

Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
по курсу «Архитектура операционных систем»
на тему
«Веб-прокси»

Выполнил:
студент группы 18ВВ1
Немцев А.А.

Принял:
доцент, к.т.н.
Егоров В.Ю.

удовл
21.09.22
ВЗ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

профессор М. А. Митрохин

«___» _____ 20__ г



ЗАДАНИЕ

на курсовое проектирование по курсу

Студенту Архитектура операционных систем
Нелицеву Александру Андреевичу Группа 18ВВ1
Тема проекта Веб-прокси

Исходные данные (технические требования) на проектирование

Требуется разработать программу, осуществляющую проксирование http запросов между браузером и веб-сервером. Прокси сервер регистрирует запрос со стороны браузера на передачу данных и ретранслирует их на заданный сервер.

Операционная система: Windows

Языки программирования: C, C++, C#

Объем работы по курсу

1. Расчетная часть

- 1.1. Разработка протокола взаимодействия процессов
- 1.2. Разработка программы

2. Графическая часть

Не менее пяти ИМЛ диаграмм, иллюстрирующих структуру процессов

3. Экспериментальная часть

Отладка программы на компьютере

Срок выполнения проекта по разделам

1 в соответствии с графиком выполнения работы

2
3
4
5

Дата выдачи задания « 22 » 09.2020

Дата защиты проекта « 21 » 04.2022

Руководитель ВЗ /

Задание получил « 22 » сентября 2020 г.

Студент Кенчев

Содержание

Содержание	2
1. Введение	5
2. Постановка задачи.....	6
3. Теоретическая часть задания.....	7
4. Описание алгоритма программы.....	11
5. Описание программы.....	13
6. Описание работы программы	14
7. Заключение.....	15
Список литературы	16
Приложение А. Листинг программы	17

1. Введение

Сегодня невозможно представить современного человека без использования глобальной сети – Internet. Данная сеть обеспечивает обмен информацией между всеми компьютерами и устройствами, подключенными к ней. Internet предоставляет своим пользователям возможность использования множества различных ресурсов.

Работа с глобальной сетью дает возможность получать необходимую информацию, не только на высокой скорости, но и в больших объемах, что позволяет затрачивать меньше времени на поиск и обработку данных, если бы человек не использовал сеть.

Помимо применения Internet, как идеального источника данных, можно использовать данную сеть для коммуникации, которая представляется не только в виде символьного и информационного содержания, но графического, мультимедийного: изображения, видео, аудио и т.д.

Благодаря появлению и развитию Internet возникла универсальная информационная среда, которая отличается высокой степенью интерактивности, оперативностью обмена информацией, доступностью информации и простотой передачи информации.

2. Постановка задачи

Требуется разработать программу, осуществляющую проксирование http запросов между браузером и веб-сервером.

Прокси сервер регистрирует запрос со стороны браузера на передачу данных и ретранслирует их на заданный сервер.

Операционная система: Windows.

Языки программирования: C, C++, C#.

3. Теоретическая часть задания

Общие теоретические сведения

На сегодняшний день множество компьютеров и устройств связаны между собой глобальной сетью – Internet, либо отдельной локальной сетью. Вся сеть включает в себя отдельные элементы – хосты, которые представляют собой компьютеры и другие подключенные устройства. Данные элементы соединены между собой каналами коммуникации. Для взаимодействия друг с другом хосты применяют протоколы – соглашения о том, как пакеты данных будут передаваться по каналам связи, то есть происходит упорядочивание взаимодействия.

Существует множество различных протоколов. Протоколы, которые используются для передачи данных по сети, составляют семейство протоколов TCP/IP, которые организованы в уровневую систему:

- сетевой уровень (IP) использует уровни, которые представляют собой физические каналы связи для передачи пакетов данных;
- транспортный уровень. Его образуют протоколы TCP и UDP, которые используют отдельные порты для передачи данных. Но приложение использует данный уровень не на прямую, а через специальное API – сокеты.

Сокеты – это интерфейс для создания сетевых приложений, который опирается на встроенные возможности операционной системы.

В итоге, когда приложение «А» посылает сообщение приложению «В», запущенному на другом хосте, то приложение «А» обращается к сокетам для передачи данных на уровень TCP / UDP. Далее с этого транспортного уровня данные передаются сетевому уровню - уровню протокола IP. Этот протокол передает данные далее физическим уровням, и после этого данные уходят по сети.

Для того, чтобы хосты были идентифицированы, каждый хост имеет свой адрес, который представлен 32-битным числом. Кроме адреса используются порты, которые представляют собой 16-битное число. Их использование помогает разграничить несколько запущенных приложений в одном хосте.

Протокол TCP

Данный протокол использует надежную связь, для ее обеспечения он посылает подтверждение о получении данных и запрашивает повторную передачу, если данные не получены или испорчены.

TCP необходимо, чтобы перед отправкой пакета данных было установлено и открыто соединение с известным портом хоста, на который передается информация.

Процедура установления соединения происходит следующим образом.

Рассмотрим пример обмена данными между серверным приложением и клиентским. В первую очередь, серверное приложение должно создать соединение с известным номером порта, а затем перейти в режим прослушивания – ожидания поступления входящих запросов. Клиентское приложение отправляет серверу TCP-сегмент, в котором хранится бит для синхронизирующего порядкового номера (SYN) и начальный номер последовательности (ISN) – число, с которого начинается нумерация отправляемых сокетов. В ответ сервер посылает свой TCP-сегмент, где в заголовке будет установлен бит, подтверждающий соединение для получения данных (ACK), SYN, означающий запрос связи для передачи данных клиенту и ISN. Следующий сегмент, посылаемый от клиентского приложения, содержит в себе ACK и данные, которые планировалось передать. При получении сообщения в ответ всегда отправляется сообщение ACK. Если до получения ACK отправителем истекает тайм-аут, сообщение помещается в очередь на повторную передачу.

Сеанс обмена данными заканчивается процедурой разрыва соединения, которая аналогична процедуре установки, с той разницей, что вместо SYN для разрыва используется служебный бит FIN, свидетельствующий об отсутствии данных для отправки, он устанавливается в заголовке последнего сегмента с данными, отправляемого узлом.

Схема работы протокола TCP отображена на рисунке 1.

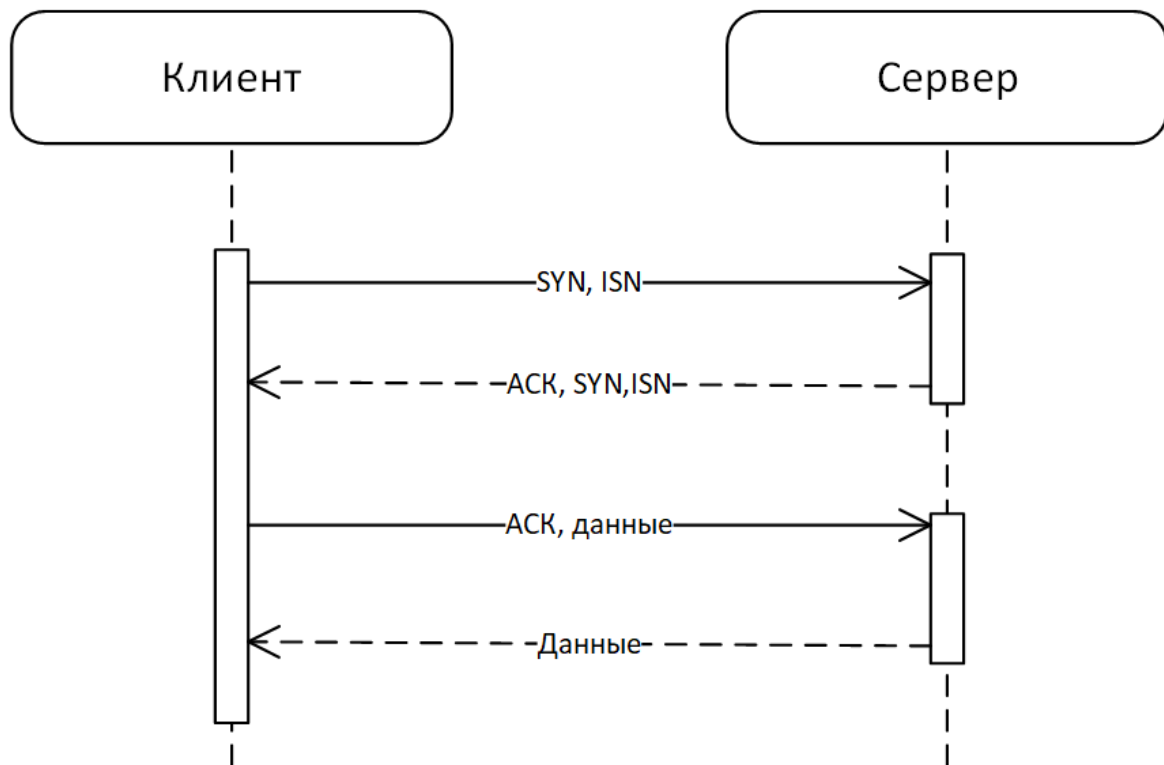


Рисунок 1 – Алгоритм установления соединения и обмена данными по протоколу TCP

Протокол HTTP

HTTP – это широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов. Передача информации по данному протоколу осуществляется через TCP/IP соединение по принципу клиент-сервер.

Основным объектом, участвующим в протоколе обмена данными, является ресурс, на который указывает URI – уникальный идентификатор ресурса в запросе клиента. Протокол дает возможность указать способ кодирования ресурса по различным параметрам, благодаря чему может происходить обмен двоичными данными.

Структура HTTP протокола определяет, что каждое HTTP сообщение состоит из трех частей:

- стартовая строка – хранит тип сообщения;
- заголовки – содержат описание тела сообщения, параметры передачи;
- тело сообщения – передаваемые данные.

Сообщения протокола делят на 2 вида: запрос (от клиента к серверу) и ответ (от сервера к клиенту). Главным отличием между сообщениями является то, что в первой строке ответа сервер всегда возвращает коды состояния, которые информируют клиента о результатах выполнения. HTTP протокол не имеет промежуточных состояний между запросом и ответом.



Рисунок 2 – Работа HTTP протокола

4. Описание алгоритма программы

Разработанная программа основывается на вышеописанных протоколах и осуществляет работу прокси-сервера для HTTP-запросов между браузером и веб-сервером таким образом, что запрошенный ресурс ретранслируется на выбранном порте локального адреса (127.0.0.1:3000).

Данный прокси-сервер – это промежуточный сервер между клиентом (веб-браузером) и целевым сервером (lib.ru / 81.176.66.163:80).

Первым шагом программы является объявление конечных сетевых точек, затем устанавливается TCP-соединение, при помощи инициализации нового экземпляра TcpListener, и сервер начинает прослушивать входящие запросы на подключение. Как только определяется, что имеются ожидающие запросы на подключение, начинает выполняться операция, чтобы принять попытку для входящего подключения.

Создается новый TcpClient для обработки связи с удаленным узлом (127.0.0.1:3000) и основной поток данных для доступа к сети. Затем при помощи реализации интерфейса сокетов инициализируется новый экземпляр Socket и устанавливается подключение к удаленному узлу (81.176.66.163:80).

Далее в цикле, пока в основном потоке имеется информация для передачи, считываются пакеты данных и отправляются в подключенный сокет. После того, как все необходимые данные переданы, создается новый цикл с условием: пока есть полученные данные из сети доступные для чтения. В этом блоке программы выполняется чтение пакетов данных, полученных от удаленного хоста и запись их в основной поток.

После выполнения операции по передаче-приему данных Socket закрывает подключение и освобождает все связанные ресурсы и TcpClient освобождает все используемые ресурсы.

Сервер переходит в ожидание нового запроса.

Диаграмма деятельности обработки запроса отображена на рисунке 2.

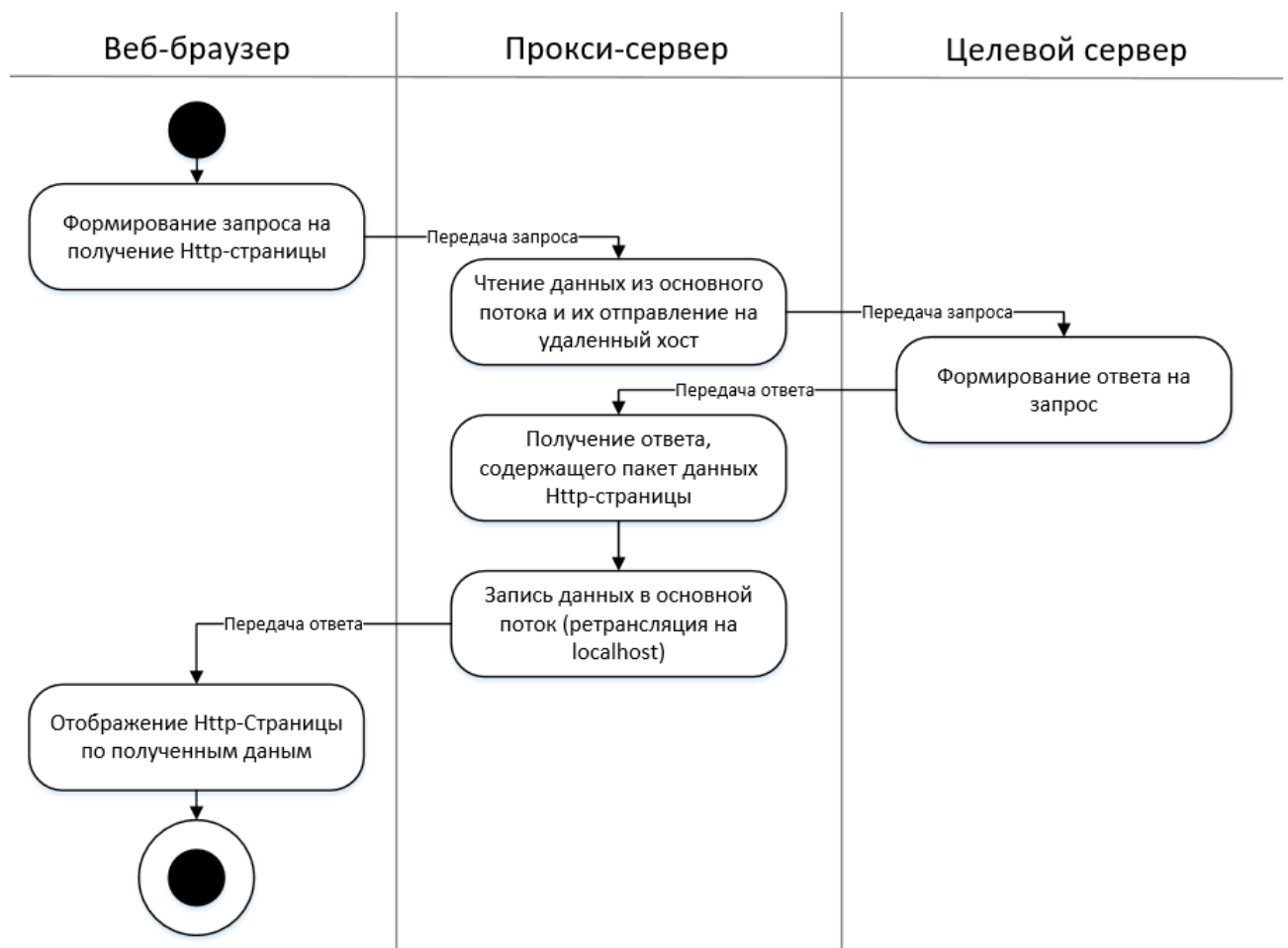


Рисунок 3 – Диаграмма деятельности программы при обработки запроса

5. Описание программы

Основные модули

Программа разбита на несколько модулей. *Program.cs* - главный файл проекта. В нем осуществляется создание нового экземпляра класса *Server* и реализация функции *Main()*.

Файл *Server.cs* содержит в себе инициализацию класса *Server*, функцию *Connect()* и функцию *ProcessClient()*.

Функция *Connect()* осуществляет запуск прослушивания входящих запросов и обработку ожидающих запросов на подключение.

Функция *ProcessClient()* выполняет соединение с удаленным хостом (*lib.ru*), далее происходит обмен пакетами данных между *localhost* и *lib.ru*. После чего осуществляется ретрансляция http-страницы на *127.0.0.1:3000*.

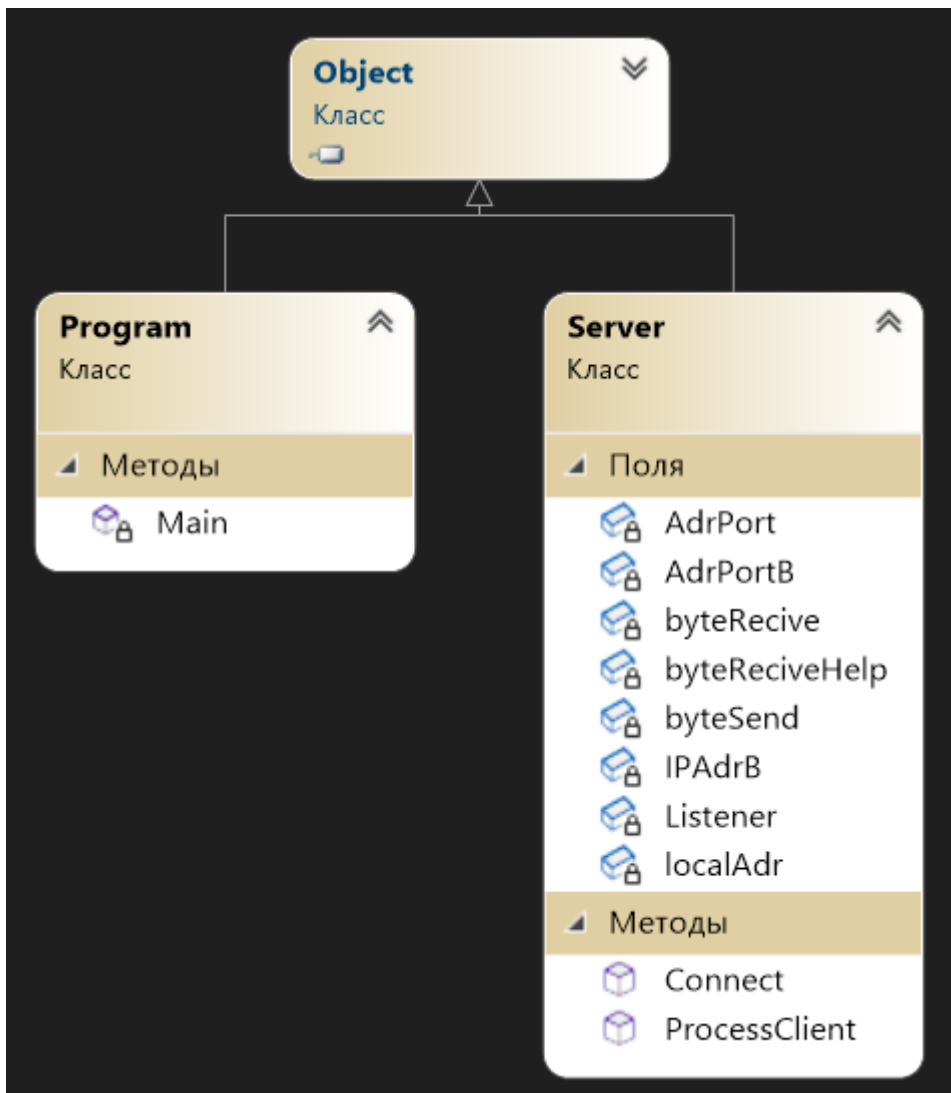


Рисунок 4 – UML-диаграмма программы

Схема алгоритма программы

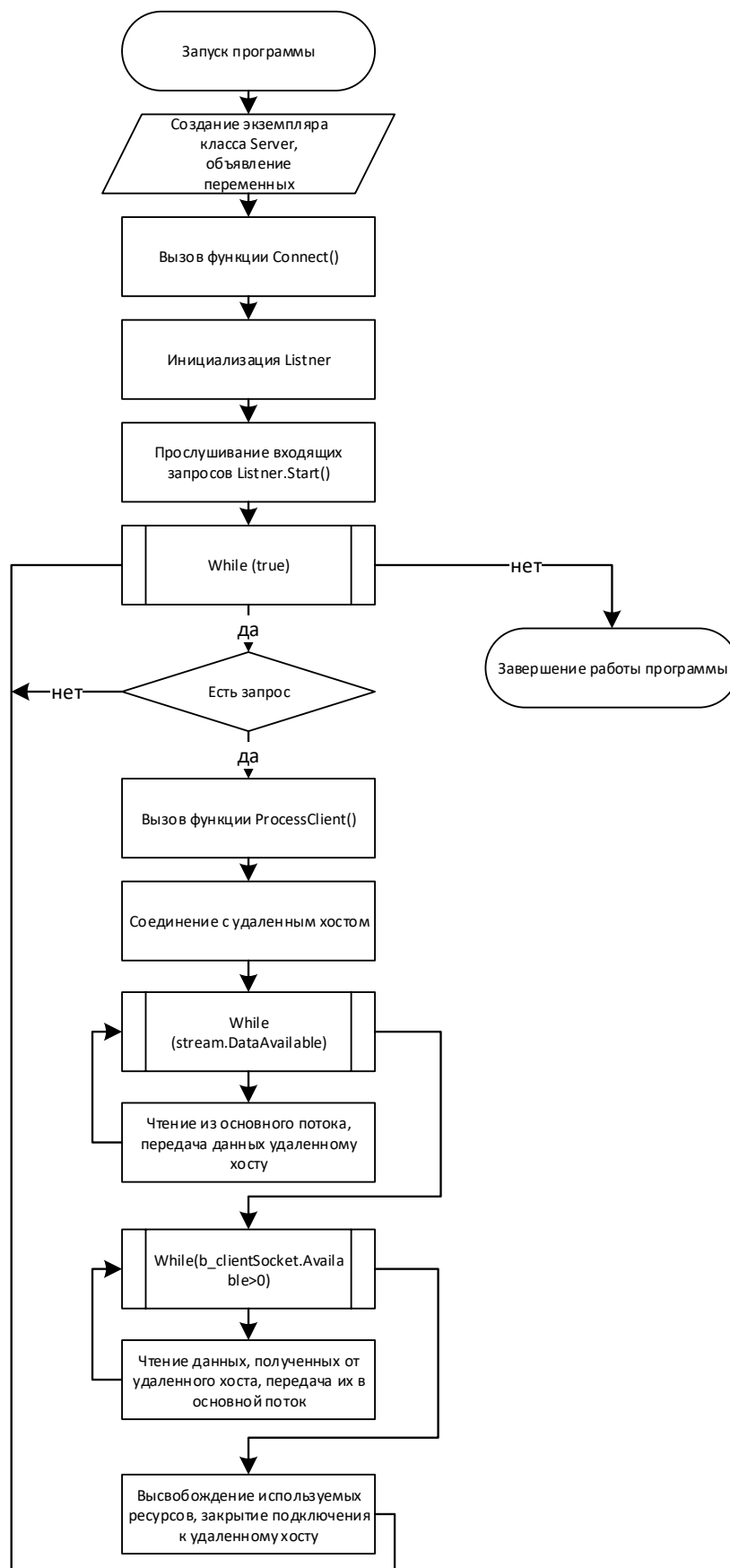


Рисунок 5 – Схема алгоритма программы

6. Описание работы программы

Работа программы изображена на рисунках 4-9.

В поисковой строке веб-браузера необходимо ввести: <http://127.0.0.1:3000> и выполнить поиск.

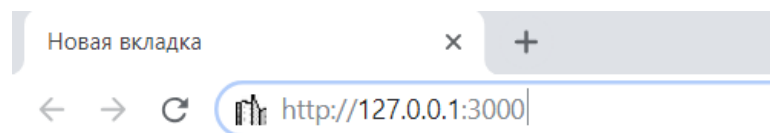


Рисунок 6 – Ввод адреса локального хоста

После выполнения вышеуказанных действий на локальном хосте отобразится начальная страница электронной библиотеки, находящейся по адресу <http://lib.ru>, но отображение страницы благодаря ретранслированию произойдет на локальном адресе <http://127.0.0.1:3000>.

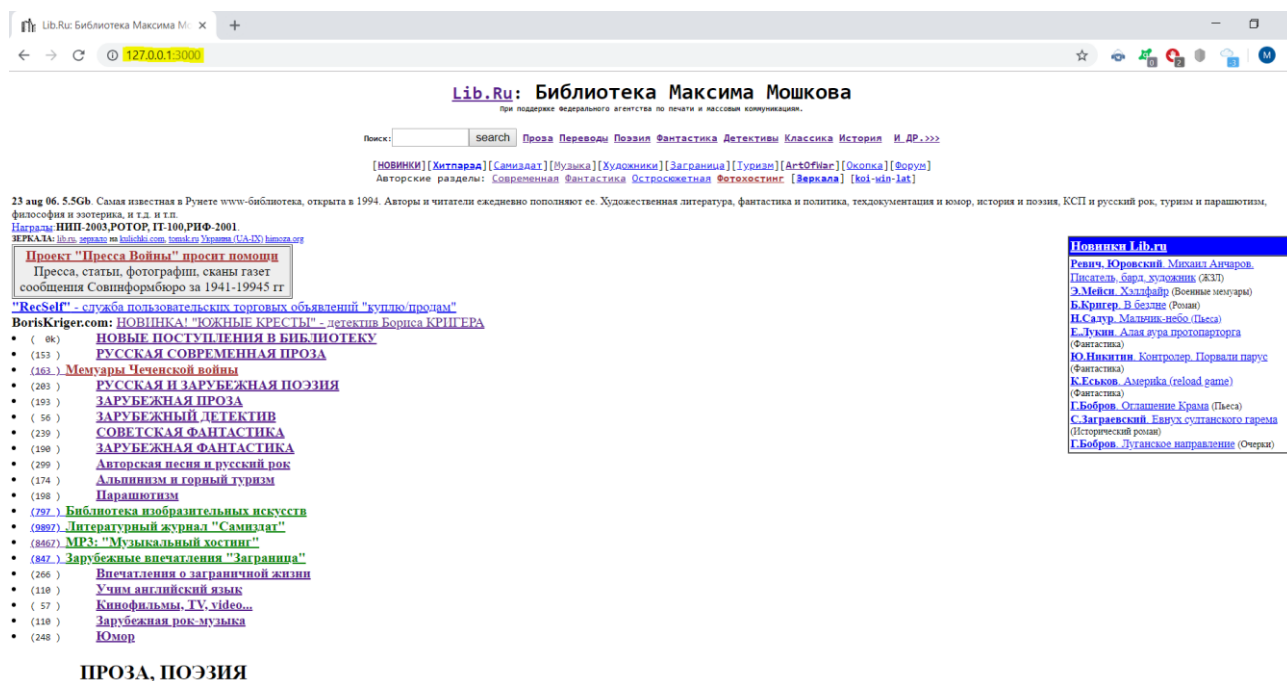


Рисунок 7 – Результат перехода по ссылке на локальный хост

В консольном окне будет отображен размер полученных пакетов, и если все данные получены, появится запись «Страница отображена, ожидается новый запрос».

```
При вводе в поисковой строке браузера: http://127.0.0.1 будет осуществлена ретрансляция http://lib.ru
Размер пакета = 157 (байт)
Размер пакета = 433 (байт)
Размер пакета = 0 (байт)
Страница отображена, ожидается новый запрос
```

Рисунок 8 – Результат перехода по ссылке на локальный хост с отображением размера полученных пакетов данных от lib.ru.

Далее кликнем по гиперссылке «Награды». Веб-браузер получит код запрашиваемой страницы путем ее проксирования на локальный адрес, и требуемая страница будет отображена в окне браузера.

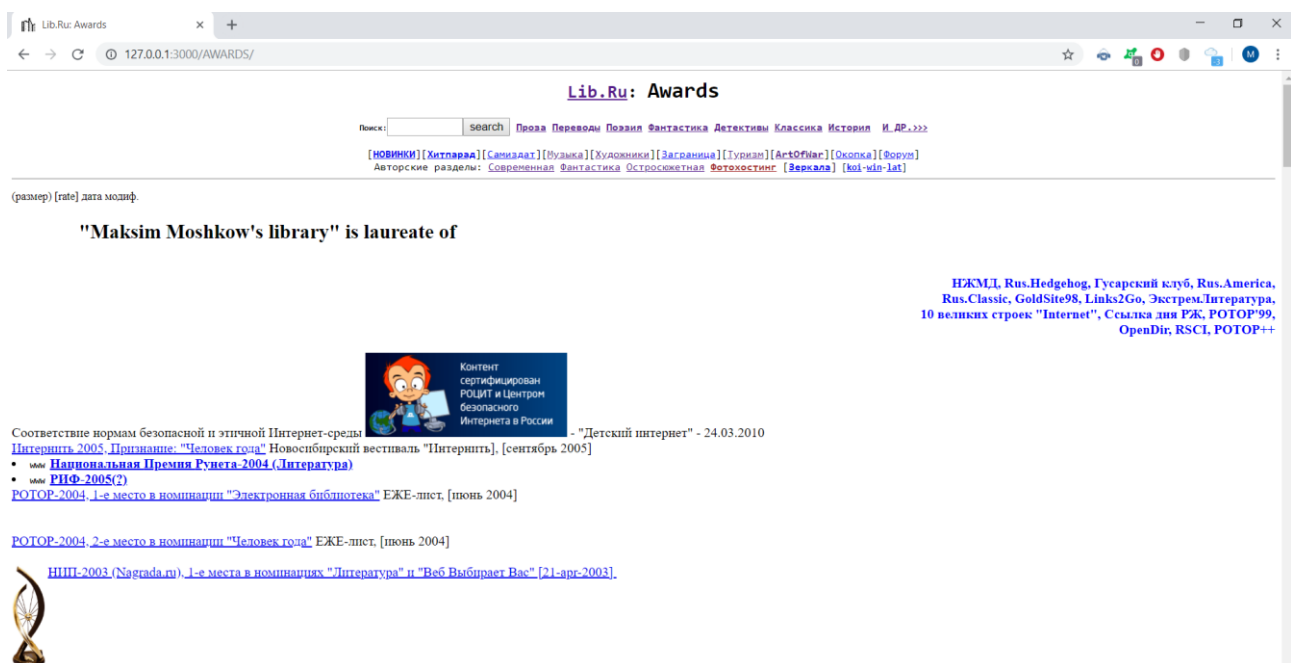


Рисунок 9 – Результат при переходе по гиперссылке «Награды»

Затем вернемся на начальную страницу и перейдем в раздел, например, «Зарубежная проза» → «Жоржи Амаду».



Рисунок 10 – Результат при переходе по гиперссылке «Зарубежная проза» → «Жоржи Амаду».

При вводе в поисковой строке браузера: http/127.0.0.1 будет осуществлена ретрансляция http://lib.ru	
Размер пакета = 157 (байт)	Отображение "Начальная страница"
Размер пакета = 433 (байт)	
Размер пакета = 0 (байт)	
Страница отображена, ожидается новый запрос	Отображение "Награды"
Размер пакета = 157 (байт)	
Размер пакета = 433 (байт)	
Размер пакета = 0 (байт)	Отображение "Начальная страница"
Страница отображена, ожидается новый запрос	
Размер пакета = 157 (байт)	
Размер пакета = 433 (байт)	Отображение "Зарубежная проза"
Размер пакета = 0 (байт)	
Страница отображена, ожидается новый запрос	
Размер пакета = 1452 (байт)	Отображение "Жоржи Амаду"
Размер пакета = 2904 (байт)	
Размер пакета = 1452 (байт)	
Размер пакета = 213 (байт)	
Размер пакета = 0 (байт)	
Страница отображена, ожидается новый запрос	
Размер пакета = 157 (байт)	
Размер пакета = 433 (байт)	
Размер пакета = 0 (байт)	
Страница отображена, ожидается новый запрос	

Рисунок 11 – Результат при переходе по гиперссылке «Зарубежная проза»
→ «Жоржи Амаду» с отображением размера полученных пакетов данных от
lib.ru.

7. Заключение

Таким образом, была разработана программа, осуществляющая проксирование http запросов между браузером и веб-сервером. При помощи разработанного прокси-сервера происходит ретрансляция полученных данных на заданный сервер.

Список литературы

1. Документация Microsoft. [Электронный ресурс]: URL: <https://docs.microsoft.com> (дата обращения: 14.12.2018)
2. Основы работы с сетями в С#. [Электронный ресурс]: URL: <https://metanit.com> (дата обращения: 14.12.2018)
3. Программирование сокетов. [Электронный ресурс]: URL: <http://www.realcoding.net/article/view/2325> (дата обращения: 14.12.2018)
4. Основы сокетов. [Электронный ресурс]: URL: <http://masters.donntu.org/2005/fvti/lukyanov/library/ipc/socket.html> (дата обращения: 14.12.2018)
5. Сетевое программирование на С#. [Электронный ресурс]: URL: https://professorweb.ru/my/csharp/web/level1/web_index.php (дата обращения: 14.12.2018)
6. Протокол TCP – назначение и функции. [Электронный ресурс]: URL: <http://ciscotips.ru/tcp> (дата обращения: 14.12.2018)
7. Олифер В.Г., Олифер Н.А./ Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 4-е издание. — СПб.: Питер, 2010. — 944 с.
8. Хейлсберг А., Торгерсен М., Вилтамут С., Голд П./ Язык программирования С#. Классика Computer Science. 4-е издание. — СПб.: Питер, 2011. — 784 с.
9. Объектно-ориентированное программирование. Практика. [Электронный ресурс]: URL: <https://metanit.com/sharp/tutorial/3.29.php> (дата обращения: 26.11.2018)
10. Вайсфельд М./ Объектно-ориентированное мышление. — СПб.: Питер, 2014. — 304 с.: ил. — (Серия «Библиотека программиста»).

Приложение А. Листинг программы

Приложение А1

Файл «Program.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    namespace ConsoleApp2
    {
        class Program
        {
            static void Main(string[] args)
            {
                Console.WriteLine("При вводе в поисковой строке браузера: http/127.0.0.1
будет осуществлена ретрансляция http://lib.ru");
                Server s = new Server();
                Server.Connect();
            }
        }
    }
}
```

Приложение А2

Файл «Server.cs»

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.IO;
using System.Threading;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    public class Server
    {
        static IPAddress localAdr = IPAddress.Parse("127.0.0.1");
        static IPEndPoint AdrPort = new IPEndPoint(localAdr, 3000);
        static TcpListener Listener;
        static IPAddress IPAdrB = IPAddress.Parse("81.176.66.163");
        static IPEndPoint AdrPortB = new IPEndPoint(localAdr, 80);
        static Byte[] byteSend, byteRecive, byteReciveHelp;

        public static void Connect()
        {
            Listener = new TcpListener(AdrPort);
            Listener.Start();

            while (true)
            {
                if (Listener.Pending())
                {
                    Listener.BeginAcceptTcpClient(new AsyncCallback(ProcessClient),
Listener);
                }
            }
        }
    }
}
```

```

    }

    public static void ProcessClient(IAsyncResult ar)
    {
        TcpListener Listener2 = (TcpListener)ar.AsyncState;
        TcpClient Client = Listener2.EndAcceptTcpClient(ar);

        NetworkStream stream = Client.GetStream();

        Socket b_clientSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
        b_clientSocket.Connect(IPAddrB, 80);

        Client.ReceiveBufferSize = 32 * 1024;

        byteSend = new byte[Client.ReceiveBufferSize];
        byteRecive = new byte[Client.ReceiveBufferSize];

        try
        {
            do
            {
                stream.Read(byteSend, 0, byteSend.Length);
                b_clientSocket.Send(byteSend, 0, byteSend.Length, SocketFlags.None);

                } while (stream.DataAvailable);
                string msg = Encoding.UTF8.GetString(byteSend);
                int readed = 0;

                do
                {
                    readed = b_clientSocket.Receive(byteRecive);

                    msg = Encoding.UTF8.GetString(byteRecive);
                    Console.WriteLine("Размер пакета = " + readed + " (байт)");

                    stream.Write(byteRecive, 0, readed);

                } while (readed > 0);

                Console.WriteLine("Страница отображена, ожидается новый запрос");
                readed = 0;
                b_clientSocket.Shutdown(SocketShutdown.Both);
                b_clientSocket.Close();
                Client.Dispose();
            }
            catch
            {
                Console.WriteLine("Ошибка! Попробуйте обновить страницу.");
            }
        }
    }
}

```