**ГБОУ ВПО «СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ Ханты-Мансийского автономного округа – Югры»**

**Политехнический институт**

**Кафедра информатики и вычислительной техники**

Использование прокси-сервера для организации работы сотрудников компании

Выполнили: Деркач Егор, Имамов Данил, студенты 1 курса по направлению 09.03.02 - Информационные системы и технологии

дисциплина «Основы информационной безопасности»

Руководитель: Воронцова Татьяна Дмитриевна, преподаватель ДПМ МИЭМ НИУ ВШЭ

**Оглавление**

[**Введение 3**](#_Toc185629465)

[**Актуальность 3**](#_Toc185629466)

[**Цель и задачи 4**](#_Toc185629467)

[**Определение и использование прокси 4**](#_Toc185629468)

[**Структура системы 5**](#_Toc185629469)

[**Анализ существующих решений 6**](#_Toc185629470)

[**Реализация проекта 7**](#_Toc185629471)

[**Основные компоненты системы 8**](#_Toc185629472)

[**Результаты 9**](#_Toc185629473)

[**Выводы 11**](#_Toc185629474)

[**Список источников 12**](#_Toc185629475)

# 

# Введение

В современном мире, где интернет стал неотъемлемой частью как личной, так и профессиональной жизни, вопросы контроля и управления сетевым трафиком приобретают все большую значимость. Организации различного масштаба сталкиваются с необходимостью мониторинга сетевой активности, обеспечения безопасности данных и оптимизации использования сетевых ресурсов. В этом контексте прокси-серверы становятся критически важным инструментом, обеспечивающим необходимый уровень контроля и безопасности.

Традиционные решения в области прокси-серверов часто представляют собой либо сложные корпоративные системы, требующие значительных ресурсов для внедрения и обслуживания, либо простые решения с ограниченным функционалом, не предоставляющие достаточных возможностей для мониторинга и управления. Особенно остро стоит вопрос удобного интерфейса управления и мониторинга, который позволил бы эффективно отслеживать и контролировать сетевой трафик в режиме реального времени.

Разработанный проект представляет собой прокси-сервер с веб-интерфейсом, который решает эти проблемы, предоставляя простой, но мощный инструмент для мониторинга и управления сетевым трафиком. Система включает в себя функционал фильтрации доменов, ведения логов доступа и удобный веб-интерфейс для управления всеми аспектами работы прокси-сервера. Особое внимание в проекте уделено простоте использования и эффективности работы, что делает его подходящим решением как для образовательных целей, так и для практического применения в небольших организациях.

**Актуальность**

В то время как существующие решения прокси-серверов часто требуют сложной настройки и обладают избыточным функционалом, наш проект предлагает легковесное решение с интуитивно понятным веб-интерфейсом, что особенно важно для небольших организаций и образовательных учреждений. Исследование методов создания простых в развертывании прокси-серверов с возможностью мониторинга в реальном времени становится особенно актуальным в контексте растущей потребности в контроле сетевого трафика и защите от нежелательного контента. Разработка такой системы позволит создать эффективное решение для организаций, которым не требуется сложная инфраструктура, но необходим базовый контроль и мониторинг сетевого трафика.

**Цель и задачи**

**Цель проекта:** Разработать систему прокси-сервера с веб-интерфейсом для мониторинга и управления сетевым трафиком.

**Задачи:**

1. Проанализировать какие функции нужны для организации компании
2. Разработать основной функционал прокси-сервера с поддержкой HTTP и HTTPS протоколов
3. Создать систему логирования сетевой активности
4. Реализовать механизм фильтрации доменов
5. Разработать веб-интерфейс для мониторинга и управления
6. Обеспечить возможность работы с блокировкой доменов через веб-интерфейс

# Определение и использование прокси

Прокси-сервер является ключевым элементом современной сетевой инфраструктуры, выступая в роли посредника между пользователем и интернетом. По своей сути, это промежуточное звено в цепочке сетевых коммуникаций, через которое проходят все запросы от клиента к целевым серверам и обратно. Такая архитектура создает дополнительный уровень контроля и безопасности, позволяя эффективно управлять сетевым трафиком.

В современном мире существует несколько основных типов прокси-серверов, каждый из которых имеет свои особенности и предназначение. HTTP прокси специализируются на обработке веб-трафика, позволяя кэшировать страницы и фильтровать контент. HTTPS прокси обеспечивают работу с зашифрованным трафиком, что особенно важно для безопасной передачи конфиденциальных данных. Отдельного внимания заслуживают SOCKS прокси, которые работают на более низком уровне и могут обрабатывать трафик различных протоколов, обеспечивая максимальную универсальность.

Применение прокси-серверов в современных сетях многогранно. В первую очередь, они служат инструментом обеспечения безопасности и контроля. Администраторы используют их для мониторинга сетевой активности, фильтрации нежелательного контента и защиты от вредоносного программного обеспечения. Не менее важна роль прокси в оптимизации производительности сети – они могут кэшировать часто запрашиваемый контент, распределять нагрузку между серверами и ускорять загрузку веб-страниц. Кроме того, прокси-серверы широко используются для обеспечения анонимности и приватности в сети, позволяя скрывать реальные IP-адреса пользователей и обходить географические ограничения.

Процесс работы прокси-сервера представляет собой сложную, но хорошо организованную последовательность действий. Когда клиент отправляет запрос, прокси-сервер сначала проверяет его корректность и соответствие установленным правилам фильтрации. Затем происходит определение целевого сервера и, при необходимости, модификация заголовков запроса. После получения ответа от целевого сервера прокси обрабатывает полученные данные и отправляет их клиенту. Важной частью этого процесса является логирование – запись информации о запросах, что позволяет вести статистику использования и отслеживать возможные проблемы.

В разработанном проекте реализован современный прокси-сервер, сочетающий в себе функции HTTP и HTTPS прокси. Особое внимание уделено созданию удобного веб-интерфейса для мониторинга и управления трафиком, что делает систему доступной для использования даже неспециалистами. При разработке учитывались современные требования к безопасности и производительности, что позволило создать эффективное решение для контроля сетевого трафика.

**Структура системы**

Система состоит из следующих основных компонентов:

1. **Прокси-сервер (simple\_proxy.py)**
   * Обработка HTTP/HTTPS соединений
   * Система фильтрации доменов
   * Логирование доступа
   * Управление соединениями
2. **Веб-интерфейс (web/)**
   * Страница мониторинга логов
   * Интерфейс управления блокировкой доменов
   * Real-time обновления через WebSocket
3. **Система управления данными (constants.py)**
   * Управление конфигурацией
   * Работа со списком заблокированных доменов
   * Определение системных констант
4. **Система логирования (sites.txt)**
   * Запись событий в файл
   * Фильтрация системных доменов
   * Временная агрегация событий

# Анализ существующих решений

Современный рынок прокси-серверов представлен множеством решений, каждое из которых имеет свои особенности и область применения. Рассмотрим наиболее значимые из них для понимания текущего состояния отрасли и выявления потребностей, которые остаются неудовлетворенными.

Squid является одним из старейших и наиболее известных прокси-серверов с открытым исходным кодом. Это мощное решение предлагает широкие возможности кэширования и фильтрации контента, но его настройка требует глубоких технических знаний. Администраторам необходимо работать с конфигурационными файлами через командную строку, что создает высокий порог входа для новых пользователей. При этом Squid не предоставляет встроенного веб-интерфейса для мониторинга и управления, что затрудняет его использование в небольших организациях.

NGINX, изначально разработанный как веб-сервер, также предлагает функциональность прокси-сервера. Его главными преимуществами являются высокая производительность и масштабируемость. Однако, как и Squid, NGINX требует настройки через конфигурационные файлы и командную строку. Хотя существуют сторонние решения для веб-управления NGINX, они часто сложны в установке и настройке.

Особого внимания заслуживает Privoxy - легковесный прокси-сервер, ориентированный на фильтрацию контента и защиту приватности. Несмотря на наличие веб-интерфейса, его функциональность ограничена базовыми настройками, а внешний вид интерфейса устарел и не соответствует современным стандартам удобства использования.

mitmproxy представляет собой интересное решение для анализа и модификации HTTP/HTTPS трафика. Этот инструмент предоставляет мощные возможности для разработчиков и тестировщиков, но его консольный интерфейс делает его неподходящим для повседневного использования в качестве корпоративного прокси-сервера.

При анализе существующих решений становится очевидным, что на рынке существует пробел между простыми, но ограниченными по функциональности прокси-серверами, и мощными, но сложными в настройке корпоративными решениями. Большинство существующих продуктов либо не имеют современного веб-интерфейса, либо предоставляют ограниченные возможности для мониторинга и управления.

Отдельно стоит отметить проблему визуализации данных о сетевом трафике. В то время как многие решения собирают подробную статистику, они часто предоставляют эту информацию в сыром виде или через сложные системы логирования, что затрудняет оперативный анализ и принятие решений.

Наш проект направлен на заполнение этого пробела путем создания прокси-сервера с современным веб-интерфейсом, который сочетает в себе простоту использования с достаточной функциональностью для решения типовых задач мониторинга и управления сетевым трафиком. Особый акцент сделан на удобстве использования и наглядном представлении данных, что делает систему доступной для широкого круга пользователей.

**Реализация проекта**

Проект реализован на языке Python с использованием следующих технологий:

1. **Основной функционал:**
   * Python 3.8+
   * Socket programming
   * Threading
2. **Веб-интерфейс:**
   * Flask
   * WebSocket (Flask-SocketIO)
   * HTML/CSS
3. **Хранение данных:**
   * JSON для конфигурации
   * Текстовые файлы для логов

# Основные компоненты системы

Разработанная система прокси-сервера представляет собой комплексное решение, состоящее из нескольких взаимосвязанных компонентов. Каждый компонент выполняет свою специфическую роль, обеспечивая надежную и эффективную работу всей системы. Рассмотрим основные компоненты:

Основные модули системы:

* simple\_proxy.py - ядро прокси-сервера
* app.py - веб-интерфейс на базе Flask
* constants.py - система управления данными
* main.py - запуск и координация работы компонентов

Центральным элементом системы является прокси-сервер, реализованный в модуле simple\_proxy.py. Этот компонент отвечает за обработку сетевых соединений и управление трафиком. Он поддерживает как HTTP, так и HTTPS протоколы, что позволяет обрабатывать практически весь веб-трафик. Особенностью реализации является использование многопоточной архитектуры, что обеспечивает эффективную обработку множества одновременных соединений.

Функциональные возможности прокси-сервера:

* Обработка HTTP и HTTPS протоколов
* Многопоточная обработка соединений
* Система фильтрации доменов
* Логирование сетевой активности

Веб-интерфейс, реализованный на базе Flask в модуле app.py, предоставляет удобные инструменты для мониторинга и управления системой. Он включает в себя:

Основные страницы веб-интерфейса:

* Страница мониторинга с real-time обновлением данных
* Страница управления блокировкой доменов
* Просмотр истории событий и логов

Система управления данными, реализованная в модуле constants.py, играет важную роль в обеспечении согласованной работы всех компонентов. Этот модуль определяет основные константы и пути к файлам, а также содержит класс DomainManager, отвечающий за работу со списком заблокированных доменов.

Особенности системы логирования:

* Фильтрация служебного трафика
* Агрегация похожих событий
* Хранение в удобном текстовом формате
* Real-time отображение в веб-интерфейсе

Взаимодействие между компонентами организовано через четко определенные интерфейсы. Прокси-сервер обращается к системе управления данными для проверки заблокированных доменов и записи логов. Веб-интерфейс, в свою очередь, использует эти же компоненты для отображения информации и управления настройками.

Технические особенности реализации:

* Строгая типизация данных
* Обработка исключительных ситуаций
* Модульная архитектура
* Документированный код
* Использование современных веб-технологий

Все компоненты системы разработаны с учетом современных практик программирования. Код хорошо структурирован и документирован, что облегчает его понимание и модификацию. Использование типизации и обработки исключений повышает надежность системы, а применение современных веб-технологий в интерфейсе делает его удобным и отзывчивым.

# Результаты

В результате выполнения проекта была разработана система прокси-сервера с веб-интерфейсом для мониторинга и управления сетевым трафиком. Основные компоненты системы реализованы в соответствии с поставленными задачами.

Реализованные компоненты системы:

1. **Основной функционал прокси-сервера:**
   * Обработка HTTP/HTTPS соединений
   * Многопоточная архитектура
   * Система фильтрации доменов
   * Базовое логирование событий
2. **Веб-интерфейс управления:**
   * Мониторинг активности в реальном времени
   * Управление списком блокировки доменов
   * Просмотр истории событий
   * Интерактивные уведомления
3. **Система логирования:**
   * Фильтрация служебного трафика
   * Объединение похожих событий
   * Сохранение истории доступа
   * Форматирование записей для анализа
4. **Управление данными:**
   * Хранение конфигурации в JSON
   * Работа со списком блокировки
   * Управление настройками сервера
   * Обработка пользовательских правил

В процессе тестирования система обработала значительное количество различных доменов, что отражено в журнале Sites.txt. Список заблокированных доменов хранится в файле blocked\_domains.json и может редактироваться через веб-интерфейс.

Использованные технологии:

* Flask для веб-интерфейса
* WebSocket для обновления данных
* JSON для хранения конфигурации
* Python для основной логики

Система не требует сложной настройки при развертывании, основное управление осуществляется через веб-интерфейс. Модульная архитектура предусматривает возможность дальнейшей доработки и расширения функционала.

# Выводы

Разработанная система прокси-сервера с веб-интерфейсом позволяет решать основные задачи по мониторингу и контролю сетевого трафика. В ходе работы над проектом были достигнуты все поставленные цели.

В первую очередь, реализован базовый функционал прокси-сервера с поддержкой HTTP и HTTPS протоколов. Многопоточная архитектура обеспечивает обработку параллельных соединений, а система фильтрации позволяет контролировать доступ к различным доменам.

Разработанный веб-интерфейс предоставляет необходимые инструменты для управления системой. Реализация WebSocket обеспечивает обновление данных в реальном времени, что упрощает мониторинг сетевой активности. Интерфейс управления блокировкой доменов позволяет эффективно контролировать доступ к ресурсам.

Система логирования с фильтрацией служебного трафика и объединением похожих событий обеспечивает необходимый уровень контроля над сетевой активностью. Хранение данных в текстовом формате упрощает их последующий анализ и обработку.

Модульная архитектура проекта и использование современных технологий создают основу для дальнейшего развития системы. Возможные направления развития включают расширение функционала фильтрации, добавление статистики использования и оптимизацию производительности.

В целом, проект демонстрирует возможность создания практичного инструмента для мониторинга и управления сетевым трафиком, который может быть полезен в образовательных целях и небольших организациях.

# Список источников

1. RFC 7230-7235 - Hypertext Transfer Protocol (HTTP/1.1): Спецификация протокола HTTP, 2014.
2. RFC 7540 - Hypertext Transfer Protocol Version 2 (HTTP/2), 2015.
3. RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3, 2018.
4. Официальная документация Python: Socket Programming HOWTO. https://docs.python.org/3/howto/sockets.html
5. Официальная документация Flask: Web Development Framework. https://flask.palletsprojects.com/
6. Официальная документация Flask-SocketIO: WebSocket интеграция для Flask. https://flask-socketio.readthedocs.io/
7. Mozilla Developer Network (MDN) - HTTP протокол: Руководство по протоколу HTTP. https://developer.mozilla.org/en-US/docs/Web/HTTP
8. «Understanding Network Hacks: Attack and Defense with Python», B. Ballmann, 2015.
9. «Hands-On Network Programming with Python», S. Mistry, 2019.
10. «Building Applications with Flask», I. Muthukadan, 2021.
11. Документация WebSocket API: Спецификация протокола WebSocket. https://websockets.spec.whatwg.org/
12. Документация Python Threading: Руководство по многопоточному программированию. https://docs.python.org/3/library/threading.html
13. OWASP Security Guidelines: Рекомендации по безопасности прокси-серверов. https://owasp.org/www-project-web-security-testing-guide/