

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Санкт-Петербургский государственный технологический институт  
(технический университет)»  
СПбГТИ(ТУ)

УГНС	09.00.00	Информатика и вычислительная техника
Направление подготовки	09.03.01	Информатика и вычислительная техника
Направленность (профиль)		Системы автоматизированного проектирования
Форма обучения		очная
Факультет		Информационных технологий и управления
Кафедра		Систем автоматизированного проектирования и управления
Учебная дисциплина		<b>ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ, СЕТИ И ТЕЛЕКОММУНИКАЦИИ</b>
Курс	III	Группа 494

**КУРСОВОЙ ПРОЕКТ**

**Тема:** Компьютерные сети  
**Задача:** Реализация ргоху-сервера

Студент	_____	Гусев А.А.
Руководитель, доцент, к.т.н.	_____	Макарук Р.В.
Оценка за курсовой проект	_____	_____

Санкт-Петербург  
2021-2022 уч. год

## Содержание

ВВЕДЕНИЕ .....	3
1 АНАЛИТИЧЕСКАЯ ЧАСТЬ.....	4
1.1 Теоретический обзор компьютерных сетей .....	4
1.2 Анализ технологии Proxu .....	7
1.3 Существующие примеры реализации проху-сервера.....	8
1.4 Обзор и обоснование выбора инструментальных средств разработки .....	9
2 ОСНОВНАЯ ЧАСТЬ.....	11
2.1 Определение структурной (иерархической) схемы решения задачи .....	11
2.2 Определение основных этапов проектирования .....	11
2.3 Анализ ограничений и исключительных ситуаций для алгоритмов .....	11
2.4 Разработка основных алгоритмов задачи.....	12
2.5 Разработка архитектуры программы .....	12
2.6 Разработка дисплейных фрагментов .....	13
2.7 Отладка программного комплекса.....	14
2.8 Тестирование разработанного программного продукта.....	14
ВЫВОДЫ ПО РАБОТЕ .....	17
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	18

## ВВЕДЕНИЕ

Благодаря высокой скорости технологического прогресса различные технологии, считавшиеся не так давно новинкой, очень быстро проникают друг в друга. При этом в XXI веке различия между сбором, транспортировкой, хранением и обработкой информации продолжают быстро исчезать. Организации с сотнями офисов, разбросанных по всему миру, должны иметь возможность получать информацию о текущем состоянии своего самого удаленного офиса мгновенно, нажатием кнопки. По мере роста нашего умения собирать, обрабатывать и распространять информацию, потребности в средствах еще более сложной обработки информации растут все быстрее.

На слияние компьютеров и коммуникаций существенно влияет принцип организации компьютерных систем. Некогда доминирующее понятие «вычислительного центра» как комнаты с большим компьютером, к которому пользователи приносят свою работу для обработки, является теперь полностью устаревшим. Старая модель единственного компьютера, служащего всем вычислительным потребностям организации, была заменена на схему, при которой задание выполняет большое количество отдельных, но связанных компьютеров. Эти системы называют компьютерными сетями [1].

На сегодняшний день компьютерные сети и сетевое программирование являются очень востребованными сферами информационных технологий. Целью данной курсовой работы является ознакомление с принципом работы компьютерных сетей, формирование понимания принципов работы локальной сети, а также реализация внутри сети промежуточного сервера, выполняющего роль посредника между пользователем и целевым сервером, который позволяет клиентам как выполнять косвенные запросы к другим сетевым службам, так и получать ответы.

В ходе работы необходимо рассмотреть основные теоретические сведения о компьютерных сетях и технологии протоколов, а также проанализировать примеры существующего программного обеспечения, реализуемого в ходе курсового проекта.

# 1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

## 1.1 Теоретический обзор компьютерных сетей

Компьютерная сеть (вычислительная сеть) — система, обеспечивающая обмен данными между вычислительными устройствами — компьютерами, серверами, маршрутизаторами и другим оборудованием или программным обеспечением [1].

Эндрю Таненбаум в книге [1] предлагает использовать термин «компьютерная сеть», чтобы обозначить набор автономных компьютеров, связанных одной технологией. Два компьютера называют связанными, если они в состоянии обмениваться информацией (рисунок 1). Сети бывают различных размеров, формы и конфигураций. Они обычно соединяются вместе, чтобы создать большие сети, самым известным примером сети сетей является Интернет [1]. Компьютерные сети, называемые также сетями передачи данных, появились в конце 1960-х гг., и являются результатом развития компьютерных и телекоммуникационных технологий [2].

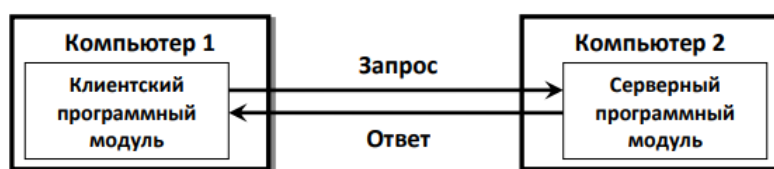


Рисунок 1 — Схема взаимодействия компьютеров

Существует путаница между понятиями компьютерная сеть и распределенная система, но это два разных понятия. Основное их различие заключается в том, что в распределенной системе наличие многочисленных автономных компьютеров незаметно для пользователя, с его точки зрения пользователя это единая связанная система. Обычно имеется набор программного обеспечения на определенном уровне (над операционной системой), которое называется связующим ПО и отвечает за реализацию этой идеи. Известный пример распределенной системы — это Всемирная паутина (World Wide Web), в которой, все выглядит как документ. Распределенная система является программной системой, построенной на базе сети. Эта программная система обеспечивает высокую степень связности и прозрачности элементов [1].

В компьютерных сетях отсутствует единая модель, а также нет программного обеспечения для ее реализации. В случае работы с компьютерной сетью пользователь имеет дело с реальными компьютерами, которые не связаны в одну систему [1].

Эти два понятия имеют очень много общего. Например, как компьютерная сеть, так и распределенная система занимаются перемещением файлов. Разница заключается в том, кто вызывает эти перемещения — система или пользователь.

Большинство современных организаций используют в своей работе большое число компьютеров. По мере роста компании и развития вычислительной техники так или иначе было принято решение соединить их, чтобы быть в состоянии передавать информацию по всей компании.

Существует несколько примеров построения сети. Наиболее распространенным примером является клиент-серверная модель (рисунок 2). Самая популярная реализация — веб-приложение, в котором сервер производит веб-страницы, основанные на его базе данных в ответ на запросы клиента, которые могут обновить базу данных [1].

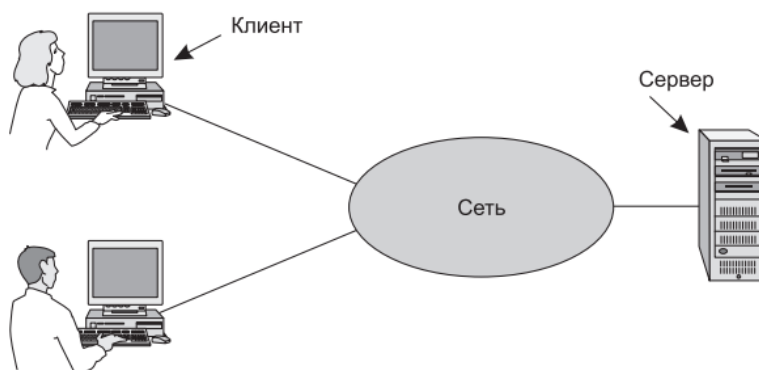


Рисунок 2 — Упрощенная модель клиент-серверной сети

Также не менее распространенным примером построения сетей является технология равноправных сетей (peer-to-peer) (рисунок 3). Люди, входящие в некоторую группу пользователей, могут общаться друг с другом, каждый может связаться с каждым, разделение на клиентские и серверные машины в этом случае отсутствует.

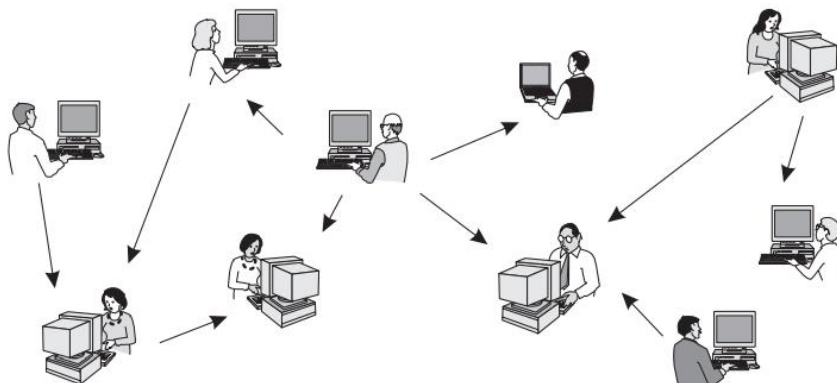


Рисунок 3 — Пример схемы peer-to-peer

Существуют варианты классификации разных вычислительных сетей по назначению и характеристикам.

По территориальной распространенности:

- BAN (Body Area Network) — сеть надеваемых или имплантированных компьютерных устройств.
- PAN (Personal Area Network) — персональная сеть, предназначенная для взаимодействия различных устройств, принадлежащих одному владельцу.
- LAN (Local Area Network) — локальные сети, имеющие замкнутую инфраструктуру до выхода на поставщиков услуг. Являются самым распространённым типом сетей.
- CAN (Campus Area Network) — кампусная сеть, объединяет локальные сети близко расположенных зданий.

- MAN (Metropolitan Area Network) — городские сети между учреждениями в пределах одного или нескольких городов.
- WAN (Wide Area Network) — глобальная сеть, покрывающая большие географические регионы [4].

Различают также сети по типу сетевой топологии. Сетевая топология — это конфигурация графа, вершинам которого соответствуют конечные узлы сети, а рёбрам — физические или информационные связи между вершинами.

Наиболее распространенными сетевыми топологиями являются:

- Шина — представляет собой общий кабель, к которому подсоединены все рабочие станции.
- Кольцо — топология, в которой каждый компьютер соединён линиями связи только с двумя другими: от одного он только получает информацию, а другому только передаёт. На каждой линии связи, работает только один передатчик и один приёмник.
- Звезда — базовая и наиболее распространённая топология компьютерной сети, в которой все компьютеры сети присоединены к центральному узлу (обычно коммутатор), образуя физический сегмент сети. Подобный сегмент сети может функционировать как отдельно, так и в составе сложной сетевой топологии.
- Дерево — топология компьютерной сети, в которой каждый узел более высокого уровня связан с узлами более низкого уровня звездообразной связью, образуя комбинацию звезд.
- Смешанная топология — сетевая топология, преобладающая в крупных сетях с произвольными связями между компьютерами. В таких сетях можно выделить отдельные связанные фрагменты (подсети), имеющие типовую топологию, поэтому их называют сетями со смешанной топологией [3].

Локальная сеть представляет собой набор компьютеров и периферийных устройств, соединенных кабелями. Локальные сети характерны тем, что расстояния между компонентами сети сравнительно невелики, как правило, не превышают нескольких километров. Выбор сети определяется числом подключаемых пользователей, их приоритетом, а также общими требованиями [3].

Независимо от типа сетей, к ним предъявляются общие требования:

- скорость;
- адаптируемость — свойство локальной сети расширяться и устанавливать рабочие станции там, где это требуется;
- надежность — свойство локальной сети сохранять полную или частичную работоспособность вне зависимости от выхода из строя некоторых узлов или конечного оборудования.

При проектировании и реализации локальных сетей часто возникает необходимость контролировать трафик, снижать нагрузку на сеть при помощи кэширования данных, защищать сеть от внешнего доступа и ограничивать доступ к некоторым ресурсам из локальной сети. Перечисленные требования достигаются путем реализации проху-сервера.

## 1.2 Анализ технологии Proxu

Proxu server (Прокси сервер) — это промежуточный компьютер, который является посредником между компьютером пользователя и Интернетом [5].

Чаще всего прокси-серверы применяются для следующих целей:

- Обеспечение доступа с компьютеров локальной сети в Интернет.
- Кэширование данных: если часто происходят обращения к одним и тем же внешним ресурсам, то можно держать их копию на прокси-сервере и выдавать по запросу, снижая тем самым нагрузку на канал во внешнюю сеть и ускоряя получение клиентом запрошенной информации [1].
- Сжатие данных: прокси-сервер загружает информацию из Интернета и передаёт информацию конечному пользователю в сжатом виде. Такие прокси-серверы используются в основном с целью экономии внешнего сетевого трафика клиента или внутреннего — компании, в которой установлен прокси-сервер .
- Защита сети от внешнего доступа: например, можно настроить прокси-сервер так, что локальные компьютеры будут обращаться к внешним ресурсам только через него, а внешние компьютеры не смогут обращаться к локальным вообще (они «видят» только прокси-сервер).
- Ограничение доступа из локальной сети к внешней: например, можно запретить доступ к определённым сайтам, ограничить использование интернета каким-то локальным пользователям, устанавливать квоты на трафик или полосу пропускания [4].

Существует несколько разновидностей прокси, главным отличием которых являются выполняемые функции:

- HTTP/HTTPS проху – наиболее распространённый тип прокси серверов, который зачастую имеет 80, 8080, 3128 номер порта. HTTP прокси делятся уровнем анонимности на: прозрачные (не скрывают реальный IP адрес клиента), непрозрачные (указывают или не указывают на то, что используется прокси, но не выдают реальный IP адрес клиента) и искажающие (искажают IP адрес клиента).
- SOCKS проху – прокси сервер, передающий абсолютно все данные от клиента к серверу, не изменяя и не добавляя ничего. Имеет подтипы SOCKS4, SOCKS4a, SOCKS5. Чаще всего SOCKS проху имеют 1080, 1081 номер порта.
- FTP проху – прокси сервер, предназначенный для работы с файловыми менеджерами.

Разница между обычными и кэширующими прокси-серверами весьма важна. Обычный прокси-сервер просто пересылает запросы и ответы. Кэширующий прокси-сервер способен поддерживать собственное хранилище ответов, полученных ранее. Когда прокси-сервер получает запрос, который может быть удовлетворен кэшированным ответом, запрос не пересылается, а ответ возвращается прокси-сервером.

Прокси-сервер может работать в следующих трех основных режимах:

- Прозрачный режим. В этом режиме HTTP соединение, осуществляемое пользователями, перенаправляется на прокси-сервер без их ведома или явной конфигурации. В этом режиме не требуется настройка клиентов;
- Аутентифицирующий режим (рисунок 4). Для работы в этом режиме компьютеры пользователей должны быть настроены для работы с прокси-сервером, т.е. явно указан адрес прокси-сервера в настройках. Для авторизованных групп возможно применение различных настроек контроля доступа;
- Обратный прокси-сервер. Прокси-сервер кэширует исходящие данные. Обратный прокси-сервер получает данные у HTTP сервера от имени пользователя и передает их обратно пользователю (например, в Интернет) [5].

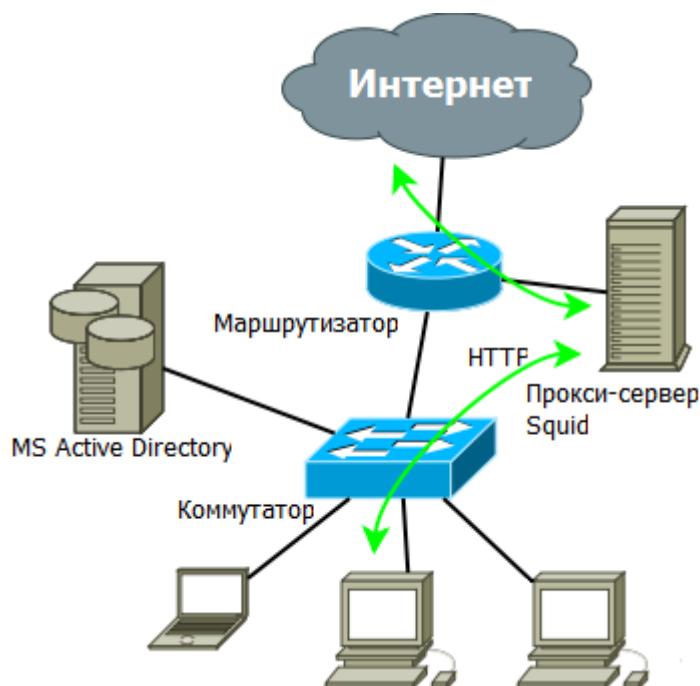


Рисунок 4 - Схема аутентифицирующего режима прокси-сервера.

### 1.3 Существующие примеры реализации прокси-сервера

**Squid** — программный пакет, реализующий функцию кэширующего прокси-сервера для протоколов HTTP, FTP, Gopher и (в случае соответствующих настроек) HTTPS. Разработан сообществом как программа с открытым исходным кодом. Все запросы выполняет как один неблокируемый процесс ввода-вывода. Используется в UNIX-подобных системах и в ОС семейства Windows NT [6].

Одной из особенностей squid является возможность работать в режиме обратного прокси. В этом случае вместо кэширования запросов нескольких пользователей к множеству сайтов кэшируются запросы множества пользователей к нескольким сайтам. В этом режиме принятый запрос проверяется на «динамичность» (необходимость каждый раз обрабатывать запрос с нуля) и «возраст» (актуальность данных). Если данные ещё актуальны и не поменялись, то запрос не передаётся серверу, а отдаётся из кэша. Таким образом существенно снижается нагрузка на серверы.



В сочетании с некоторыми межсетевыми экранами и маршрутизаторами Squid может работать в режиме прозрачного прокси. Таким образом, администратору прозрачного прокси-сервера не нужно настраивать каждую клиентскую машину для работы с прокси, однако, в режиме прозрачности не проксируются FTP- и HTTPS-запросы [6].

**SurfSecure** — отечественный шлюз информационной безопасности, позволяет решать задачи по URL-фильтрации, управлению интернет-трафиком, контролю приложений, защите от вредоносных объектов. Помимо этого, SurfSecure выполняет функцию классического прокси-сервера. Чтобы расширить возможности аутентификации пользователей, SurfSecure может активно взаимодействовать с Active Directory, а также интегрироваться с другими системами обеспечения информационной безопасности через ICAP. В результате сотрудники могут безопасно работать в Сети, а руководители — контролировать рабочий процесс. Интерес к SurfSecure может вызвать балансировка нагрузки, которая совместно с другими функциональными возможностями формирует универсальное решение [7].

Основные функциональные характеристики:

- Антивирусная защита, основанная на программном ядре Kaspersky.
- Обширная URL-фильтрация (включая SSL-фильтрацию).
- Идентификация и блокирование приложений с использованием более чем 600 протоколов.
- Возможность блокирования различного вредоносного программного обеспечения, приложений для удалённого администрирования, HTTP-туннелей, VPN, P2P, VoIP и др.
- Регулирование нагрузки с использованием балансировщика.
- Гибкое управление политиками доступа в сеть Интернет.
- Возможность установки ограничений на загрузку файлов определённого типа и размера.
- Кеширование запросов для управления скоростью доступа к веб-ресурсам [7].

#### 1.4 Обзор и обоснование выбора инструментальных средств разработки

Реализация поставленной задачи требует разработки программного продукта для обеспечения перенаправления трафика через прокси-сервер. Для этого было выбрано 3 языка программирования C#, C++ и Python и составлена их сравнительная характеристика, представленная в таблице ниже:

Таблица 1 — Сравнительная характеристика языков программирования

Критерий сравнения	C#	C++	Python
Поддержка ООП	+	+	+
Сетевое программирование	+	+	+
Статическая типизация	+	+	-
Автоматический сбор мусора	+	-	+
Компилируемый язык	+	+	-

По результатам сравнения (таблица 1) и учитывая, что для реализации простейшей модели проху-сервера не требуется гибкое распределение памяти и возможности более низкоуровневого программирования, был выбран язык C#, располагающий автоматическим сборщиком мусора и распределением используемой памяти. Также данный язык программирования имеет сравнительно невысокий порог вхождения, достаточно часто используется для реализации задач, похожих на задачи данного курсового проекта, а значит располагает необходимыми инструментами и библиотеками.

В качестве среды разработки были отобраны Visual Studio 2019 и Sublime Text. Sublime Text является простым редактором кода с возможностями автодополнения и выбора используемого языка, но без возможности отладки программ без установки дополнительных расширений.

Для данного курсового проекта в качестве среды разработки была выбрана Visual Studio 2019 от компании Microsoft, так как она имеет максимальную совместимость с выбранным языком программирования, позволяет отлаживать программы, располагает множеством инструментов для быстрой и качественной разработки, а также включает в себя все возможности Sublime Text.

Одной из задач данного курсового проекта является обеспечение возможности переноса программного продукта на другие компьютеры. Для выполнения данной задачи необходимо проанализировать возможности современных пакетов программ, обеспечивающих создание установочных файлов, иными словами, упаковку программного комплекса в специальных архив, запустив который пользователь сможет установить программу на своём компьютере. Далее приведена сравнительная таблица отобранных для анализа пакетов программ.

Таблица 2 — Сравнительная характеристика программ-установщиков

<b>Критерий сравнения</b>	<b>Clickteam Install Creator 2</b>	<b>Actual Installer</b>	<b>Advanced Installer</b>
Пошаговое создание	+	+	+
Мультиязычность	-	+	+
Создание и выпуск обновлений	-	+	+
Использование алгоритмов компрессии	+	+	+
Проверка зависимостей	-	+	+

По результатам сравнения (таблица 2) наиболее подходящими пакетами программ являются Actual Installer и Advanced Installer. В силу того, что для данного проекта не требуются специализированные функции и возможности, предоставляемые Advanced Installer, был выбран пакет программ Actual Installer.

## 2 ОСНОВНАЯ ЧАСТЬ

### 2.1 Определение структурной (иерархической) схемы решения задачи

В рамках данного курсового проекта необходимо спроектировать простейший прокси-сервер для моделирования работы посредника между пользователем и целевым сервером. Необходимо реализовать требуемые возможности, изображенные в виде иерархической схемы ниже.



Рисунок 5 — Иерархическая схема решения задачи

### 2.2 Определение основных этапов проектирования

Среди этапов проектирования приложения выделены следующие:

- анализ требований и составление иерархической схемы решения задачи;
- анализ требуемых методов решения задачи;
- выбор программного обеспечения для реализации программного продукта;
- разработка программного обеспечения;
  - разработка интерфейсной составляющей проекта;
  - разработка алгоритмов работы программного продукта
  - обеспечение корректности соединения с удаленным сервером;
  - обработка исключительных ситуаций;
- тестирование программного продукта;
- составление технической документации по проекту.

### 2.3 Анализ ограничений и исключительных ситуаций для алгоритмов

В процессе разработки программного продукта возникал ряд исключительных ситуаций, требующих обработки. Основные ограничения введены для обеспечения корректности подключения к требуемому удаленному серверу через разрабатываемый прокси-сервер. Для поиска хоста и порта было создано регулярное выражение, обеспечивающее выборку требуемых данных из заголовка веб-страницы. Также были обработаны различные ошибки, возникающие при обращении к серверу, например, ошибка,

возникающая при отправке данных удаленному серверу или ошибка, возникающая ввиду отсутствия протокола HTTP на удаленном сервере.

## 2.4 Разработка основных алгоритмов задачи

Основным алгоритмом программы является обеспечение обработки запроса пользователя на подключение к удаленному веб-серверу. Приложение определяет, имеются ли ожидающие запросы на подключение. Данная функция обеспечивается циклично методом для опроса базового экземпляра сокета на предмет запросов на подключение клиента. В силу того, что сервер располагается на локальном компьютере и на нем же выполняет работу, данный метод использует локальный IP адрес 127.0.0.1 и порт 8888.

Когда сервер получает запрос для обработки, происходит выделение отдельного потока для данного запроса и начинается его обработка. В теле основного алгоритма программы происходит получение и обработка тела запроса. Это необходимо для получения имени хоста и номера удалённого порта для подключения. Если же номер порта отсутствует, то по умолчанию используется 80 порт. Далее по хосту происходит получение IP, создается точка доступа и запрос передается сокету. В случае успешной работы происходит получение и отправка ответа пользователю, иначе сервер выводит сообщение о возникшей в ходе работы ошибке.

Специальных математических методов решение поставленной задачи не требует.

## 2.5 Разработка архитектуры программы

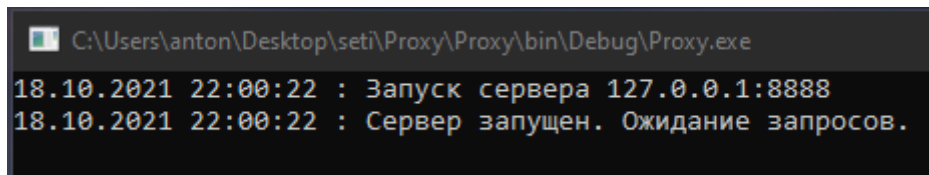
Программный продукт состоит из 1 класса, который содержит в себе логику работы основного алгоритма. В таблице 3 представлены основные методы класса.

Таблица 3 — Основные методы приложения

Название класса	Название метода	Назначение
Programm	ExecuteRequest	Обработка запроса пользователя. Отвечает за создание соединения и получение ответа от сервера.
	ReadToEnd	Метод получения данных из открытого сокета. Обрабатывает запрос и полученный ответ.
	WriteLog	Вывод информационных сообщений о состоянии сервера на экран в отформатированном виде
	Main	Главный метод, определяет IP и порт сервера, обеспечивает работу цикла мониторинга запросов пользователя

## 2.6 Разработка дисплейных фрагментов

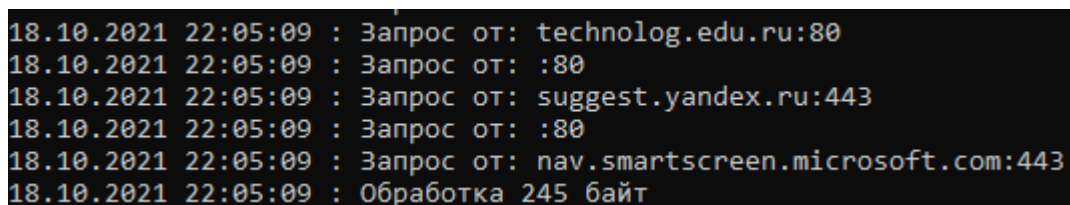
При первом запуске программного продукта появляется рабочая область с информацией об IP адресе и номере порта, на котором был запущен прокси-сервер, а также сообщение о готовности принять запрос пользователя (рисунок 6). Также все сообщения сопровождаются датой и временем обращения.



```
C:\Users\anton\Desktop\seti\Proxy\Proxy\bin\Debug\Proxy.exe
18.10.2021 22:00:22 : Запуск сервера 127.0.0.1:8888
18.10.2021 22:00:22 : Сервер запущен. Ожидание запросов.
```

Рисунок 6 — Запуск прокси-сервера

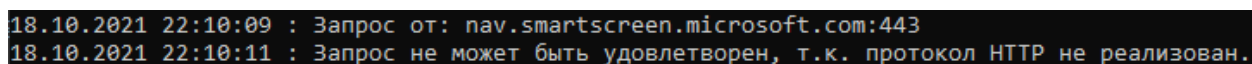
При появлении запросов сервер начинает работу по их обработке. В процессе работы выводятся информационные сообщения с именем хоста, номером порта и количеством обрабатываемых байт (рисунок 7).



```
18.10.2021 22:05:09 : Запрос от: technolog.edu.ru:80
18.10.2021 22:05:09 : Запрос от: :80
18.10.2021 22:05:09 : Запрос от: suggest.yandex.ru:443
18.10.2021 22:05:09 : Запрос от: :80
18.10.2021 22:05:09 : Запрос от: nav.smartscreen.microsoft.com:443
18.10.2021 22:05:09 : Обработка 245 байт
```

Рисунок 7 — Фрагмент работы программы

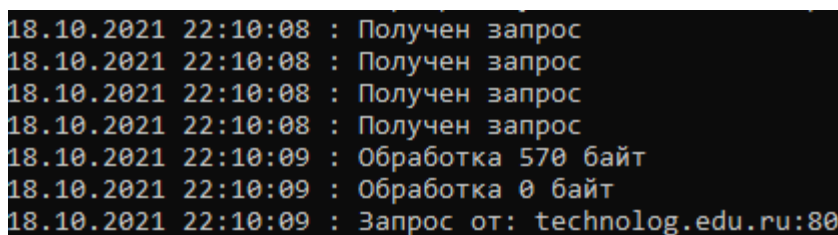
Если же целевой сервер не имеет реализацию HTTP, сервер сообщает об этом в виде сообщения, показанного на рисунке 8.



```
18.10.2021 22:10:09 : Запрос от: nav.smartscreen.microsoft.com:443
18.10.2021 22:10:11 : Запрос не может быть удовлетворен, т.к. протокол HTTP не реализован.
```

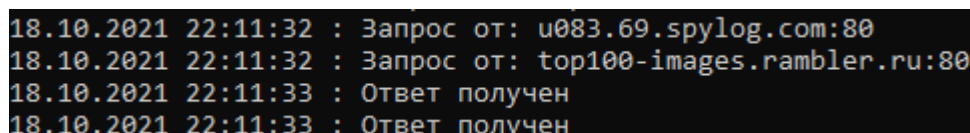
Рисунок 8 — Отсутствие реализации HTTP протокола

Приложение также может обрабатывать несколько запросов одновременно, выводя соответствующие сообщения (рисунок 9, 10).



```
18.10.2021 22:10:08 : Получен запрос
18.10.2021 22:10:08 : Получен запрос
18.10.2021 22:10:08 : Получен запрос
18.10.2021 22:10:08 : Получен запрос
18.10.2021 22:10:09 : Обработка 570 байт
18.10.2021 22:10:09 : Обработка 0 байт
18.10.2021 22:10:09 : Запрос от: technolog.edu.ru:80
```

Рисунок 9 — Получено несколько запросов



```
18.10.2021 22:11:32 : Запрос от: u083.69.spylog.com:80
18.10.2021 22:11:32 : Запрос от: top100-images.rambler.ru:80
18.10.2021 22:11:33 : Ответ получен
18.10.2021 22:11:33 : Ответ получен
```

Рисунок 10 — Получены запросы от разных хостов

## 2.7 Отладка программного комплекса

Отладка программного комплекса производилась путем проверки поведения различных частей программного кода при изменяющихся входных параметрах и возмущающем воздействии. В процессе отладки были выявлены, локализованы и устранены следующие ошибки:

- ошибки, возникающие при нескольких запросах к проху-серверу;
- ошибки, возникающие отсуствии реализации HTTP протокола;
- ошибки, возникающие при отправки данных удаленному серверу;
- некорректная работа главного алгоритма программы.

## 2.8 Тестирование разработанного программного продукта

Для тестирования программного продукта был выбран набор сайтов, имеющих реализацию HTTP протоколов и произведена загрузка веб-страниц для проверки корректности работы приложения. Полученные результаты работы совпали с ожидаемыми результатами. Для начала необходимо настроить в операционной системе использование прокси сервера для подключения к Интернету (рисунок 11).

**Настройка прокси вручную**

Использование прокси-сервера для подключений к Ethernet или сетям Wi-Fi. Эти параметры не применяются для VPN-подключений.

Использовать прокси-сервер

☒ Вкл.

Адрес: 127.0.0.1      Порт: 8888

Не использовать прокси-сервер для адресов, которые начинаются с указанных ниже записей. Для разделения записей используйте точку с запятой (;).

☐ Не использовать прокси-сервер для локальных (внутрисетевых) адресов

Сохранить

Рисунок 11 — Настройка подключения к проху-серверу

```

18.10.2021 22:21:51 : Получен запрос
18.10.2021 22:21:51 : Получен запрос
18.10.2021 22:21:51 : Получен запрос
18.10.2021 22:21:52 : Получен запрос
18.10.2021 22:21:52 : Получен запрос
18.10.2021 22:21:52 : Получен запрос
18.10.2021 22:21:52 : Получен запрос
18.10.2021 22:21:52 : Обработка 245 байт
18.10.2021 22:21:52 : Запрос от: suggest.yandex.ru:443
18.10.2021 22:21:52 : Обработка 0 байт
18.10.2021 22:21:52 : Обработка 0 байт
18.10.2021 22:21:52 : Запрос от: :80
18.10.2021 22:21:52 : Запрос от: :80
18.10.2021 22:21:53 : Обработка 493 байт
18.10.2021 22:21:53 : Обработка 424 байт
18.10.2021 22:21:53 : Запрос от: top100-images.rambler.ru:80
18.10.2021 22:21:53 : Обработка 433 байт
18.10.2021 22:21:53 : Обработка 425 байт
18.10.2021 22:21:53 : Запрос от: u083.69.spylog.com:80
18.10.2021 22:21:53 : Запрос от: top.list.ru:80
18.10.2021 22:21:53 : Запрос от: counter.rambler.ru:80
18.10.2021 22:21:54 : Ответ получен
18.10.2021 22:21:54 : Ответ получен
18.10.2021 22:21:54 : Ответ получен

```

При работе проху-сервера было отмечено, что происходит получение и обработка не только пользовательского запроса, но и побочных запросов, например, рекламы и сбора статистики.

Рисунок 12 — Пример работы проху-сервера

При обновлении загруженной страницы сервер получает новые запросы и обрабатывает их (рисунок 13).

```

18.10.2021 22:24:09 : Получен запрос
18.10.2021 22:24:09 : Получен запрос
18.10.2021 22:24:10 : Обработка 545 байт
18.10.2021 22:24:10 : Обработка 0 байт
18.10.2021 22:24:10 : Запрос от: lib.ru:80
18.10.2021 22:24:10 : Запрос от: :80
18.10.2021 22:24:11 : Ответ получен

```

Рисунок 13 — Обновление загруженной страницы

В ходе тестирования запрашиваемая веб-страницы была загружена. Также у пользователя есть возможность использовать гиперссылки на загруженной странице, что порождает новые запросы к проху-серверу (рисунок 14).

```
18.10.2021 22:26:42 : Получен запрос
18.10.2021 22:26:42 : Получен запрос
18.10.2021 22:26:42 : Получен запрос
18.10.2021 22:26:43 : Обработка 517 байт
18.10.2021 22:26:43 : Обработка 0 байт
18.10.2021 22:26:43 : Запрос от: lib.ru:80
18.10.2021 22:26:43 : Запрос от: :80
18.10.2021 22:26:43 : Обработка 95 байт
18.10.2021 22:26:43 : Запрос от: nav.smartscreen.microsoft.com:443
18.10.2021 22:26:45 : Ответ получен
18.10.2021 22:26:46 : Запрос не может быть удовлетворен, т.к. протокол HTTP не реализован.
```

Рисунок 14 — Пример работы приложения при перемещении по сайту при помощи ссылок на другие страницы.

Однако как видно из рисунка 14 не всегда все содержимое сайта может быть загружено, так как отдельные его элементы (реклама и сбор статистических данных) могут не иметь реализацию HTTP протокола.



## ВЫВОДЫ ПО РАБОТЕ

В процессе выполнения данной курсовой работы был создан пример простейшего проху-сервера, получены навыки работы с сокетами и протоколом TCP, создан сервер способный обрабатывать запросы к удаленному хосту. В ходе разработки были выполнены следующие задачи:

- обоснование выбора инструментальных средств разработки;
- разработка структурной схемы решения задачи;
- анализ ограничений и исключительных ситуаций;
- разработка основных алгоритмов программы;
- разработка архитектуры и дисплейных фрагментов приложения;
- создание и отладка модулей программного комплекса;
- тестирование разработанного программного продукта;
- разработка эксплуатационного документа «Руководство системного программиста»;
- оформление отчетной документации по проекту.

Перспективы развития приложения предполагают обеспечения большего количества поддерживаемых протоколов, а также функцию кэширования данных для обеспечения более высокого быстродействия разработанного приложения.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Таненбаум, Э. Компьютерные сети / Э. Таненбаум. — 4-е изд. — Санкт-Петербург : Питер, 2006. — 992 с. — ISBN 0-13-066102-3.
- 2 Галкин, В.А. Телекоммуникации и сети: Учеб. пособие для вузов. / В.А. Галкин — Москва : Изд-во МГТУ им. Н.Э. Баумана. — 2005. — 608 с. — ISBN 5-7038-1961-X
- 3 Сергеев, А. Н. Основы локальных компьютерных сетей / А. Н. Сергеев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2021. — 184 с. — ISBN 978-5-8114-8260-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/173807> (дата обращения: 09.09.2021). — Режим доступа: для авториз. пользователей.
- 4 Бертсекас, Д. Сети передачи данных. / Д. Бертсекас. Москва : Мир. — 2002. — 544 с. — ISBN 5-03-000639-7.
- 5 Смелянский, Р. Л. Компьютерные сети. Сети ЭВМ. / Р.Л. Смелянский. Москва. — 2011. — 240 с. — ISBN 978-5-7695-7153-4.
- 6 Официальный сайт Squid : сайт. — Санкт-Петербург, 2021 — . — URL: <http://www.squid-cache.org/> (дата обращения 25.09.2021). — Режим доступа: свободный.
- 7 Прoxy-сервер SurfSecure : сайт. — Санкт-Петербург, 2021 — . — URL: <https://www.anti-malware.ru/products/surfsecure> (дата обращения 25.09.2021). — Режим доступа: свободный.
- 8 Ракитин, Р. Ю. Компьютерные сети : учебное пособие / Р. Ю. Ракитин, Е. В. Москаленко. — Барнаул : АлтГПУ, 2019. — 340 с. — ISBN 978-5-.88210-942-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/139182> (дата обращения: 09.09.2021). — Режим доступа: для авториз. пользователей.
- 9 Норенков, И.П. Телекоммуникационные технологии и сети. / И.П. Норенков — Москва : МГТУ им.Н.Э.Баумана, 2005. — 248 с. — ISBN 5-7038-1564-9.

## **ПРИЛОЖЕНИЕ А**