Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Операционные среды и системное программирование

К защите допустить:
И.о. заведующего кафедрой
информатики
_____С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

ОБЗОР СЕТИ, ОБНАРУЖЕНИЕ КОМПЬЮТЕРОВ, ГРУПП, РЕСУРСОВ; ОГРАНИЧЕННОЕ УПРАВЛЕНИЕ УЗЛАМИ

БГУИР КП 1-40 04 01 011 ПЗ

Студент Д. А. Демидова

Руководитель С. И. Сиротко

Минск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире сети играют ключевую роль в повседневной деятельности как частных лиц, так и организаций. Они предоставляют возможность обмена данными, доступа к ресурсам, а также обеспечивают коммуникацию между устройствами на различных уровнях. Однако с увеличением размеров и сложности сетей возникают новые вызовы, связанные с обнаружением устройств в сети, контролем и управлением ими, а также обеспечением безопасности и эффективности сетевых операций.

В контексте этой проблематики данный курсовой проект представляет собой важное исследование и практическую разработку инструментов, направленных на повышение эффективности управления сетями. Он нацелен на изучение методов обнаружения устройств в сети, выявления и анализа их характеристик, таких как IP-адреса, MAC-адреса, открытые порты, а также на разработку механизмов для ограничения доступа и управления данными узлами сети.

Целью данного курсового проекта является разработка программы, осуществляющей обзор сети.

Исходя из цели проекта был составлен следующий перечень задач:

1 Изучить информацию о характеристиках сети и способах их обнаружения.

2 Определить функционал программы.

3 Подготовить окружение и программу, которая будет производить обзор сети.

4 Продемонстрировать работу программы на реальных примерах.

Результатом выполнения данного курсового проекта будет десктопное приложение, позволяющее пользователю получить информацию о текущей сети. Данный проект будет способствовать более глубокому пониманию устройства сети, смысла и взаимосвязи ее характеристик.

1 ТЕОРЕТИЧЕСКИЙ ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

Сетевой протокол – это набор правил, определяющий принципы взаимодействия устройств в сети. Чтобы отправка и получение информации прошли успешно, все устройства-участники процесса должны принимать условия протокола и следовать им. В сети их поддержка встраивается или в аппаратную часть (в «железо»), или в программную часть (в код системы), или и туда, и туда.

Для взаимодействия протоколов между собой существует модель OSI, или Open Systems Interconnection. Дословно название переводится как «взаимодействие открытых систем».

OSI – эталонная модель взаимодействия устройств в сети. Модель OSI – это модель, позволяющая разным системам связи коммуницировать между собой по общепринятым стандартам. Ее можно сравнить с английским, то есть глобальным, универсальным языком в мире сетей.

Модель основана на принципе разделения коммуникационной системы на семь отдельных уровней представлена на рисунке 1.1.

Семиуровневая модель OSI	
7	Прикладной уровень (application layer)
6	Уровень представления (presentation layer)
5	Сеансовый уровень (session layer)
4	Транспортный уровень (transport layer)
3	Сетевой уровень (network layer)
2	Канальный уровень (data link layer)
1	Физический уровень (physical layer)

Рисунок 1.1 – Семиуровневая модель OSI

Если в передаче информации случаются сбои, модель помогает быстрее и легче локализовать проблему на конкретном уровне и значительно ускорить процесс восстановления работоспособности системы.

Модель OSI является эталонным стандартом, но на данный момент она устарела, поскольку современные протоколы работают сразу на нескольких уровнях модели OSI. На смену модели OSI пришла модель TCP/IP, на основе которой работает большая часть устройств в современном мире.

TCP/IP – модель, на которой работает сеть Интернет

Модель TCP/IP помогает понять принцип работы и взаимодействия узлов в сети Интернет. Ее название включает в себя два основных протокола, на которых построен интернет. TCP/IP расшифровывается как Transmission Control Protocol/Internet Protocol, или протокол управления передачей (данных)/интернет-протокол.

Модель используется во всем современном интернете, новые сетевые протоколы разрабатываются с опорой на модель TCP/IP. Например, подключаясь к сайту Selectel, вы используете протоколы IP, TCP и HTTPS, которые работают в рамках упомянутой модели. Ниже представлено сравнение на рисунке 1.2 модели OSI и модели TCP/IP.

<table>
<tr><th colspan="2">Модель OSI</th><th colspan="2">Модель TCP/IP</th></tr>
<tr><td>Прикладной уровень (application layer)</td><td>7</td><td rowspan="3">4</td><td rowspan="3">Прикладной уровень (application layer)</td></tr>
<tr><td>Уровень представления (presentation layer)</td><td>6</td></tr>
<tr><td>Сеансовый уровень (session layer)</td><td>5</td></tr>
<tr><td>Транспортный уровень (transport layer)</td><td>4</td><td>3</td><td>Транспортный уровень (transport layer)</td></tr>
<tr><td>Сетевой уровень (network layer)</td><td>3</td><td>2</td><td>Межсетевой уровень (internet layer)</td></tr>
<tr><td>Канальный уровень (data link layer)</td><td>2</td><td rowspan="2">1</td><td rowspan="2">Канальный уровень (link layer)</td></tr>
<tr><td>Физический уровень (physical layer)</td><td>1</td></tr>
</table>

Рисунок 1.2 – Модель OSI и модель TCP/IP

Далее рассмотрены основные протоколы межсетевого, транспортного уровней, а также уровня приложений. Именно с ними взаимодействие происходит чаще всего, анализируя какие-либо проблемы в сети или на сервере.

Internet Protocol (IP) – это наиболее простой протокол, объединивший отдельные компьютеры в глобальную сеть. Главной его задачей является маршрутизация дейтаграмм – определение маршрута следования пакетов по узлам сети. Каждое устройство – ПК, принтер и т.д. – имеет IP-адрес, чтобы

данные попадали к нужному адресату. Так, например, отправленный на печать файл не окажется вместо принтера в личном ПК вашего коллеги.

В качестве минусов протокола можно отметить низкую надежность. Он не определяет факт передачи пакета и не контролирует целостность данных. IP просто осуществляет пересылку.

Для пересылки пакетов необходимо определить, на какой порт отправить пакет. Для этого протокол имеет свою систему адресации. В качестве адресов выступает 32-битные (IPv4) или 128-битные (IPv6) адреса. Перед отправкой пакета в него добавляются header (заголовок) и payload (данные для доставки).

IPv4 является 32-разрядной системой, состоящей из четырех разделов (123.123.123.123). Он поддерживает до 4 294 967 296 адресов и является протоколом по умолчанию. Основным его преимуществом является простота. В недостатках – ограниченное адресное пространство, также называемое «исчерпанием адресов».

IPv6, напротив, – 128-битное адресное пространство, которое обеспечивает приблизительно 2^128 степени адресов. Формат записи состоит из восьми разделов, в каждый из которых записывается четыре 16-ричных цифры. Недостаток протокола – в сложности сетевого администрирования. При аренде сервера или виртуальной машины в Selectel выдается IPv4, однако можно запросить и IPv6-адреса, в облаке на базе VMware выдаются только IPv4-адреса.

Один из основных протоколов, который работает поверх IP, – это протокол TCP, из-за чего его часто обозначают как TCP/IP. Но это не единственный протокол, который является частью интернет-протокола.

TCP – протокол обмена сообщениями в сети Интернет. TCP помогает устройствам в сети обмениваться сообщениями. Он работает на четвертом, транспортном, уровне модели OSI.

Для передачи информации происходит дробление исходного файла на части, которые передаются получателю, а далее собираются обратно. Например, человек запрашивает веб-страницу, далее сервер обрабатывает запрос и высылает в ответ HTML-страницу при помощи протокола HTTP. Он, в свою очередь, запрашивает уровень TCP для установки требуемого соединения и отправки HTML-файла. TCP конвертирует данные в блоки, передавая их на уровень TCP пользователя, где происходит подтверждение передачи [1].

Свойства протокола TCP:

1 Система нумерации сегментов. (Segment Numbering System). TCP отслеживает передаваемые или принимаемые сегменты, присваивая номера каждому из них. Байтам данных, которые должны быть переданы, присваивается определенный номер байта, в то время как сегментам присваиваются порядковые номера.

2 Управление потоком. Эта функция ограничивает скорость, с которой отправитель передает данные. Это делается для обеспечения надежности доставки. Получатель постоянно сообщает отправителю о том, какой объем данных может быть получен.

3 Контроль ошибок. Данная функция реализуется для повышения надежности путем проверки байтов на целостность.

4 Порт источника и порт назначения. Протокол TCP использует специальные порты для связи различных протоколов. Например, протокол SSH использует 22й порт, HTTP – 80, HTTPS – 443, Gopher – 70. Все порты делятся на три диапазона – общеизвестные (0–1023), зарегистрированные (1024–49151) и динамические (49152–65535) представлены на рисунке 1.3.

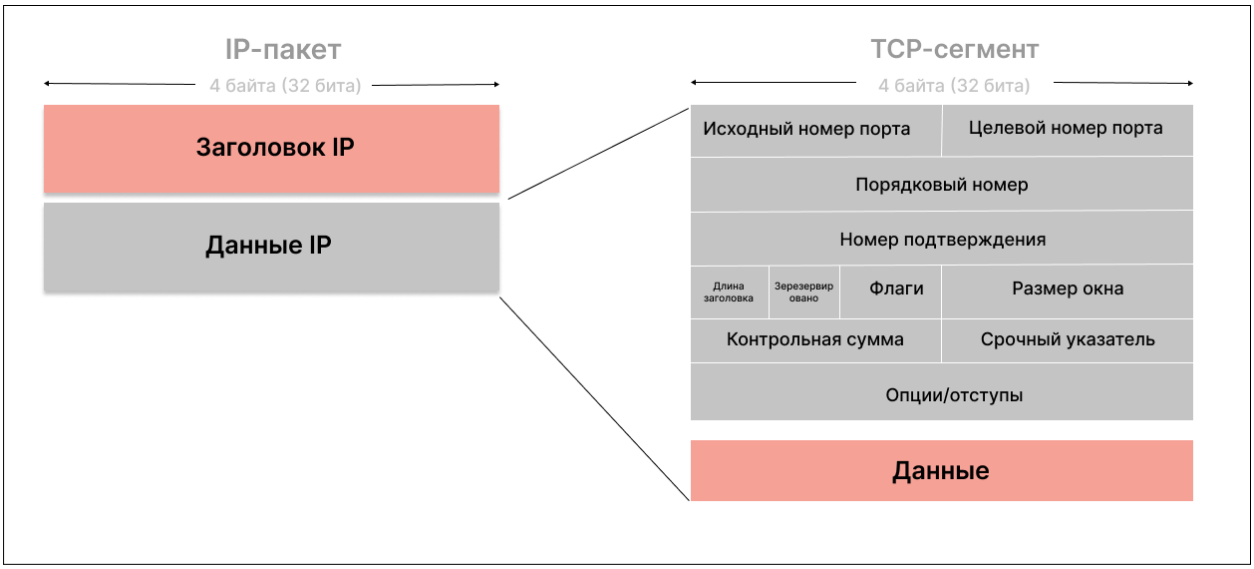

Рисунок 1.3 – TCP сегмент и IP-пакет

В отличие от протокола TCP User Datagram Protocol обеспечивает передачу данных без получения подтверждения от пользователя о результате действия. Благодаря этому достигается большая скорость работы и передачи данных в ущерб надежности и безопасности.

Особенности протокола диктуют специфику его применения. Так, он подходит для приложений, например, Skype, Discord и другие, которые

работают в реальном времени и где задержка передачи данных может быть проблемой. Также его предпочтительно использовать в приложениях с большим количеством подключенных клиентов – например, в играх, голосовых или видеоконференциях, а также при потоковой передаче мультимедиа.

UDP работает путем сбора данных в UDP-пакете и добавления в пакет собственной информации заголовка. Заголовок UDP включает четыре поля, объем которых составляет 2 байта каждый: номер порта источника, номер порта назначения, длина заголовка и контрольная сумма блока. Пример заголовка UDP представлен на рисунке 1.4.

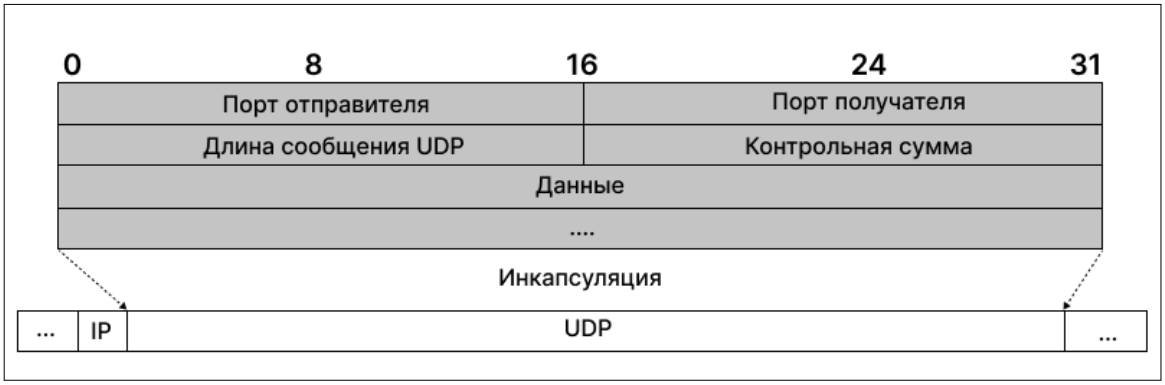

Рисунок 1.4 – Пример заголовка UDP

Протокол UDP любят злоумышленники при организации DDOS – или DOS-атак. Из-за того, что данный протокол не требует подтверждения от сервера, открывается возможность просто «залить» сервер запросами. Стандартная атака подразумевает отправку большого количества дейтаграмм. Это заставляет сервер отвечать на каждый из них, расходуя вычислительные мощности. [2]

Simple Network Management Protocol (SNMP) – это протокол прикладного уровня, он делает возможным обмен данными между сетевыми устройствами.

SNMP – это не продукт, а свод правил. Он определен Советом по архитектуре Интернета и является частью пакета TCP/IP. SNMP управляется и поддерживается Инженерной группой Интернета (IETF).

Протокол позволяет системному администратору проводить мониторинг, контролировать производительность сети и изменять конфигурацию подключенных устройств. SNMP используют в сетях любого

размера: чем крупнее сеть, тем лучше раскрываются преимущества протокола. Он позволяет просматривать, контролировать и управлять узлами через единый интерфейс с функциями пакетных команд и автоматического оповещения.

Таким образом, SNMP избавляет администратора от необходимости ввода команд вручную. Всего были разработаны и развернуты три версии. Все они используются до сих пор, а самой распространенной стала вторая – SNMPv2c.

MIB – это иерархическая база данных со сведениями об устройстве. У каждого типа устройства своя MIB-таблица: у принтера в ней содержится информация о состоянии картриджей, а у коммутатора – данные о трафике. Благодаря MIB менеджер знает, какую информацию он может запросить у агента устройства.

Каждый объект в MIB имеет свой уникальный ID – OID, который представлен в числовом формате и имеет иерархическую структуру. OID – это числовой эквивалент пути к файлу. Он присваивает значения каждой таблице в MIB, каждому столбцу в таблице и каждому значению в столбце.

Часть значений в OID содержит данные о производителе устройства, что позволяет быстро получить определенную информацию о девайсе [3].

Древовидная иерархия MIB и OID в SNMP выглядит несколько запутанной, но у нее есть свои преимущества. Это простая и гибкая система организации сетевых устройств, она работает вне зависимости от размера сети.

Изначально протокол должен был предоставить системным администраторам инструмент для управления интернетом. Однако, гибкая архитектура SNMP позволила проводить мониторинг всех сетевых устройств и управлять ими с одной консоли. Это и стало причиной распространения SNMP.

Таким образом, рассмотренные темы, такие как IP-адресация, протоколы TCP/IP и UDP, SNMP (Simple Network Management Protocol), являются основой для построения сетевых приложений и инструментов управления сетью.

2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Выбор операционной системы

В качестве операционной системы для написания проекта используется Linux. Выбор операционной системы Linux для данного проекта имеет свои преимущества и обоснования. Linux обладает мощными средствами командной строки, которые позволяют выполнять широкий спектр задач, включая работу с сетью, администрирование системы, автоматизацию задач и многое другое. Это делает Linux идеальным выбором для сетевого программирования и управления сетью. Linux имеет богатую экосистему программного обеспечения, включая множество инструментов и библиотек для сетевого программирования, анализа трафика, обнаружения устройств. Это обеспечивает разработчикам широкие возможности для реализации функциональности проекта без необходимости создания всего с нуля [4].

В сравнении с Windows, Linux обладает рядом особенностей, которые делают его более подходящим для данного проекта. Linux позволяет пользователю полностью настраивать и настраивать систему под свои потребности. Это позволяет легко интегрировать различные инструменты и решения для решения конкретных задач. Linux имеет мощные средства для работы с сетью, включая поддержку различных протоколов, инструменты администрирования и мониторинга сети. Это делает его идеальным выбором для сетевых приложений и управления сетью. Linux обеспечивает высокий уровень безопасности благодаря своей архитектуре и подходу к безопасности. Это делает его предпочтительным выбором для работы с сетевыми данными и защиты сетевой инфраструктуры.

В целом, Linux предоставляет разработчикам широкие возможности для реализации проекта по обзору сети, обнаружению компьютеров, групп, ресурсов и ограниченному управлению узлами. Его гибкость, мощные средства командной строки и сетевые возможности делают его отличным выбором для данного типа проекта.

2.2 Выбор платформы для написания программы

В качестве языка программирования для написания программы используется Python. Преимущества использования Python для данного проекта:

1 Python славится своей простотой и доступностью. Его лаконичный и выразительный синтаксис делает язык привлекательным для новичков и опытных разработчиков. Благодаря этому, разработчики могут быстрее освоить язык и приступить к реализации проекта без лишних сложностей.

2 Python обладает обширной библиотекой инструментов для работы с сетями. Библиотеки такие, как Scapy, PySNMP, Nmap и др., предоставляют мощные возможности для создания сетевых приложений, анализа трафика, обнаружения устройств и многое другое. Это существенно упрощает реализацию функциональности проекта без необходимости писать все с нуля.

3 Python предоставляет удобные инструменты для отладки и разработки, что облегчает создание и отладку кода, работающего с потоками и процессами.

4 Python поддерживается на различных операционных системах, включая Windows, macOS и различные дистрибутивы Linux. Это позволяет создавать кроссплатформенные приложения, которые могут работать на различных платформах без изменений в исходном коде [5].

5 Python имеет огромное и активное сообщество разработчиков по всему миру. Это обеспечивает доступ к множеству ресурсов, форумов, онлайн-курсов и документации, которые помогают в решении любых проблем и вопросов, возникающих в процессе разработки.

В качестве платформы для разработки используется PyCharm. Главная причина этому то, что PyCharm является одной из наиболее популярных и мощных IDE для разработки на Python, предоставляя разработчикам множество инструментов и удобств для повышения производительности и качества кода. PyCharm предоставляет разработчику широкий набор инструментов, которые значительно упрощают процесс разработки. Включая функции автодополнения кода, статический анализ, отладчик, систему управления версиями и многое другое. Это позволяет сосредоточиться на написании кода и повысить производительность. PyCharm специально разработан для работы с Python, что делает его идеальным выбором для проектов на этом языке программирования. Он обладает широкой поддержкой стандартных библиотек Python, включая различные фреймворки и инструменты для сетевого программирования. Интерфейс PyCharm интуитивно понятен и удобен в использовании. Он предоставляет удобные инструменты для организации проекта, навигации по коду и выполнения различных задач, что делает процесс разработки более эффективным и приятным.

Исходя из всех этих факторов, Python и PyCharm являются отличным выбором для написания данного курсового проекта. Они обеспечивают простоту, мощные инструменты, гибкость.

Таким образом, Python и Linux обладают обширными возможностями для разработки и администрирования сетей. Python, как язык программирования, предоставляет простой и эффективный способ создания сетевых приложений и автоматизации задач сетевого управления.

Сочетание Python и Linux обеспечивает гибкость и мощные возможности для работы с различными типами сетей и устройств. С помощью Python можно легко осуществлять обзор сети, обнаруживать компьютеры и устройства, а также ограничивать доступ к ресурсам сети. Благодаря богатым библиотекам и инструментам, доступным для Python, разработчики могут реализовывать различные сетевые протоколы и стандарты, такие как TCP/IP, UDP, ICMP, SNMP и другие. [6]

Linux, в свою очередь, является популярной операционной системой для разработки сетевых приложений и администрирования сетевой инфраструктуры.

Python и Linux обеспечивают возможность автоматизации и сценаризации сетевых задач, что позволяет упростить управление сетью и повысить ее эффективность. Благодаря активному сообществу разработчиков и пользователей, существует множество ресурсов и поддержки для работы с Python и Linux в области сетевого программирования и администрирования.

3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

3.1 Обоснование необходимости разработки

Данный проект обосновывается необходимостью эффективного управления и мониторинга сетевой инфраструктуры в современных информационных средах. В современном мире сетевые технологии играют ключевую роль в повседневной деятельности компаний, организаций и домашних пользователей. В этом контексте, создание инструментов и приложений, способных обеспечить надежный и безопасный функционирование сети, является критической задачей.

Современные сети становятся все более сложными и масштабными. Управление такими сетями требует постоянного мониторинга и анализа ее состояния, а также быстрых реакций на возникающие проблемы. Разработка инструментов для мониторинга сетевой активности и администрирования сетевыми ресурсами является необходимой составляющей эффективного управления сетью.

С увеличением числа киберугроз и уязвимостей сетевая безопасность становится все более важной. Отслеживание активности сети, обнаружение аномального трафика и сканирование портов являются важными методами защиты сети от внешних угроз.

Анализ трафика и оценка скорости сети позволяют выявить узкие места и проблемы, которые могут замедлять работу сети. Инструменты для ограничения трафика и управления пропускной способностью могут помочь оптимизировать производительность сети и обеспечить качественное предоставление услуг [7].

Создание простых и интуитивно понятных интерфейсов для управления сетью позволяет сократить время настройки и обслуживания сетевых устройств, а также упростить процесс принятия решений администратором сети.

Все эти факторы подчеркивают значимость разработки инструментов, позволяющих проводить мониторинг, анализ и управление сетевой инфраструктурой. Данный проект нацелен на предоставление комплексного подхода к управлению сетью, что позволит пользователям эффективно контролировать и оптимизировать работу своих сетей.

Таким образом, разработка проекта по обзору сети, обнаружению компьютеров, групп, ресурсов, а также ограниченному управлению узлами представляет собой важную задачу в контексте современных сетевых технологий. Разработанные инструменты и приложения позволяют администраторам сетей эффективно контролировать, анализировать и управлять сетевой инфраструктурой, что способствует повышению безопасности, оптимизации производительности и обеспечению надежности работы сети.

Путем использования средств программирования, таких как Python, и современных сетевых технологий, включая протоколы SNMP, TCP/IP, а также инструменты мониторинга и анализа трафика, проект позволяет пользователям проводить обширный анализ состояния сети, обнаруживать уязвимости, оптимизировать ее производительность и реагировать на изменения в сетевой среде.

Комбинация языка программирования Python и операционной системы Linux для реализации данного проекта обеспечивает высокую гибкость, эффективность и масштабируемость при разработке и эксплуатации сетевых приложений. Это обеспечивает администраторам и инженерам сетей мощные инструменты для управления сетевой инфраструктурой и обеспечивает более высокий уровень безопасности и производительности сетей.

3.2 Технологии программирования, используемые для решения поставленных задач

Для решения задач курсового проекта используются следующие технологии программирования (библиотеки):

1 Tkinter является стандартным набором инструментов для создания графического пользовательского интерфейса (GUI) в Python. Tkinter обеспечивает простой и интуитивно понятный способ создания графического интерфейса без необходимости в обширных знаниях программирования GUI. Она интегрируется хорошо с Python и позволяет создавать кроссплатформенные приложения, которые могут запускаться на различных операционных системах, включая Windows, macOS и Linux. В контексте данного проекта, Tkinter помогает создать главное окно приложения и реализовать различные экраны и функциональность, такие как отображение информации, взаимодействие с пользователем и управление основными операциями.

2 Библиотека ipaddress в Python предоставляет удобные средства для работы с IP-адресами и сетями. Она позволяет создавать, анализировать и манипулировать IP-адресами, а также проводить проверки и сравнения между ними. С помощью ipaddress можно создавать объекты для представления IP-адресов IPv4 и IPv6, а также сетей, подсетей и интервалов адресов. Библиотека ipaddress также предоставляет удобные методы для проверки вхождения адреса в сеть, вычисления подсети по IP-адресу и маске подсети, разбора строкового представления IP-адресов и многое другое [8].

3 Библиотека socket в Python предоставляет удобные средства для работы с сетевыми соединениями и протоколами. С ее помощью можно создавать сокеты для установления TCP, UDP и других типов соединений, а также выполнять различные операции с сетевыми ресурсами. Библиотека позволяет создавать клиентские и серверные приложения, обмениваться данными через сеть, устанавливать и закрывать соединения, а также управлять сетевыми параметрами.

4 Библиотека platform предоставляет средства для получения информации о характеристиках операционной системы, на которой выполняется приложение. С ее помощью можно получать данные о версии ОС, архитектуре процессора, имени узла, версии Python и многое другое. Это позволяет создавать кроссплатформенные приложения, которые могут адаптироваться к различным операционным системам.

5 Библиотека subprocess предоставляет удобные средства для запуска внешних процессов из Python-скриптов. С ее помощью можно выполнять команды в командной строке, запускать исполняемые файлы, обмениваться данными с другими процессами и многое другое [9].

6 Библиотека scapy является мощным инструментом для работы с сетевыми пакетами в Python. Ее гибкость и функциональность позволяют создавать разнообразные сетевые приложения, начиная от простых сканеров сети и анализаторов трафика, и заканчивая сложными инструментами для тестирования безопасности сетей. С использованием scapy можно создавать, отправлять и принимать пользовательские сетевые пакеты на различных уровнях сетевой модели OSI. Это дает возможность проводить разнообразные сетевые операции, такие как сканирование портов, перехват и анализ трафика, а также тестирование сетевой безопасности [10].

7 Библиотека requests предоставляет удобный интерфейс для выполнения HTTP-запросов в Python. С ее помощью можно взаимодействовать с веб-серверами, отправлять запросы на получение

данных, отправку данных и выполнение других действий, связанных с протоколом HTTP [11].

8 Библиотека geopy предоставляет удобные средства для работы с географическими данными в Python. Она позволяет определять координаты местоположений по адресам, расчитывать расстояния между точками, находить ближайшие объекты и многое другое. Это полезный инструмент для создания приложений, связанных с картографией, геолокацией и геоинформационными системами [12].

9 Библиотека speedtest представляет собой инструмент для измерения скорости интернет-соединения в Python. Он позволяет проводить тесты скорости загрузки и выгрузки данных, оценивать стабильность и производительность сетевого соединения. Это полезный инструмент для мониторинга качества интернет-соединения и диагностики сетевых проблем.

10 Библиотека pysnmp предоставляет возможности для работы с протоколом SNMP (Simple Network Management Protocol) в Python. Она позволяет создавать SNMP-клиентов и агентов, осуществлять мониторинг и управление сетевыми устройствами, получать информацию о системе, интерфейсах, процессах и других аспектах сетевой инфраструктуры [13].

Таким образом, используемые технологии программирования позволяют выполнить цели данного курсового проекта.

4 ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ ПРОГРАММЫ

4.1 Меню

При запуске приложения пользователя встречает меню, в котором перечислен доступный функционал (рисунок 4.1).

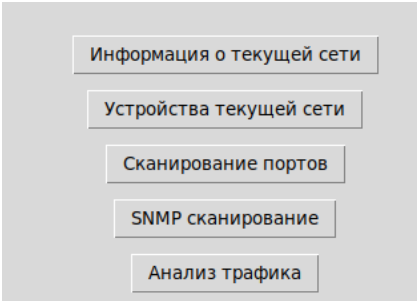

Рисунок 4.1 – Меню приложения

В следующих пунктах следует подробное описание данных функций.

4.2 Информация о текущей сети

При выборе информации о сети пользователь может узнать не только данные локальной сети, но также и публичные, которые доступны пользователям из других сетей (рисунок 4.2).

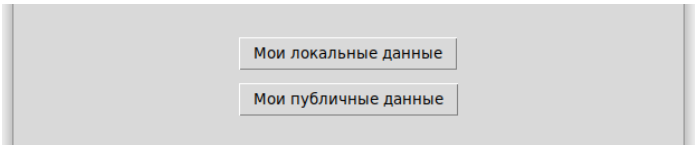

Рисунок 4.2 – Информация о текущей сети

В качестве локальных данных пользователь получает информацию об интерфейсе сети, о своем IP-адресе, адресе роутера, MAC-адресе (рисунок 4.3).

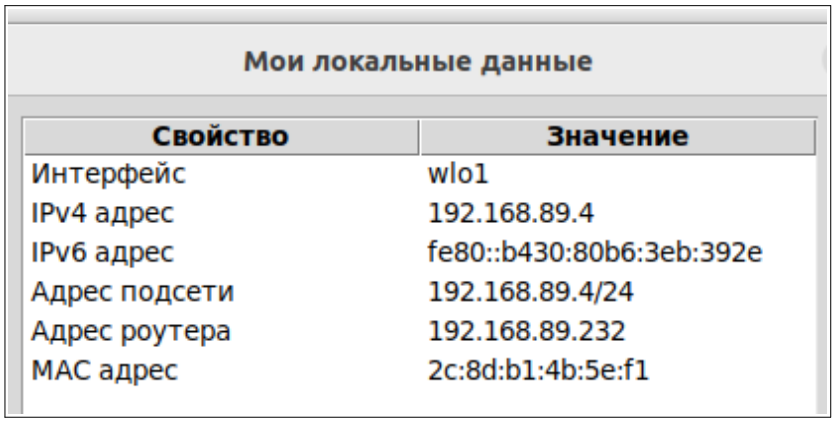

Рисунок 4.3 – Локальные данные

В качестве публичных данных пользователь получает IP-адрес, под которым его видят другие в сети Интернет, а также примерные координаты и адрес, которые может узнать любой человек в сети (рисунок 4.4).

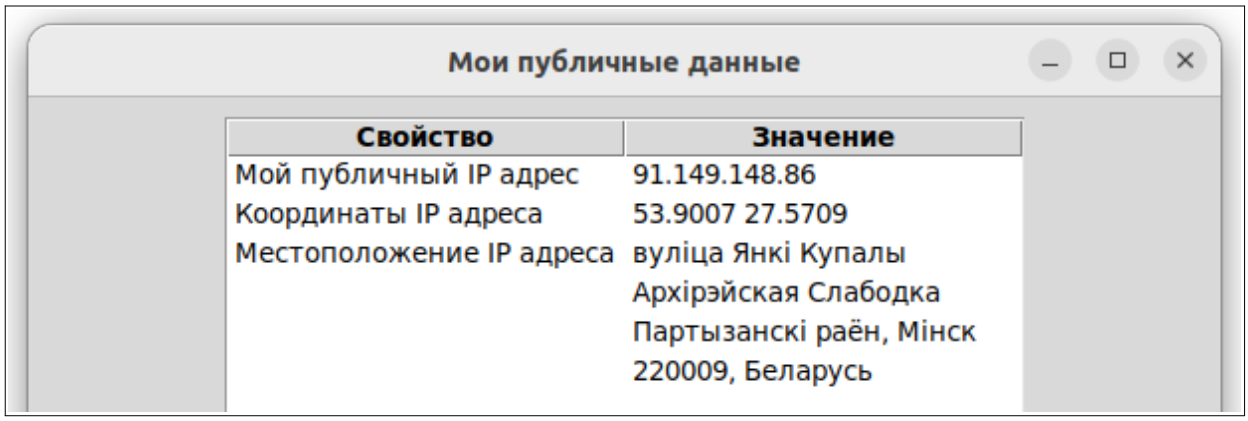

Рисунок 4.4 – Публичные данные

Получение характеристик сети немного отличается на Windows и Linux, поэтому используется библиотека platform, чтобы, узнав операционную систему хоста, применить верный поход для обзора сети.

4.3 Устройства текущей сети

Список устройств, также находящихся в данной подсети выводится в виде таблицы. Пользователь получает информацию об IP- и MAC-адресах устройств сети (рисунок 4.5).

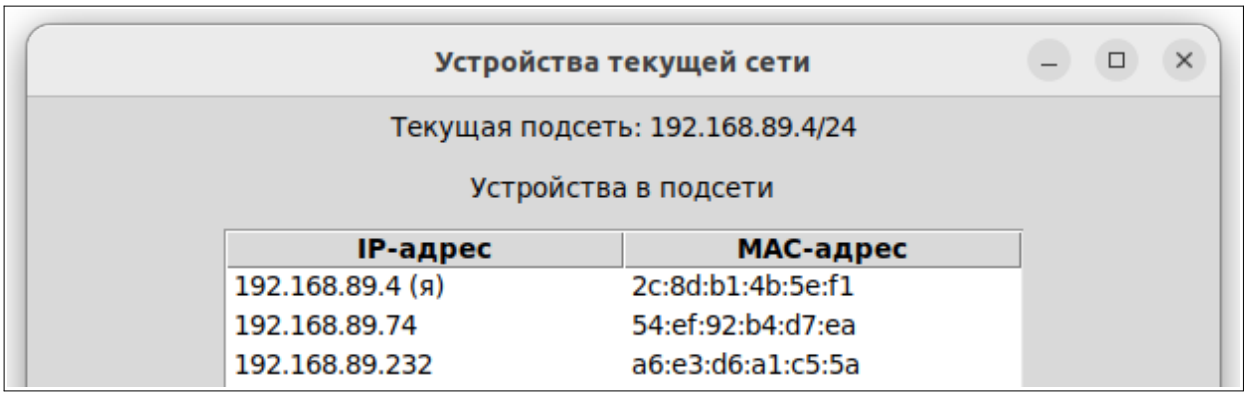

Рисунок 4.5 – Устройства сети

Для обнаружения устройств сети используется ARP сканирование. По диапазону IP-адресов рассылаются широковещательные ARP пакеты вида «компьютер с IP-адресом XXX, сообщите свой MAC-адрес компьютеру с MAC-адресом запросившего», и если ответ получен, то устройство считается обнаруженным.

4.4 Информация о портах

Просканировать порты можно как по адресу своей сети, так и по любому введенному. В таблицу выводятся данные об открытых портах и службах, за которыми они закреплены (рисунок 4.6).

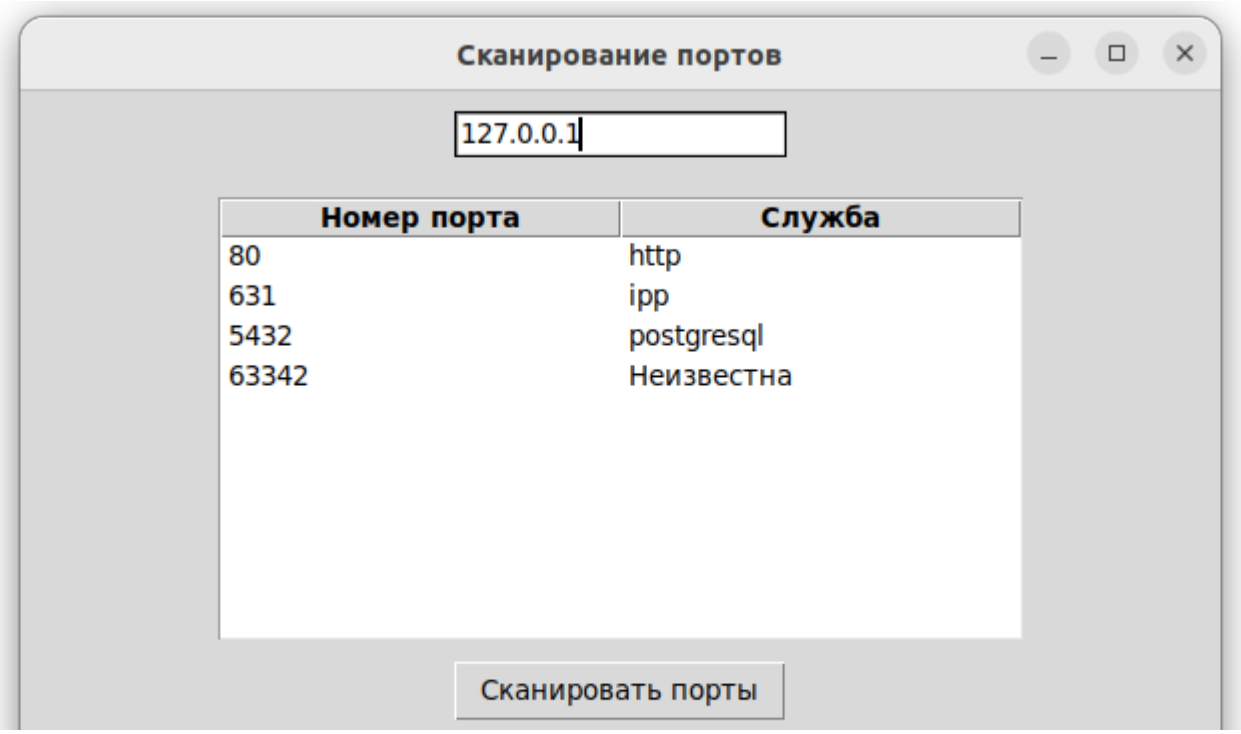

Рисунок 4.6 – Сканирование портов

Сканнер реализован на сокетах. Если подключение прошло успешно, значит порт открыт.

4.5 SNMP сканирование

По запросу к сетевому устройству выводится базовая информация о нем: описание, имя, время работы (рисунок 4.7).

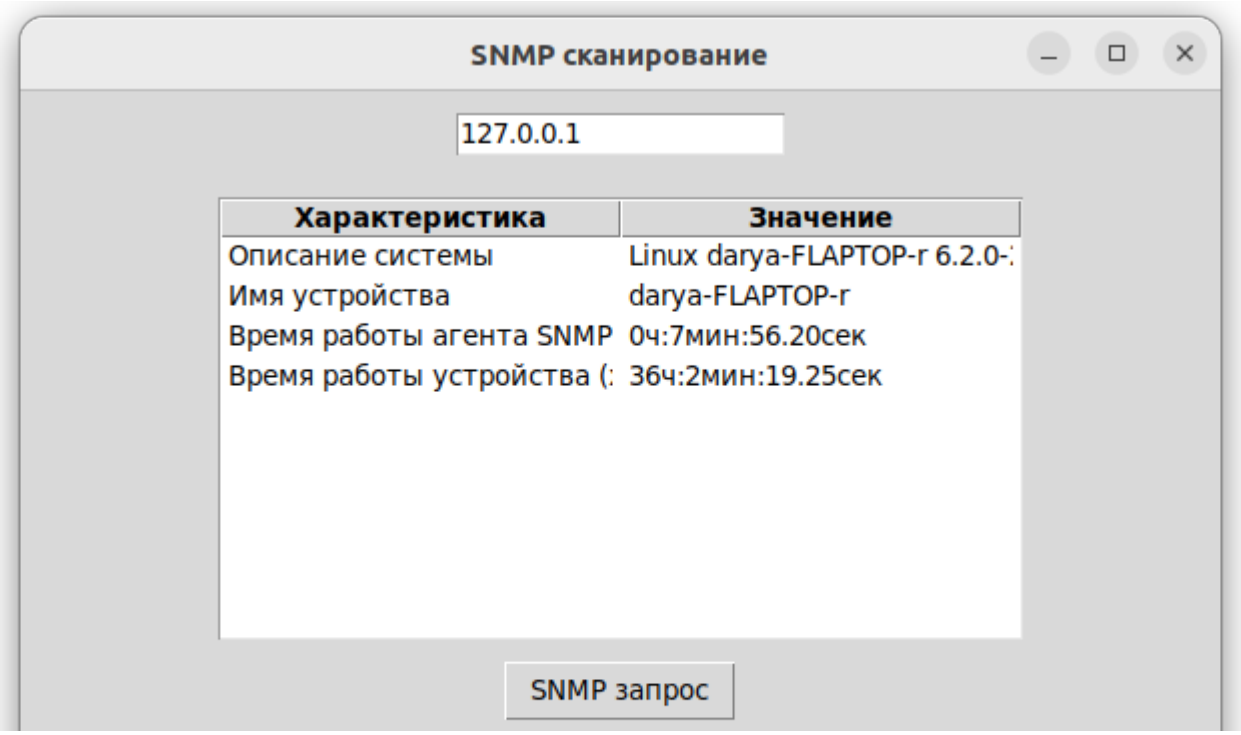

Рисунок 4.6 – SNMP запрос

SNMP сканирование производится на таких сетевых устройствах, как роутеры и коммутаторы. Чтобы сымитировать такое сетевое устройство, на компьютере-хосте запускается SNMP-агент, или так называемый SNMP-демон.

4.6 Анализ и ограничение трафика

Для рабыты с трафиком пользователю предоставлено несколько опций: анализ пакетов, тест скорости сети, а также ограничение пропускной способности сети (рисунок 4.7).

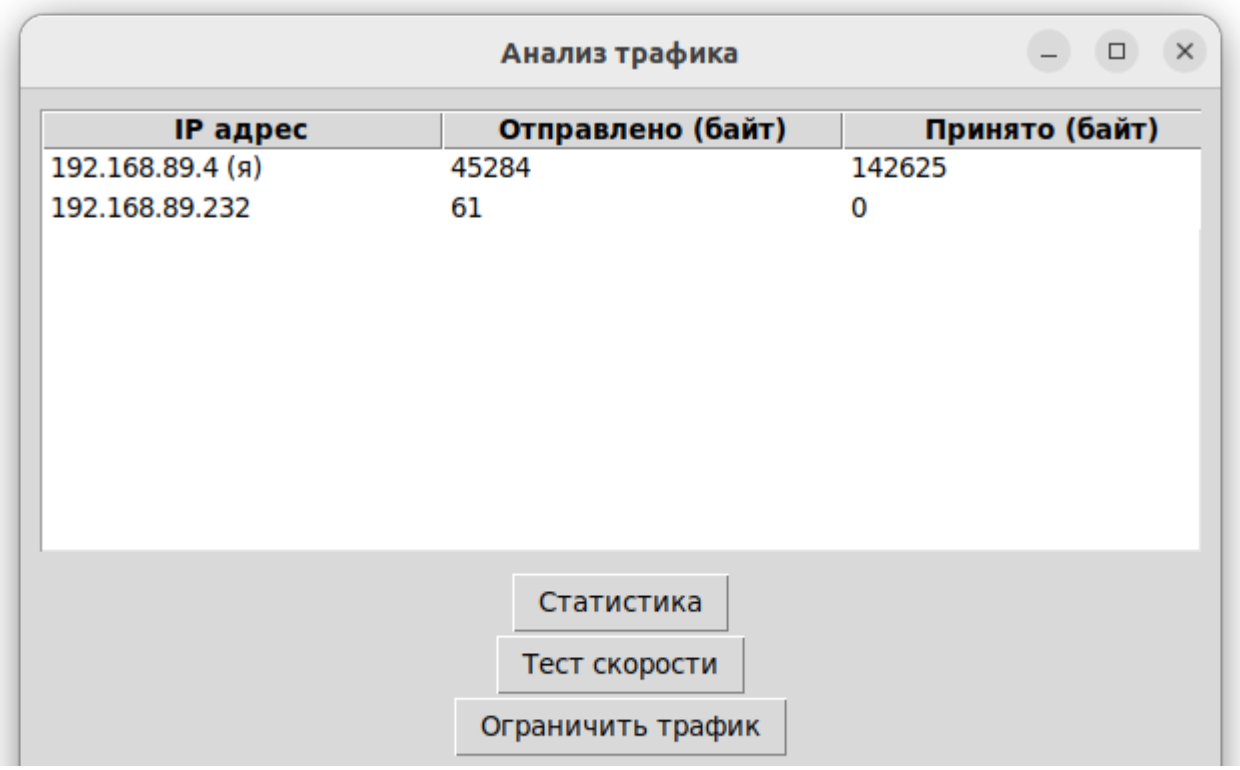

Рисунок 4.7 – Статистика трафика

При тесте скорости выводятся показатели скорости для получения и отправки пакетов соответственно (рисунок 4.8).

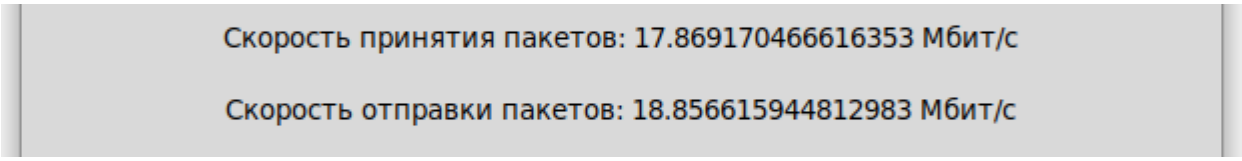

Рисунок 4.8 – Скорость передачи пакетов

Предоставляется также возможность эту скорость ограничить. Необходимо задать порог пропускной способности в Кбит/с, а также выбрать IP-адрес, на который будут накладываться ограничения (рисунок 4.9).

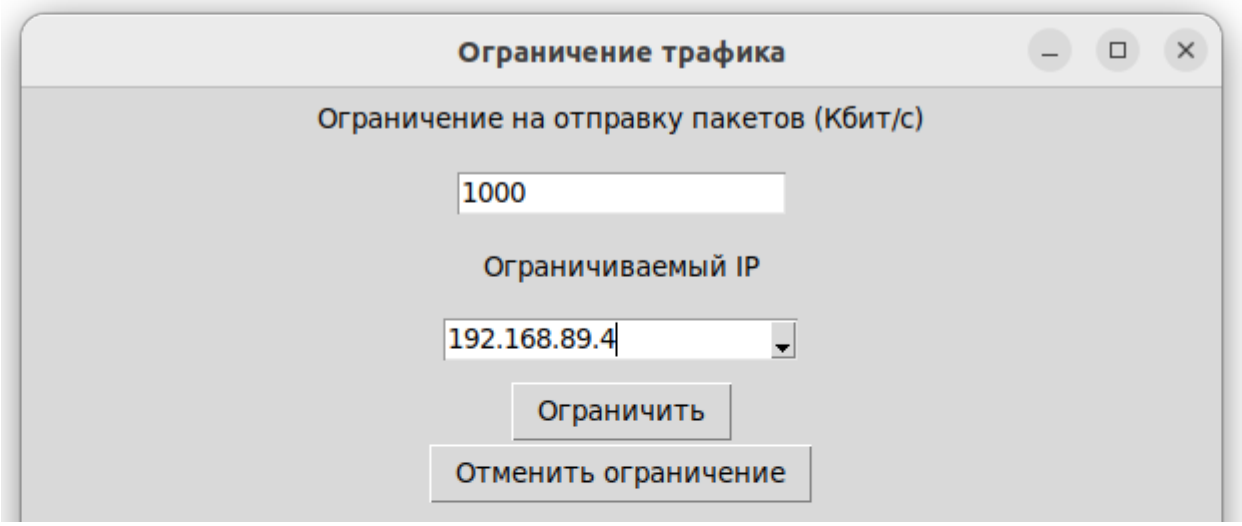

Рисунок 4.9 – Ограничение скорости

Можно проверить, что после наложения данных ограничений скорость действительно уменьшена (рисунок 4.10).

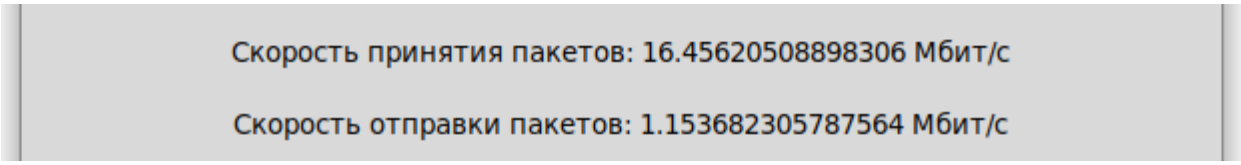

Рисунок 4.10 – Замедление скорости из-за ограничения

После отмены ограничения скорость вновь возвращается к привычным цифрам.

Таким образом, программа позволяет провести обзор сети, ее характеристик, устройств и их параметров, а также осуществлять управление трафиком ее узлов.

ЗАКЛЮЧЕНИЕ

В рамках курсового проекта было создано приложение, позволяющее проводить обзор сети. Оно предоставляет пользователю возможность получить подробную информацию о сетевых устройствах, включая IP-адреса, MAC-адреса и географические координаты.

Кроме того, приложение осуществляет сканирование портов на устройствах в сети, позволяя определить открытые порты и соответствующие им службы. С использованием протокола SNMP оно выполняет сканирование устройств для получения различной информации о системе, интерфейсах и процессах.

Одной из важных возможностей приложения является анализ трафика в сети. Пользователь может проанализировать объем переданных и принятых данных для каждого устройства, что позволяет выявить наиболее активные узлы в сети.

Помимо этого, приложение позволяет провести тест скорости сети, оценив скорость загрузки и выгрузки данных. И, наконец, пользователь имеет возможность ограничивать скорость передачи данных для определенных устройств в сети

Таким образом, результатом выполнения данного курсового проекта стала программа, которая позволяет провести обзор сети, ее характеристик, устройств и их параметров, а также осуществлять управление трафиком ее узлов. Данный проект поспособствовал более глубокому пониманию работы сети, а также значения ее характеристик.

СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1] RFC 791. Спецификация IP. [Электронный ресурс]. – Режим доступа : https://datatracker.ietf.org/doc/html/rfc791. – Дата доступа : 20.03.2024.

[2] RFC 793. Спецификация TCP. [Электронный ресурс]. – Режим доступа : https://datatracker.ietf.org/doc/html/rfc793. – Дата доступа : 20.03.2024.

[3] Python documentation [Электронный ресурс]. – Режим доступа : https://www.python.org/doc/. – Дата доступа : 20.03.2024.

[4] Обзор PyCharm [Электронный ресурс]. – Режим доступа : https://www.jetbrains.com/ru-ru/pycharm/features/. – Дата доступа : 20.03.2024.

[5] Обнаружение сетевых устройств [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/350394/ – Дата доступа: 10.04.2024.

[6] Сканим на Python. Как написать и улучшить собственный сканер портов [Электронный ресурс]. – Режим доступа: https://xakep.ru/2022/09/27/python-scanner/ – Дата доступа: 10.04.2024.

[7] Протокол ARP и «с чем его едят» [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/80364/ – Дата доступа: 15.04.2024.

[8] Протокол управления SNMP [Электронный ресурс]. – Режим доступа: https://selectel.ru/blog/snmp/ – Дата доступа: 04.10.2024.

[9] Список SNMP OID для мониторинга значений переменных [Электронный ресурс]. – Режим доступа: https://www.10-strike.ru/network-monitor/pro/spisok-poleznih-oid.shtml – Дата доступа: 15.04.2024.

[10] Настройка демона SNMP [Электронный ресурс]. – Режим доступа: https://www.ibm.com/docs/ru/aix/7.1?topic=snmpv1-snmp-daemon-configuration – Дата доступа: 20.04.2024.

[11] Мониторинг сетей при помощи Python [Электронный ресурс]. – Режим доступа:http://onreader.mdl.ru/MasteringPythonNetworking/content/Ch07.html – Дата доступа: 24.04.2024.

[12] Получение базовых параметров сетевых интерфейсов по-умолчанию для использования в проектах Python [Электронный ресурс]. – Режим доступа: https://codeby.net/threads/poluchenie-bazovyx-parametrov-setevyx-interfejsov-po-umolchaniju-dlja-ispolzovanija-v-proektax-python.80824/ – Дата доступа: 30.04.2024.

[13]Throttling Linux Network Bandwidth by IP Address and Time of Day [Электронный ресурс]. – Режим доступа: https://herrbischoff.com/2016/06/throttling-linux-network-bandwidth-by-ip-address-and-time-of-day/ – Дата доступа: 30.04.2024.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг программного кода

Файл check_params.py.

```
from ipaddress import IPv4Address
from socket import gethostbyname
from platform import system
from subprocess import check_output



def network_param() -> dict:
    if system() == "Linux":
        try:
                        com_run  =  check_output('ip  -h  -br  a  |  grep  UP',
shell=True).decode()
            default_interface = com_run.split()[0].strip()
            ipv4 = com_run.split()[2].strip().split("/")[0]
            ipv6 = com_run.split()[3].strip().split("/")[0]
        except Exception:
            com_run, default_interface, ipv4, ipv6 = None, None, None, None
        try:
                       router_ip  =  str(check_output('route  -n  |  grep  UG',
shell=True).decode().split()[1])
            try:
                IPv4Address(router_ip)
            except Exception:
                pass
            else:
                try:
                    ip = gethostbyname(router_ip)
                    router_ip = ip
                except Exception:
                    pass
        except Exception:
            router_ip = None
        try:
             mac_address = check_output(f'ifconfig {default_interface} | grep
"ether"', shell=True).decode(). \
                split()[1]
        except Exception:
            mac_address = None


         return {"Default Interface": default_interface, "IPv4 address": ipv4,
"IPv6 address": ipv6,
                "Router ip-address": router_ip, "MAC-address": mac_address}
    elif system() == "Windows":
        try:
            net_set = check_output(
```

```
                    "wmic nicconfig get IPAddress, MACAddress, IPEnabled,
SettingID, DefaultIPGateway /value"). \
                decode("cp866"). strip().split("\r\r\n")

               default_interface, ipv4, ipv6, router_ip, mac_address = None,
None, None, None, None
            text = ""
            for net in net_set:
                if net.strip() == "":
                    text = f"{text}|"
                else:
                    text = f"{text}~{net.strip()}"
            text = text.strip().split("||")
            for tx in text:
                if tx.split("~")[-3].split("=")[1] != "TRUE":
                    continue
                for item in tx.split("~"):
                    if item.strip() == "":
                        continue
                    if item.strip().split("=")[0] == "IPEnabled":
                        continue
                    if item.strip().split("=")[1] != "":
                        if item.strip().split("=")[0] == "SettingID":
                            default_interface = item.strip().split("=")[1]
                        if item.strip().split("=")[0] == "DefaultIPGateway":
                                        router_ip = item.strip().split("=")
[1].replace("{", "").replace("}", "").replace('"', '')
                        if item.strip().split("=")[0] == "MACAddress":
                            mac_address = item.strip().split("=")[1]
                        if item.strip().split("=")[0] == "IPAddress":
                                ipv4 = item.strip().split("=")[1].split(",")
[0].replace('"', '').replace("{", "")
                                ipv6 = item.strip().split("=")[1].split(",")
[1].replace('"', '').replace("}", "")
        except Exception:
                default_interface, ipv4, ipv6, router_ip, mac_address = None,
None, None, None, None
         return {"Default Interface": default_interface, "IPv4 address": ipv4,
"IPv6 address": ipv6,
                 "Router ip-address": router_ip, "MAC-address": mac_address}



if __name__ == "__main__":
    res = network_param()
    print(res)
```

Файл get_network.py.

```
from scapy.all import ARP, Ether, srp
import subprocess
import re
```

```
def get_network_info():
    try:
        output = subprocess.check_output(["ip", "addr", "show"]).decode()
        matches = re.findall(r"inet (\d+\.\d+\.\d+\.\d+)/(\d+)", output)
        for ip, mask in matches:
            if ip != "127.0.0.1":  # Пропускаем loopback интерфейсы
                return ip, mask
    except subprocess.CalledProcessError:
        print("Ошибка при выполнении команды ip addr show")
    return None, None



def format_network(ip, mask):
    # Форматируем IP и маску подсети в строку подсети вида "192.168.1.0/24"
    return f"{ip}/{mask}"



def get_network():
    return format_network(*get_network_info())



def scan_network(ip):
    arp = ARP(pdst=ip)
    ether = Ether(dst="ff:ff:ff:ff:ff:ff")
    packet = ether / arp
    result = srp(packet, timeout=3, verbose=False)[0]

    devices = []
    for sent, received in result:
        devices.append({'ip': received.psrc, 'mac': received.hwsrc})
    return devices



if __name__ == "__main__":
    ip, mask = get_network_info()
    if ip and mask:
        network = format_network(ip, mask)
        print("Полученная подсеть:", network)
    else:
        print("Не удалось получить информацию о сетевом интерфейсе")
    devices = scan_network(network)
    print("Список обнаруженных устройств:")
    print("IP\t\t\tMAC Address")
    print("-----------------------------------------")
    for device in devices:
        print(f"{device['ip']}\t\t{device['mac']}")
```

Файл public_ip.py.

```
from requests import get
from geopy.geocoders import Nominatim



def public_ip() -> str:
    try:
        return get('https://api.ipify.org/').text
    except Exception:
        return '127.0.0.1'



def geo_ip(ip) -> (list, bool):
    try:
        req = get(url=f'http://ip-api.com/json/{ip}').json()
        return [req['lat'], req['lon']]
    except Exception:
        return



def get_city(coords: list) -> (list):
    try:
        return Nominatim(user_agent="GetLoc").reverse(coords).address
    except Exception:
        return "Не удалось получить местоположение!"



if __name__ == "__main__":
    ip = public_ip()
    print(ip)
    coordinates = geo_ip(ip)
    print(coordinates)
    print(get_city(coordinates))
```

Файл scan_ports.py.

```
import socket


def scan_ports(ip, start_port=0, end_port=2 ** 16 - 1):
    open_ports = []


    for port in range(start_port, end_port + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)  # Установка таймаута для соединения
```

```
            result = sock.connect_ex((ip, port))
            if result == 0:
                service = get_service(port)
                open_ports.append((port, service))
            sock.close()

    return open_ports

def get_service(port):
    try:
        service = socket.getservbyport(port)
        return service
    except OSError:
        return "Неизвестна"

def main():
    ip = "192.168.89.4"#"127.0.0.1"  # "192.168.89.4"
    start_port = 0
    end_port = 2 ** 16 - 1

    print(f"Сканирование портов с {start_port} по {end_port} на {ip}...")

    open_ports = scan_ports(ip, start_port, end_port)

    if len(open_ports) == 0:
        print("Открытые порты не обнаружены")
    else:
        print("Список открытых портов:", open_ports)

if __name__ == "__main__":
    main()
```

Файл snmp.py.

```
from pysnmp.hlapi import *

def snmp_get(ip, oid):
    errorIndication, errorStatus, errorIndex, varBinds = next(
        getCmd(SnmpEngine(),
               CommunityData('dasha', mpModel=0),
               UdpTransportTarget((ip, 161)),
               ContextData(),
               ObjectType(ObjectIdentity(oid)))
    )

    if errorIndication:
        print(f"Ошибка: {errorIndication}")
        return None
    elif errorStatus:
```

```
        print(f"Ошибка: {errorStatus}")
        return None
    else:
        for varBind in varBinds:
            return varBind[1].prettyPrint()



def pretty_time(time: str):
    dot = int(time) % 100
    time = int(time) // 100
    hours = time // 60 // 60
    minutes = (time - hours * 3600) // 60
    seconds = time - hours * 3600 - minutes * 60
    formatted_time = f"{hours}ч:{minutes}мин:{seconds}.{dot}сек"
    return formatted_time



def get_sys_info(ip):
      sys_descr_oid = ".1.3.6.1.2.1.1.1.0"   # OID для описания системы -
описание устройства
    sys_name_oid = ".1.3.6.1.2.1.1.5.0"  # имя устройства
    sys_timeup_oid = ".1.3.6.1.2.1.25.1.1.0"  # время работы устройства
    daemon_timeup_oid = ".1.3.6.1.2.1.1.3.0"  # время работы snmp демона



    sys_descr = snmp_get(ip, sys_descr_oid)
    sys_name = snmp_get(ip, sys_name_oid)
    sys_timeup = snmp_get(ip, sys_timeup_oid)
    daemon_timeup = snmp_get(ip, daemon_timeup_oid)

    if sys_descr and sys_name and daemon_timeup:
        sys_descr = ("Описание системы", str(sys_descr))
        sys_name = ("Имя устройства", str(sys_name))
                        daemon_timeup   =   ("Время   работы   агента   SNMP",
pretty_time(daemon_timeup))
                    sys_timeup   =   ("Время   работы   устройства   (хоста)",
pretty_time(sys_timeup))

        return sys_descr, sys_name, daemon_timeup, sys_timeup
    else:
        return [("Не удалось получить",  "информацию о системе")]



def main():
    ip = "127.0.0.1"

    print("Получение информации о системе...")

    res = get_sys_info(ip)
```

```
    if len(res) > 1:
        for item in res:
            print(f"{item[0]}: {item[1]}")
    else:
        print("Не удалось получить информацию о системе")



if __name__ == "__main__":
    main()
```

Файл speed_test.py.

```
from speedtest.speedtest import Speedtest


def test_network_speed():
    st = Speedtest()
    st.get_best_server()

    download_speed = st.download() / 1024 / 1024  # в мегабитах в секунду
    upload_speed = st.upload() / 1024 / 1024  # в мегабитах в секунду

    return download_speed, upload_speed


def main():
    download_speed, upload_speed = test_network_speed()
    print(f"Скорость загрузки: {download_speed:.2f} Мбит/с")
    print(f"Скорость выгрузки: {upload_speed:.2f} Мбит/с")

if __name__ == "__main__":
    main()
```

Файл traffic.py.

```
from scapy.all import *
from get_network import get_network_info, format_network
from check_params import network_param
import ipaddress


def capture_traffic(interface, subnet, max_packets=1000):
    # Преобразуем строку подсети в объект ipaddress.IPv4Network
    network = ipaddress.IPv4Network(subnet, strict=False)
    # Словарь для хранения статистики трафика по IP-адресам
    traffic_stats = defaultdict(lambda: {"sent_bytes": 0, "recv_bytes": 0})

    # Функция обработки захваченных пакетов
    def packet_callback(packet):
```

```
        if IP in packet:
            src_ip = packet[IP].src
            dst_ip = packet[IP].dst
                            if ipaddress.IPv4Address(src_ip) in network or
ipaddress.IPv4Address(dst_ip) in network:
                if TCP in packet:
                                # Если есть TCP заголовок, учитываем количество
переданных байт
                    sent_bytes = len(packet[TCP].payload)
                    if ipaddress.IPv4Address(src_ip) in network:
                        traffic_stats[src_ip]["sent_bytes"] += sent_bytes
                    if ipaddress.IPv4Address(dst_ip) in network:
                        traffic_stats[dst_ip]["recv_bytes"] += sent_bytes
                elif UDP in packet:
                                # Если есть UDP заголовок, учитываем количество
переданных байт
                    sent_bytes = len(packet[UDP].payload)
                    if ipaddress.IPv4Address(src_ip) in network:
                        traffic_stats[src_ip]["sent_bytes"] += sent_bytes
                    if ipaddress.IPv4Address(dst_ip) in network:
                        traffic_stats[dst_ip]["recv_bytes"] += sent_bytes


    # Захватываем пакеты на указанном интерфейсе
    sniff(iface=interface, prn=packet_callback, count=max_packets)


    return traffic_stats


def main():
    params = network_param()
    interface = params["Default Interface"]
    my_ip = params["IPv4 address"]
    subnet = format_network(*get_network_info())
    traffic_stats = capture_traffic(interface, subnet)


    me = "(данное устройство)"
    print("Статистика трафика:")
    for ip, stats in traffic_stats.items():
              print(f"IP: {ip} {me if ip == my_ip else ''}, Отправлено:
{stats['sent_bytes']} байт, Принято: {stats['recv_bytes']} байт")


if __name__ == "__main__":
    main()
```

Файл traffic_limit.py.

```
import os
from check_params import network_param


def limit_traffic(ip_address, max_rate):
    params = network_param()
```

```
    interface = params["Default Interface"]
    # Определение правила для ограничения входящего трафика
    os.system(f"tc qdisc add dev {interface} root handle 1: htb default 11")
     os.system(f"tc class add dev {interface} parent 1: classid 1:1 htb rate
{max_rate}kbit")
     os.system(f"tc filter add dev {interface} protocol ip parent 1: prio 1
u32 match ip src {ip_address} flowid 1:1")


def remove_traffic_limit(ip_address):
    params = network_param()
    interface = params["Default Interface"]
    # Удаление правил для ограничения входящего трафика
     os.system(f"tc filter del dev {interface} protocol ip parent 1: prio 1
u32 match ip src {ip_address} flowid 1:1")



def main():
     ip_address = "192.168.89.4"#"192.168.89.232"   # IP-адрес, для которого
будем ограничивать трафик
    max_rate = 1000  # Максимальная скорость в килобитах в секунду
    #
    # limit_traffic(ip_address, max_rate)
       # print(f"Трафик для IP-адреса {ip_address} ограничен до {max_rate}
kbps")
    #
    # remove_traffic_limit(ip_address)
    # print(f"Ограничение трафика для IP-адреса {ip_address} снято")


if __name__ == "__main__":
    main()
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Ведомость курсового проекта