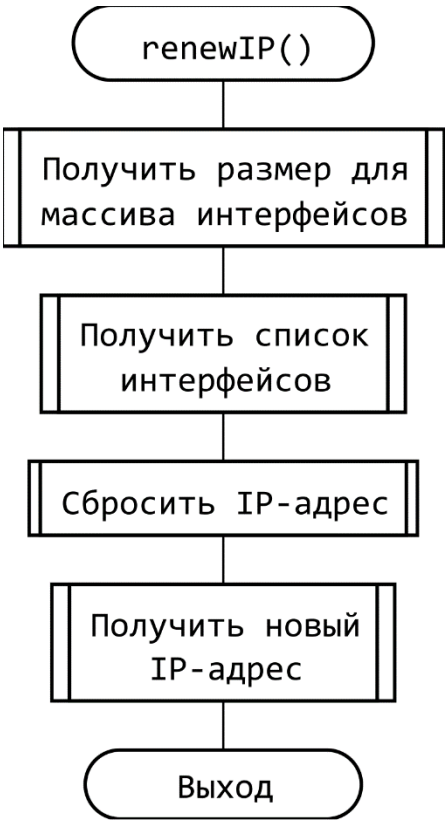
- **DhcpCApiInitialize –** загружает библиотеку dhcpcsdk
- **DhcpCApiCleanup –** выгружает библиотеку dhcpcsdk
- **getaddrinfo –** выполняет DNS-запрос по строке pNodeName, сохраняет полученный IP-адрес в ppResult
- **getnameinfo –** выполняет реверсивный DNS-запрос, по IP адресу pSockaddr сохраняет результат в строку pNodeBuffer

- **GetAdaptersInfo** – сохраняет информацию об адаптерах устройства, которая включает в себя IP-адрес DHCP-сервера.
- **GetInterfaceInfo** – возвращает доступные интерфейсы на устройстве
- **IpReleaseAddress –** запрашивает у DHCP-сервера сброс IP-адреса
- **IpRenewAddress –** запрашивает у DHCP-сервера новый IP-адрес

Разработка программы. Блок-схемы программы.




init()
Загрузить WSA
Загрузить DHCP
Выход
detach()
Выгрузить WSA
Выгрузить DHCP
Выход
getAddrInfo()
Получить IP-адрес, getaddrinfo
Вернуть IP-адрес
getNameInfo()
Получить имя домена, getnameinfo
Вернуть имя домена
printDHCPServerInfo()
Получить размер массива адаптеров
Получить массив адаптеров
Для всех адаптеров
-
+
В адаптере установлен DHCP?
-
+
Вывести IP-адрес
Выход



Анализ функционирования программ

Программа корректно выполняет DNS-запрос, а также реверсивный DNS-запрос. Так для сайта ya.ru был получен IP-адрес: 5.255.255.242. Реверсивный запрос также вернул доменное имя ya.ru. Для некоторых других сайтов, например vk.ru, реверсивный DNS возвращает srv133-129-240-87.vk.com, это связано с тем, что сервис использует несколько сервисов. Вывод информации о DHCP-сервере выводит адрес роутера 192.168.1.1, что логично, так как он распределяет адреса устройств в локальной сети.

Вывод: в ходе лабораторной изучили протоколы DHCP, DNS и составили программы согласно заданию

Текст программ. Скриншоты программ.

Ссылка на репозиторий с кодом: https://github.com/IAmProgrammist/comp_net/tree/lab6


```cpp
#include <iostream>
#include <algorithm>
#include <ws2tcpip.h>
#include <webstur/dhcp.h>

int main() {
    try {
        DHCPHelper::init();
        std::string input;

        // Считытваем команду пользователя
        while (true)
        {
            std::cout
```

```
                << "Enter T to get IP adress from domain name\n"
                << "Enter A to get domain name from IP address\n"
                << "Enter D to print IP address of DHCP server\n"
                << "Enter S to renew IP address\n"
                << "Enter E to exit loop"
                << std::endl;

            std::cin >> input;
            std::cin.get();
            std::transform(input.begin(), input.end(), input.begin(), toupper);
            if (input == "T") {
                // Ввести имя домена
                std::cout << "Enter domain name: ";
                std::cout.flush();
                std::string name;
                std::getline(std::cin, name);

                // Получить IP адрес
                auto result = DHCPHelper::getAddrInfo(name);

                char buf[64];
                InetNtopA(AF_INET, &((sockaddr_in *) result->ai_addr)->sin_addr, buf,
sizeof(buf));

                std::cout << "IP address: " << std::string(buf, buf + strlen(buf)) << std::endl;
            }
            else if (input == "A") {
                // Ввести IP адрес
                sockaddr_in info;
                info.sin_family = AF_INET;
                std::cout << "Enter IP address: ";
                std::cout.flush();
                std::string name;
                std::getline(std::cin, name);

                char buf[64] = {};
                memcpy(buf, name.c_str(), name.size());
                InetPtonA(AF_INET, buf, &info.sin_addr);

                std::cout << "Name: " << DHCPHelper::getNameInfo(info) << std::endl;
            }
            else if (input == "D") {
                // Вывести информацию о сервере
                DHCPHelper::printDHCPServerInfo(std::cout << "DHCP address: ") << std::endl;
            }
            else if (input == "S") {
                // Обновить информацию об IP
                DHCPHelper::renewIP();
            }
            else if (input == "E") {
                // Выход из цикла
                break;
            }
        }
    }
    catch (const std::runtime_error& error) {
        std::cerr << "Failed while running server. Caused by: '" << error.what() << "'" <<
std::endl;

        return -1;
    }

    // Выгрузка библиотеки WSA
    DHCPHelper::detach();

    return 0;
```

```
}
```

```
#include "pch.h"
#include <iostream>
#include <dhcpcsdk.h>
#include <iphlpapi.h>
#include <ws2tcpip.h>
#include <webstur/dhcp.h>

void DHCPHelper::init() {
    std::clog << "Loading WSA library" << std::endl;
    loadWSA();

    std::clog << "Loading DHCP library" << std::endl;
    DWORD dwVersion;
    if (DhcpCApiInitialize(&dwVersion) != 0) {
        throw std::invalid_argument(getErrorTextWithWSAErrorCode("Unable to load DHCP library"));
    }
}

void DHCPHelper::detach() {
    std::clog << "Unloading WSA library" << std::endl;
    unloadWSA();

    std::clog << "Unloading DHCP library" << std::endl;
    DhcpCApiCleanup();
}

addrinfo* DHCPHelper::getAddrInfo(std::string name) {
    addrinfo* res = nullptr;

    if (getaddrinfo(name.c_str(), nullptr, nullptr, &res) != 0)
        throw std::invalid_argument(getErrorTextWithWSAErrorCode("Unable to get addr info"));

    return res;
}

std::string DHCPHelper::getNameInfo(sockaddr_in ip) {
    char host[512];

    if (getnameinfo((sockaddr*) &ip, sizeof(ip), host, sizeof(host), nullptr, 0, 0) != 0)
        throw std::invalid_argument(getErrorTextWithWSAErrorCode("Unable to get name info"));

    return std::string(host, host + strlen(host));
}

std::ostream& DHCPHelper::printDHCPServerInfo(std::ostream& out) {
    ULONG ulOutBufLen = 0;
    PIP_ADAPTER_INFO pInfo = (PIP_ADAPTER_INFO) malloc(sizeof(IP_ADAPTER_INFO));
    PIP_ADAPTER_INFO pAdapter;
    if (GetAdaptersInfo(NULL, &ulOutBufLen) != NO_ERROR) {
        free(pInfo);
        pInfo = (PIP_ADAPTER_INFO) malloc(ulOutBufLen);
    }
    if (GetAdaptersInfo(pInfo, &ulOutBufLen) != NO_ERROR) {
        free(pInfo);
        throw std::invalid_argument(getErrorTextWithWSAErrorCode("Can't get interface info"));
    }

    for (pAdapter = pInfo; pAdapter != NULL; pAdapter = pAdapter->Next) {
        if (pAdapter->DhcpEnabled) {
            out << pAdapter->DhcpServer.IpAddress.String << std::endl;

            free(pInfo);
            return out;
```

```
        }
    }

    free(pInfo);

    return out;
}

void DHCPHelper::renewIP() {
    std::clog << "Getting interface info" << std::endl;
    ULONG ulOutBufLen = 0;
    PIP_INTERFACE_INFO pInfo;
    pInfo = (IP_INTERFACE_INFO*) malloc(sizeof(IP_INTERFACE_INFO));
    if (GetInterfaceInfo(NULL, &ulOutBufLen) != NO_ERROR) {
        free(pInfo);
        pInfo = (IP_INTERFACE_INFO*) malloc(ulOutBufLen);
    }
    if (GetInterfaceInfo(pInfo, &ulOutBufLen) != NO_ERROR) {
        free(pInfo);
        throw std::invalid_argument(getErrorTextWithWSAErrorCode("Can't get interface info"));
    }

    std::clog << "Resetting DHCP settings" << std::endl;
    if (IpReleaseAddress(&pInfo->Adapter[0]) != NO_ERROR)
        std::cerr << getErrorTextWithWSAErrorCode("Can't reset ip address") << std::endl;

    if (IpRenewAddress(&pInfo->Adapter[0]) != NO_ERROR)
        std::cerr << getErrorTextWithWSAErrorCode("Can't reset ip address") << std::endl;

    if (pInfo != NULL)
        free(pInfo);

    return;
}
```

```
#pragma once

#include <ostream>
#include <webstur/utils.h>

class DLLEXPORT DHCPHelper {
public:
    // Возвращает информацию об IP адресе по доменному имени
    //   name             доменное имя
    static addrinfo* getAddrInfo(std::string name);
    // Возвращает информацию о доменном имени по IP адресу
    //   addrinfo         информация об адресе
    static std::string getNameInfo(sockaddr_in ip);
    // Выводит информацию о DHCP-сервере
    static std::ostream& printDHCPServerInfo(std::ostream& out);
    // Обновить IP адрес при помощи DHCP
    static void renewIP();
    // Загружает библиотеку DHCP
    static void init();
    // Выгружает библиотеку DHCP
    static void detach();
};
```