МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Исследование алгоритмов оптимизации компьютерных сетей»

Обучающийся А.А. Землянин
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., Т.Г. Султанов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент, О.Н. Брега
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Тема выпускной работы: Исследование алгоритмов оптимизации компьютерных сетей.

Данная работа посвящена исследованию различных алгоритмов, которые используются для оптимизации компьютерных сетей. Она охватывает различные аспекты сетевой оптимизации, такие как управление трафиком, оптимизация маршрутизации, балансировка нагрузки и т.д.

Объектом исследования являются компьютерные сети, которые используются в различных организациях и предприятиях.

Предметом исследования являются алгоритмы, которые могут быть использованы для оптимизации компьютерных сетей.

Целью данной работы является исследование и анализ различных алгоритмов оптимизации компьютерных сетей, а также определение наиболее эффективных алгоритмов для использования в конкретных сетевых сценариях.

В результате работы будет разработано программное обеспечение для оптимизации компьютерных сетей, использующее наиболее эффективные алгоритмы.

Abstract

The topic of the graduate work: The study of algorithms for optimizing computer networks.

What the paper is about: This paper is devoted to the study of various algorithms that are used to optimize computer networks. It covers various aspects of network optimization, such as traffic management, routing optimization, load balancing, etc.

Object of study: The object of the study is computer networks that are used in various organizations and enterprises.

Subject of study: The subject of the study are algorithms that can be used to optimize computer networks.

Purpose of the study: The purpose of this work is to study and analyze different algorithms to optimize computer networks, as well as determining the most effective algorithms for use in specific network scenarios. The work will result in the development of computer network optimization software that uses the most effective algorithms.

The introduction defines the relevance of the topic, the goal and objectives set in the work, as well as the object and subject of the study.

Оглавление

Введение

Информационные технологии являются неотъемлемой частью нашей жизни, и компьютерные сети играют ключевую роль в обеспечении связи между различными устройствами и ресурсами. Сегодня мы сталкиваемся с огромным количеством данных, которые передаются через сети, и в этом контексте оптимизация компьютерных сетей является необходимой задачей.

Основная задача алгоритмов оптимизации компьютерных сетей заключается в улучшении производительности сетей путем оптимального распределения ресурсов и управления трафиком. Это включает в себя решение таких проблем, как минимизация задержек передачи данных, увеличение пропускной способности, балансировка нагрузки и обеспечение безопасности передачи информации.

В последние годы было предложено и исследовано множество алгоритмов оптимизации компьютерных сетей.

Некоторые из них основаны на классических математических методах, таких как линейное программирование и теория графов, в то время как другие используют эволюционные алгоритмы, искусственные нейронные сети или методы машинного обучения.

В рамках бакалаврской работы по теме "Исследование алгоритмов оптимизации компьютерных сетей" мы проведем исследование различных алгоритмов, которые могут быть использованы для повышения производительности и надежности сети. Мы также рассмотрим основные характеристики компьютерных сетей, которые влияют на ее производительность, и применим наши исследования для поиска оптимального решения.

Объектом исследования являются компьютерные сети, которые используются в различных организациях и предприятиях.

Предметом исследования являются алгоритмы, которые могут быть

использованы для оптимизации компьютерных сетей.

Целью данного исследования является анализ различных алгоритмов оптимизации компьютерных сетей, разработка программного обеспечения и проведение тестирования для оценки их производительности и эффективности. Для достижения поставленной цели необходимо выполнить следующие задачи исследования:

- анализ существующих алгоритмов оптимизации компьютерных сетей;
- разработка программного обеспечения для оптимизации компьютерных сетей;
- тестирование программного обеспечения.

В результате выполнения данного исследования ожидается получение практических результатов, которые будут полезны для разработчиков сетевых систем и специалистов по оптимизации компьютерных сетей.

Основные выводы и рекомендации будут представлены в заключительной главе работы.

Имеет большое значение в современном мире, и наша программа позволит провести данное исследование более эффективно и точно.

Выпускная квалификационная работа содержит пояснительную записку объемом 53 страниц, 22 рисунка, 1 таблицу и список используемой литературы, состоящий из 25 источников.

Глава 1 Анализ алгоритмов оптимизации компьютерных сетей

1.1 Определение проблемы оптимизации в компьютерных сетях

Компьютерные сети являются важной составляющей современной информационной инфраструктуры и широко применяются в различных сферах, таких как бизнес, образование, здравоохранение и многих других.

Оптимизация компьютерных сетей имеет существенное значение для обеспечения их эффективного функционирования. Проблема оптимизации в компьютерных сетях возникает из-за различных факторов, таких как увеличение объема передаваемых данных, разнообразие типов трафика, динамически изменяющиеся потребности пользователей и требования к качеству обслуживания (Quality of Service - QoS) [16]. Оптимизация сетей направлена на достижение оптимального использования ресурсов сети, улучшение производительности, снижение задержек и потерь данных, повышение надежности и обеспечение высокого уровня QoS для пользователей.

Оптимизация компьютерных сетей основана на ряде принципов, которые направлены на повышение производительности, снижение задержек, обеспечение надежности и энергоэффективности, а также управление качеством обслуживания [25].

Максимизация пропускной способности является важным принципом оптимизации. Это достигается оптимальным распределением ресурсов сети и использованием эффективных алгоритмов маршрутизации [6].

Минимизация задержек передачи данных играет ключевую роль в оптимизации сетей. Это достигается выбором оптимальных маршрутов, оптимизацией времени обработки пакетов и применением различных техник снижения задержек.

Обеспечение надежности сети является неотъемлемым принципом оптимизации. Сеть должна быть способна обнаруживать и исправлять ошибки, а также восстанавливаться после сбоев. Это достигается через применение дублирования данных, резервирования каналов связи и автоматического восстановления.

Энергоэффективность является важным аспектом оптимизации сетей. Снижение энергопотребления позволяет уменьшить экологическую нагрузку и снизить затраты на энергию. Это достигается через управление энергопотреблением устройств и применение энергосберегающих алгоритмов.

1.2 Классификация алгоритмов оптимизации компьютерных сетей

В компьютерных сетях протокол – это набор правил и соглашений, определяющих формат и порядок обмена данными между устройствами в сети. Протоколы определяют структуру сообщений, используемых для передачи данных, а также определяют, как устройства взаимодействуют друг с другом и какие действия выполняются при передаче и приеме сообщений. Протоколы в компьютерных сетях определяют различные аспекты коммуникации, включая установление и разрыв соединений, адресацию, маршрутизацию, обнаружение и исправление ошибок, контроль потока данных и управление перегрузками. Каждый протокол выполняет определенные функции, обеспечивая эффективную и надежную передачу данных в сети.

Наиболее известным и широко используемым протоколом в компьютерных сетях является TCP/IP (Transmission Control Protocol/Internet Protocol). TCP/IP является основным протоколом Интернета и обеспечивает передачу данных между устройствами в глобальной сети [8]. Он определяет структуру IP-адресов, формат пакетов данных, механизмы маршрутизации и

другие аспекты сетевой коммуникации.

Для того, чтобы лучше понять, что такое протокол, на рисунке 1 приведена аналогия с человеком.

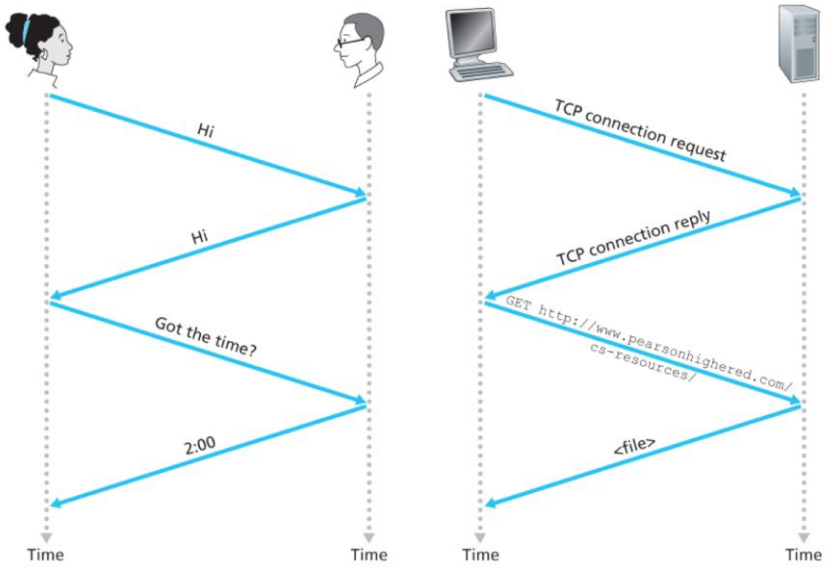

Рисунок 1 - Аналогия протокола

Кроме TCP/IP, существует множество других протоколов, используемых в компьютерных сетях, таких как HTTP (Hypertext Transfer Protocol) для передачи веб-страниц [24], FTP (File Transfer Protocol) для передачи файлов, SMTP (Simple Mail Transfer Protocol) для отправки электронной почты и многие другие. Каждый протокол имеет свою спецификацию и используется для определенных типов коммуникации в сети.

В компьютерных сетях протоколы необходимы для взаимодействия устройств и связи. Они обеспечивают успешный обмен данными, позволяя различным устройствам и программам "говорить на одном языке".

Алгоритмы оптимизации компьютерных сетей могут быть классифицированы по целому ряду факторов, включая стратегии оптимизации, уровни абстракции, методы решения проблем и области

применения.

Алгоритмы маршрутизации. Выбор путей для передачи данных в сети оптимизируется с помощью этих методов. Они могут быть основаны на различных идеях, включая наименьшую стоимость, наибольшую пропускную способность, наименьшую задержку или сочетание этих идей. Алгоритм Дейкстры является иллюстрацией такого алгоритма.

Алгоритмы планирования ресурсов. Эти методы позволяют максимизировать производительность и удовлетворить потребности различных приложений путем оптимизации распределения сетевых ресурсов, включая пропускную способность, процессорное время и память. Алгоритмы справедливой очереди, алгоритмы планирования временных интервалов и алгоритмы динамического распределения ресурсов - вот несколько примеров таких алгоритмов.

Методы оптимизации трафика. Эти алгоритмы оптимизируют потоки сетевого трафика для повышения эффективности и сокращения задержек. Они могут включать методы приоритизации трафика, управления буфером, управления потоками и сжатия данных. Алгоритмы контроля перегрузок и алгоритмы адаптивного управления буфером - два примера таких алгоритмов.

Классификация алгоритмов оптимизации сети позволяет систематизировать различные стратегии и методы, помогая исследователям и разработчикам выбрать наиболее эффективные алгоритмы для решения определенных проблем оптимизации сети.

1.3 Необходимость оптимизации компьютерных сетей

Компьютерные сети играют крайне важную роль в современном обществе. Они позволяют устройствам общаться друг с другом, обмениваться информацией и выполнять задачи. Однако компьютерные сети могут столкнуться с проблемами, влияющими на их производительность и безопасность. Поэтому компьютерные сети необходимо оптимизировать, чтобы они могли работать лучше.

Одной из причин оптимизации компьютерных сетей является постоянно увеличивающееся количество пользователей и подключенных устройств. Это может привести к снижению производительности, так как нагрузка на сеть увеличивается с увеличением количества пользователей. Кроме того, новые гаджеты и технологии потребляют больше сетевых ресурсов и пропускной способности, что может снизить производительность сети.

Угрозы безопасности являются еще одним мотивом для оптимизации компьютерных сетей. Существует множество угроз, которые могут поставить под угрозу вашу сеть или украсть очень важные данные. Антивирусное программное обеспечение, брандмауэры и другие методы оптимизации безопасности могут остановить только некоторые из этих угроз. Однако эффективная защита требует постоянных проверок безопасности и обновлений программного обеспечения.

Оптимизация компьютерной сети также снижает затраты на техническое обслуживание и значительно снижает риск сбоев и простоев. Некоторые методы оптимизации, в частности мониторинг сети и диагностика ошибок, могут помочь быстро выявить и устранить проблемы, что может сократить время простоя и повысить эффективность сети.

Наконец, оптимизация сети может повысить удовлетворенность клиентов и качество обслуживания. Пока сеть работает быстро и без помех,

пользователи могут выполнять задачи, обмениваться данными, общаться и получать доступ к сетевым ресурсам более эффективно, без задержек и проблем.

1.4 Задачи оптимизации компьютерных сетей

Масштабируемость. По мере масштабирования сети обслуживание и оптимизация сети становится все более сложной задачей. Для обработки возросшего сетевого трафика может потребоваться дополнительное оборудование, включая коммутаторы и маршрутизаторы, а также программное обеспечение. Это может привести к увеличению сложности и затруднить решение проблем. Проблемы с масштабируемостью также могут повлиять на производительность, поскольку перегрузка сети становится все более распространенным явлением.

Производительность. Перегрузка сети, потеря пакетов и задержка — это лишь некоторые из многих причин низкой производительности сети. Эти проблемы могут замедлить передачу данных и затруднить получение пользователями необходимых ресурсов. Низкая производительность может отрицательно сказаться на вовлеченности пользователей, что особенно неприятно для компаний, которые полагаются на сетевые ресурсы для обслуживания критически важных приложений. безопасность. Защита конфиденциальных данных, предотвращение кибератак и поддержание целостности сетевых ресурсов — все это зависит от кибербезопасности. Нарушения безопасности возникают, когда сеть не защищена должным образом, что подвергает ее таким угрозам, как вредоносное ПО, попытки фишинга и атаки типа «отказ в обслуживании». Реализация тщательной стратегии безопасности, включая брандмауэры, антивирусное программное обеспечение и системы обнаружения и предотвращения вторжений, имеет

решающее значение.

Совместимость. Могут существовать проблемы совместимости между различным сетевым оборудованием и программным обеспечением, что может привести к проблемам интеграции и совместимости. Это может быть чрезвычайно сложной задачей при попытке соединить исторические системы с современными технологиями. Проблемы совместимости могут привести к простоям, потере данных и снижению производительности, что может дорого обойтись бизнесу.

Надежность. Потеря данных, потеря производительности и перебои в работе сети могут нанести огромный ущерб бизнесу. Проблемы с надежностью могут возникнуть, если возникают программные ошибки, сбои сетевых компонентов или проблемы с базовой инфраструктурой. Чтобы снизить риск простоя и потери данных, необходимо иметь резервные системы и резервные копии.

Сложность. Современные сети имеют несколько уровней оборудования, программного обеспечения и протоколов и могут быть чрезвычайно сложными. Из-за этой сложности поиск и устранение проблем могут быть затруднены, что может привести к длительному простою и снижению производительности. Внедрение решения для мониторинга и контроля имеет решающее значение для оптимизации сетевой среды и облегчения устранения неполадок [4].

Стоимость. Для оптимизации сети могут потребоваться значительные инвестиции в оборудование, программное обеспечение и персонал. Предприятиям с ограниченными финансовыми возможностями может быть трудно внедрять передовые технологии или увеличивать число сотрудников. Очень важно сопоставить расходы на оптимизацию с теми преимуществами, которые она может принести, например, повышение производительности и безопасности сети.

Обслуживание. Для обслуживания сети может потребоваться много времени и ресурсов. Для поддержания эффективной работы сети, а также для выявления и устранения возможных проблем до того, как они станут серьезными, необходимо регулярное обслуживание. Обновление программного обеспечения, исправления, отслеживание производительности и модернизация оборудования – вот несколько примеров задач по обслуживанию.

Администрирование сети. Любая попытка оптимизации должна успешно управляться в сети. Мониторинг производительности сети, выявление и устранение проблем, настройка сетевого оборудования и поддержание оптимальной эффективности сети - все это является частью управления сетью. Для эффективного управления сетью необходима тщательная стратегия управления, компетентный персонал и соответствующие инструменты [2].

Гибкость. Адаптируемая сеть может меняться в соответствии с требованиями бизнеса и технологическим прогрессом. Отсутствие гибкости может помешать способности сети адаптироваться к меняющимся требованиям и затруднить оптимизацию. Использование открытых стандартов и адаптируемых архитектур, позволяющих легко интегрировать новые технологии, приведет к созданию гибкой сети.

Сложность трафика. Различные типы трафика требуют различных уровней приоритета и пропускной способности, а сетевой трафик может быть сложным и изменчивым. Может быть трудно оптимизировать сеть для всех видов трафика, особенно в обширных и сложных контекстах. Такие методы управления трафиком, как формирование трафика и процедуры обеспечения качества обслуживания (QoS), могут помочь убедиться в том, что ключевые приложения получают необходимые им ресурсы [16].

Интеграция облака. Поскольку все больше компаний переносят свои

данные и приложения в облако, оптимизация сети может столкнуться с трудностями. Требуется тщательное планирование и управление для интеграции облачных сервисов с локальной инфраструктурой, чтобы сеть могла обрабатывать возросший трафик и безопасно передавать данные.

Помехи. Перебои в работе сети и проблемы с производительностью могут возникнуть из-за помех. Физические препятствия, такие как стены и другие структуры, могут препятствовать передаче сигналов, также как и электромагнитные помехи от другого электрического оборудования. Особенно чувствительна к помехам беспроводная сеть, что может привести к снижению производительности и надежности [13].

Ограничения пропускной способности. При оптимизации сети может возникнуть серьезная проблема с ограничением пропускной способности. Когда к сети одновременно подключается множество пользователей, ограничения пропускной способности могут привести к снижению скорости передачи данных и ухудшению производительности сети. Эффективное использование полосы пропускания может быть обеспечено за счет внедрения методов управления ею и предоставления приоритета основным приложениям [3].

Соблюдение правил. Оптимизация сети может столкнуться со значительными препятствиями из-за соблюдения нормативных требований, особенно для предприятий, занятых в высокорегулируемых секторах, таких как здравоохранение и банковское дело. Положения, регулирующие безопасность и конфиденциальность данных, требования по хранению данных и другие юридические и нормативные требования являются примерами требований по соблюдению нормативных требований.

Удаленная работа. Оптимизация сети сталкивается с дополнительными трудностями в результате роста удаленной работы. Удаленным сотрудникам требуется безопасное и надежное подключение к сети, что ограничивает

доступные ресурсы и повышает опасность кибератак. Использование защищенных виртуальных частных сетей и внедрение решений для удаленного доступа может помочь гарантировать удаленным работникам доступ к необходимым ресурсам, обеспечивая при этом безопасность сети.

Обучение. Оптимизация сети требует квалифицированного персонала, прошедшего обучение передовому опыту и новейшим технологиям. Убедиться, что сотрудники имеют соответствующее образование и квалификацию, может быть непросто, особенно для малых и средних предприятий. Подготовить персонал к управлению и оптимизации сети можно, инвестируя в обучение и развитие.

В этой главе были рассмотрены основные вопросы и трудности, связанные с оптимизацией компьютерных сетей, а также потенциальные решения. Однако вопросы оптимизации сетей могут быть решены с использованием широкого спектра дополнительных методик и алгоритмов. Например, одной из стратегий является использование машинного обучения. Модели, которые прогнозируют спрос на сетевые ресурсы и на их основе принимают решения по оптимизации сети, могут быть разработаны с использованием методов машинного обучения [5].

Использование графовых алгоритмов является альтернативной стратегией. Используя графы для описания топологии сети и ее ресурсов, графовые алгоритмы способны решать проблемы маршрутизации, планирования ресурсов и балансировки нагрузки.

Также можно выделить использование анализа данных и статистических методов для выбора оптимальных параметров сети и прогнозирования ее производительности.

Каждая метод и подход имеет свои преимущества и недостатки, и выбор конкретного подхода зависит от особенностей и характеристик рассматриваемой сети, а также от ее целей. В следующей главе основное

внимание будет уделено конкретным иллюстрациям различных методов и алгоритмов оптимизации компьютерных сетей.

1.5 Сравнение существующих алгоритмов оптимизации компьютерных сетей

Поддержание и развитие современного информационного общества в значительной степени зависит от оптимизации компьютерных сетей. Для увеличения скорости, надежности и безопасности сети можно применять различные стратегии оптимизации компьютерных сетей. В данном исследовании будет проведено сравнение нескольких современных стратегий оптимизации компьютерных сетей.

Методы маршрутизации. Алгоритмы маршрутизации являются одним из наиболее широко используемых методов оптимизации компьютерных сетей. Эти методы позволяют улучшить канал передачи данных в сети, что может снизить задержки и увеличить скорость передачи данных. Протокол Open Shortest Path First (OSPF) является одним из наиболее широко используемых алгоритмов маршрутизации.

Высокая скорость, гибкость в настройке различных параметров маршрутизации и автоматическое обнаружение изменений топологии сети - все это преимущества алгоритма OSPF. Однако требование более тщательной настройки и поддержка более сложной сетевой архитектуры являются некоторыми недостатками этого подхода.процедуры управления трафиком.

Алгоритмы управления трафиком часто используются при оптимизации компьютерных сетей. Эти алгоритмы управляют потоком данных в сети, что может помочь снизить задержку и увеличить пропускную способность. Метод WRED (Weighted Random Early Detection) является одной из наиболее широко используемых систем управления трафиком [7].

Управление потоком данных и снижение задержки в сети — два преимущества алгоритма WRED. Однако он не обеспечивает полной защиты от перегрузки сети и может даже привести к увеличению потери пакетов.

Оптимизация алгоритма оптимизации пропускной способности заключается в достижении наилучшей эффективности. Совместимость алгоритмов оптимизации сети с текущим аппаратным и программным обеспечением также является важным фактором. Некоторые операционные системы и сетевое оборудование могут быть несовместимы с определенными алгоритмами, что может вызвать проблемы. Поэтому перед выбором алгоритма оптимизации необходимо тщательно рассмотреть его совместимость с текущей инфраструктурой.

Наконец, важно учитывать, сколько будет стоить разработка и поддержка алгоритма. В то время как некоторые алгоритмы могут быть доступны бесплатно и с открытым исходным кодом, для реализации и обслуживания других может потребоваться больше средств и ресурсов. Поэтому перед выбором метода оптимизации следует провести стоимостной и бюджетный анализ проекта.

В целом, уникальные требования и характеристики организации, а также ее размер, бюджет и инфраструктура определяют, какой метод оптимизации компьютерной сети является лучшим. При выборе алгоритма следует учитывать такие требования, как скорость, надежность, безопасность, совместимость, а также затраты на внедрение и поддержку. Для выбора наилучшего алгоритма оптимизации для конкретного случая также необходимо рассмотреть преимущества и недостатки каждого алгоритма и провести сравнительное исследование его характеристик.

1.6 Обзор инструментов и технологий для анализа проблем и оптимизации компьютерных сетей

Одной из ключевых частей информационной архитектуры современного предприятия является компьютерная сеть [1]. Безопасная и эффективная передача данных обеспечивается хорошо функционирующими сетями, которые могут повысить производительность и прибыльность организации. Однако проблемы, низкая производительность и даже сбои в сети могут иметь серьезные финансовые и репутационные последствия для бизнеса. Для оценки и оптимизации компьютерных сетей требуются специальные методы и приемы. В этой статье мы рассмотрим некоторые из наиболее широко используемых методов и советов по устранению неполадок и повышению производительности компьютерной сети.

Сетевой анализатор. Инструменты сетевого анализа можно использовать для изучения сетевого трафика и выявления проблем с производительностью. С их помощью можно найти неисправные устройства, неэффективное потребление сетевых ресурсов, источники утечек данных и другие проблемы.

Системы мониторинга. Система мониторинга может использоваться для наблюдения за состоянием компьютерных сетей и оборудования в режиме реального времени. Они способны обнаруживать и регистрировать сетевые проблемы и предоставлять данные для дальнейшего расследования и устранения неполадок.

Программное обеспечение для управления сетью. Программное обеспечение для управления сетью предоставляет возможность управления сетями, включая настройку устройств, мониторинг и управление безопасностью. Его можно использовать для управления сетевой инфраструктурой, предоставляя информацию о состоянии сети, выявляя

проблемы и устраняя неполадки.

Кроме того, доступны различные инструменты, такие как сканеры уязвимостей, программное обеспечение для резервного копирования и т. д., для анализа проблем и оптимизации производительности компьютерной сети.

Выводы по первой главе

В первой главе рассматриваются основы компьютерных сетей и методы анализа и оптимизации. В частности, обсуждаются различные типы сетей и методы передачи данных, используемые для связи между сетевыми устройствами. Затем обсуждаются основные проблемы, которые могут повлиять на компьютерные сети, включая потерю пакетов, задержку данных и проблемы безопасности. Предоставляет инструменты для мониторинга и анализа сетевого трафика для выявления проблем и повышения производительности сети.

Защита компьютерных сетей, которая также описана в этой главе, является ключевым компонентом. Описываются инструменты для обнаружения и предотвращения сетевых атак, а также методы аутентификации, авторизации и шифрования. Также обсуждаются основные идеи по оптимизации компьютерных сетей. Изучаются методы совершенствования протоколов передачи данных и средства мониторинга и оптимизации пропускной способности сети.

Затем обсуждаются основные компоненты компьютерных сетей и методы их анализа и оптимизации. Мы рассмотрим основные проблемы, которые могут возникнуть в сети, и доступные решения. В следующей главе диссертации обсуждается, как на самом деле использовать эти инструменты для решения реальных проблем.

Глава 2 Обзор и сравнительный анализ существующих алгоритмов оптимизации компьютерных сетей

2.1 Алгоритмы маршрутизации

В компьютерных сетях алгоритмы маршрутизации используются для выбора наилучшего пути передачи данных от отправителя к получателю. Они минимизируют задержки и потери пакетов, при этом эффективно управляя сетью для приема огромных объемов трафика.

Каждый из нескольких алгоритмов маршрутизации может быть оптимизирован для уникальных потребностей сети и ее пользователей.

Алгоритм Дейкстры является одним из наиболее широко используемых алгоритмов маршрутизации. Он работает путем определения кратчайшего маршрута между каждой вершиной в сетевом графе. Каждому узлу в этом методе присваивается стоимость, которая представляет собой сумму расстояний от узла-источника. Затем алгоритм находит вершину с наименьшей стоимостью и обновляет стоимости всех соседних вершин, учитывая стоимость пути через текущую вершину. Процесс повторяется до тех пор, пока не будет найден кратчайший путь до всех вершин в графе. На рисунок 2 схематично изображен алгоритм Дейкстры. Кружками обозначены вершины, линиями – пути между ними (ребра графа). В кружках обозначены номера вершин, над ребрами обозначен их вес – длина пути. Рядом с каждой вершиной обозначена метка – длина кратчайшего пути в эту вершину из вершины 1.

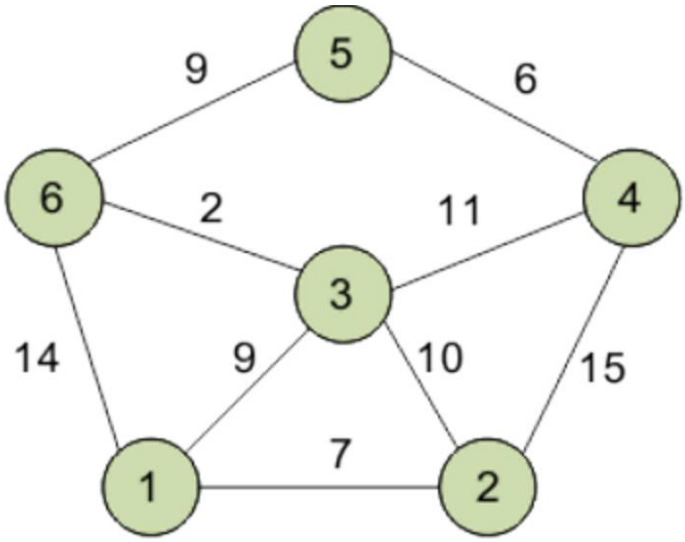

Рисунок 2 - Алгоритм Дейкстры

Сформулируем задачу.

Пусть задан неориентированный граф $G = (V, E)$ с числом вершин $n = V(G)$ и числом ребер $m = E(G)$. Путем длинной h от вершины p до вершины q будем называть упорядоченную последовательность связанных звеньев $(e_p, \ldots, e_q)_h = (p, i_1), \ldots, (i_j, i_{j+1}), \ldots, (i_h, q)$.

Каждому ребру (i, j) поставим в соответствие положительное число $\omega_{i,j}$, которое будем называть весом ребра (i, j). В случае, когда $(i, j) \notin E$, положим $\omega_{i,j} = \infty$. Величину (1)

$$I_{pq}(h) = \sum_{(j,j+1)\in(e_p,\ldots,e_q)} \omega_{i,j+1} \qquad (1)$$

будем называть взвешенной длинной пути от вершины p до q. Обозначим через Ψ_{pq} множество всех допустимых путей из p в q.

Путь $(e_p, \ldots, e_q)^*_{\underline{h}}$, который определяет число I^*_{pq} по формуле (2)

$$I^*_{pq} = \min_{\in(e_p,\ldots,e_q)\in\Psi_{pq}} \sum_{(j,j+1)\in(e_p,\ldots,e_q)} \omega_{i,j+1} \qquad (2)$$

будем называть наикратчайшим взвешенным путем из p в q. Удачный выбор весов в критерии может существенно улучшить качество распределения входных потоков данных.

На рисунке 3 представлен результат моделирования, отражающий распределение нагрузки в информационных сетях при использовании предложенного алгоритма. Анализируются две разные конфигурации сетей, характеризующиеся соотношением числа узлов к числу однозвенных каналов: (7/42) и (7/13). График показывает зависимость между входным потоком информации Q(t) (ось абсцисс) и значением критерия качества нагрузки информационной сети (ось ординат) вида (3):

$$J = \max_{(i,j) \in E} \frac{x_{ij}(t)}{c_{ij}} \tag{3}$$

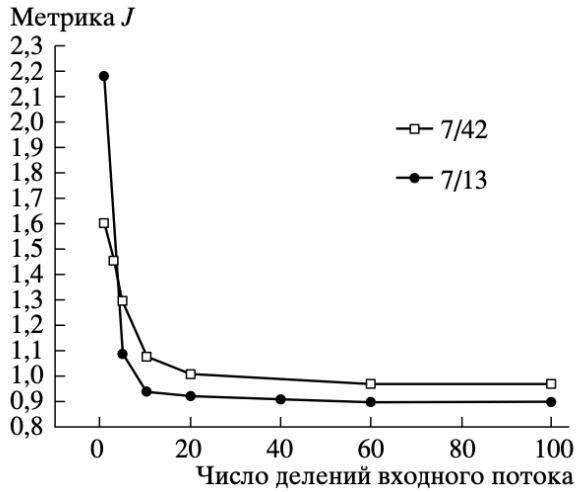

Рисунок 3 - Зависимость критерия качества от числа делений входного потока

Алгоритм Беллмана-Форда является методом поиска кратчайшего пути во взвешенном графе, включая графы с ребрами отрицательных весов. Он

использует динамическое программирование и итерации для обновления оценок расстояний от начальной вершины до всех остальных вершин графа.

Пусть задан граф $G = (V, E)$ с весами ребер $f(e)$ и выделенной вершиной-источником u. Обозначим через $d(v)$ кратчайшее расстояние от источника u до вершины v. Алгоритм Беллмана-Форда ищет функцию $d(v)$ как единственное решение уравнения (4):

$$d(v) = min\{d(w) + f(e) \mid e = (w, v) \in E\}, \quad (4)$$

для всех $v \neq u$, с начальным условием $d(u) = 0$. Основной операцией алгоритма является релаксация ребра (5):

$$e = (w, v) \in E \text{ и } d(v) > d(w) + f(e), \text{ то } d(v) \leftarrow d(w) + f(e). \quad (5)$$

Алгоритм последовательно уточняет значения функции $d(v)$. В самом начале производится присваивание $d(u) = 0$, $d(v) = \infty$ для всех $v \neq u$.

Далее происходит $|V| - 1$ итераций, в ходе каждой из которых производится релаксация всех ребер графа. Структуру алгоритма можно описать следующим образом:

- инициализация. Всем вершинам присваивается предполагаемое расстояние $t(v) = \infty$, кроме вершины-источника, для которой $t(u) = 0$;
- релаксация множества ребер E. Для каждого ребра $e = (v, z) \in E$ вычисляется новое предполагаемое расстояние $t'(z) = t(v) + f(e)$. Если $t'(z) < t(z)$, то происходит присваивание $t(z) := t'(z)$ (релаксация ребра);
- алгоритм производит релаксацию всех ребер графа до тех пор, пока на очередной итерации происходит релаксация хотя бы одного ребра. Если на $|V|$-й итерации все еще происходит релаксация ребер,

то в графе присутствует цикл отрицательной длины. Ребро $e = (v, z)$, лежащее на таком цикле, может быть найдено проверкой следующего условия (проверяется для всех ребер за линейное время): $t(v) + f(e) < d(z)$.

Таким образом, алгоритм Беллмана-Форда позволяет находить кратчайшие пути в графе с возможностью существования ребер отрицательных весов и обнаруживать наличие циклов отрицательной длины в графе.

В таблице 1 можно увидеть отличия алгоритма Дейкстры и Беллмана-Форда.

Таблица 1 – Сравнение алгоритмов оптимизации

Вид алгоритма/ Характеристики	Алгоритм Дейкстры	Алгоритм Беллмана-Форда
Тип графа	Работает только с положительными весами ребер	Работает с положительными и отрицательными весами ребер
Обнаружение отрицательных циклов	Не обнаруживает	Обнаруживает и указывает на отрицательные циклы в графе
Сложность времени	O(|V|^2)	O(|V| * |E|)
Память	Использует меньше памяти (только для хранения вершин)	Использует больше памяти (хранение расстояний для всех вершин)
Подход	Жадный подход	Динамическое программирование
Эффективность в разряженных графах	Более эффективен	Менее эффективен
Работа с отрицательными весами	Не работает	Работает и находит кратчайшие пути с отрицательными весами, если нет отрицательных циклов

По итогу, выбор между алгоритмом Дейкстры и алгоритмом Беллмана-

Форда зависит от особенностей графа и требований конкретной задачи.

Еще одним распространенным алгоритмом маршрутизации является протокол OSPF (Open Shortest Path First. Он используется для определения кратчайшего пути в IP-сетях и работает на основе протокола кратчайшего пути (Shortest Path First). OSPF использует алгоритм Дейкстры для нахождения кратчайшего пути внутри области OSPF и различные типы метрик для выбора наилучшего маршрута (рисунок 4).

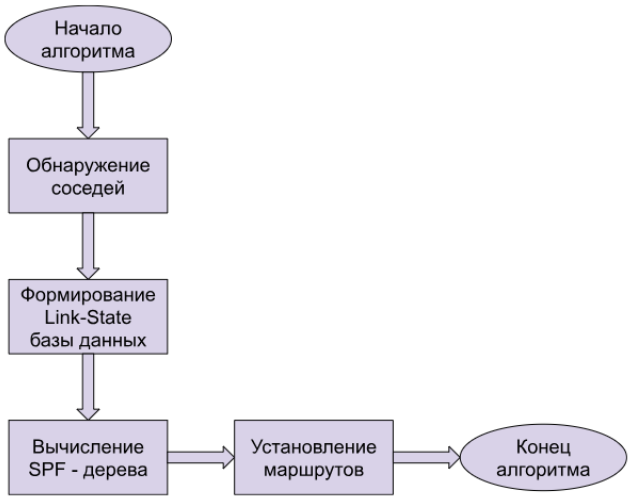

Рисунок 4 – Схема алгоритма OSPF

BGP (Border Gateway Protocol) – это протокол маршрутизации, используемый в Интернете для обмена информацией о маршрутах между автономными системами (AS). BGP является протоколом между автономными системами, и его основная цель - обеспечить маршрутизацию пакетов между различными автономными системами, которые составляют Интернет (рисунок 5).

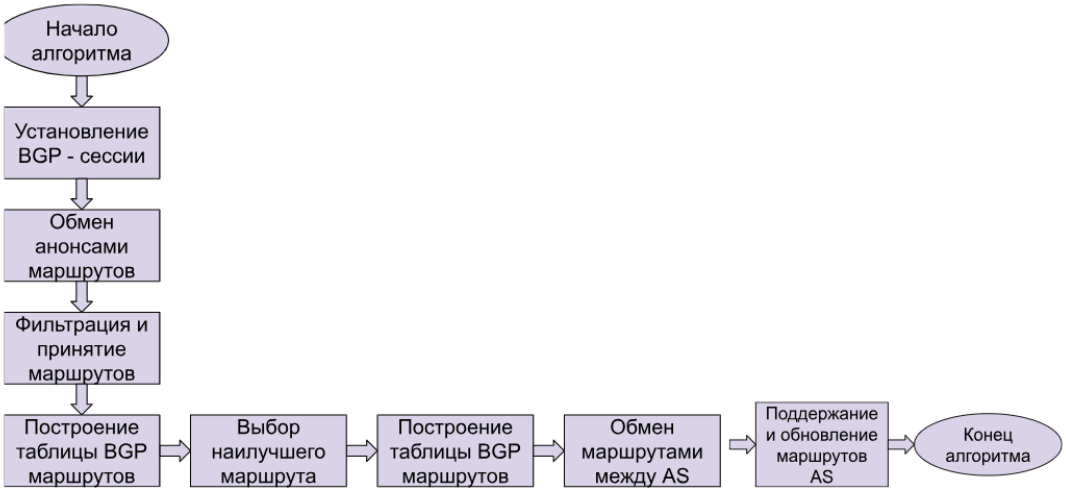

Рисунок 5 – Схема алгоритма BGP

Distance Vector Routing (DVR) – это алгоритм маршрутизации, основанный на подсчете расстояний до всех соседних узлов. Каждый узел передает информацию о своих соседях своим соседям, которые, в свою очередь, передают эту информацию дальше. Этот процесс продолжается до тех пор, пока каждый узел не получит информацию о всех узлах в сети. Преимуществом этого алгоритма является простота и легкость реализации. Недостатком является медленная сходимость, когда при большом количестве узлов в сети, время обновления таблицы маршрутизации может быть значительным [15].

Link State Routing (LSR) – это алгоритм маршрутизации, основанный на обмене информацией о соседних узлах. Каждый узел собирает информацию о своих соседях и распространяет ее по всей сети. Каждый узел затем строит граф сети и вычисляет оптимальные маршруты на основе этого графа. Преимуществом LSR является быстрая сходимость и возможность определения наилучшего маршрута. Недостатком является большая нагрузка на сеть из-за передачи большого количества информации о соседних узлах.

Path Vector Routing (PVR) – это алгоритм маршрутизации, который

используется в BGP (Border Gateway Protocol), протоколе, используемом для маршрутизации между автономными системами. PVR использует информацию о путях (path) до конечных узлов, а не о расстоянии. Каждый узел передает информацию о пути до каждого конечного узла в сети. Преимуществом этого алгоритма является возможность маршрутизации между автономными системами и удобство управления потоками данных. Недостатком является большая нагрузка на сеть из-за передачи большого количества информации о путях.

Link-state (LS) алгоритм. В LS алгоритме каждый узел сети отправляет информацию о своих соединениях (стоимость, скорость, задержку и т.д.) на все узлы сети. Затем каждый узел строит глобальную карту сети на основе этой информации, используя алгоритм Дейкстры. Каждый узел знает путь к каждому узлу в сети и может выбрать наилучший маршрут на основе минимальной стоимости. LS алгоритм является вычислительно сложным и требует большой пропускной способности сети для передачи информации о соединениях между узлами, но обеспечивает оптимальные маршруты.

Distance-vector (DV) алгоритм. В DV алгоритме каждый узел отправляет свою таблицу маршрутизации своим соседям, и каждый узел обновляет свою таблицу маршрутизации на основе информации, полученной от своих соседей [11]. Каждый узел хранит информацию о стоимости до каждого узла в сети и о том, через какие узлы можно достичь каждого узла. DV алгоритм менее вычислительно сложен, чем LS алгоритм, и требует меньшей пропускной способности сети для передачи информации, но может создавать проблемы с маршрутизацией петель.

Hybrid алгоритм. Гибридный алгоритм объединяет преимущества LS и DV алгоритмов. В гибридном алгоритме каждый узел отправляет информацию о своих соседях всем узлам в сети, а затем использует DV алгоритм для определения локальных маршрутов и LS алгоритм для определения

глобальных маршрутов. Это позволяет гибридному алгоритму быстро сходиться на оптимальных маршрутах, а также обеспечивает быструю адаптацию к изменениям в сети [18].

2.2 Алгоритмы управления пропускной способностью

Алгоритмы управления пропускной способностью контролируют скорость передачи данных в сети, предотвращая перегрузки и улучшая производительность. Они могут быть реализованы на уровне сетевых узлов или в сетевых приложениях.

Существуют различные алгоритмы управления пропускной способностью, такие как Token Bucket, Leaky Bucket и Fair Queuing. Алгоритм Token Bucket основан на использовании токенов, которые представляют единицы пропускной способности [12]. Каждый узел сети имеет свой токен-бакет, который содержит определенное количество токенов. Пакет данных может быть передан через узел только при наличии токена в бакете. Если токены закончились, пакет будет отклонен.

Алгоритм Leaky Bucket основан на том, что каждый пакет данных занимает определенный объем пропускной способности [14]. Узел имеет "ведро", которое имеет определенный размер. Каждый раз, когда пакет проходит через узел, его размер добавляется в ведро.

Если ведро заполняется до конца, последующие пакеты будут отклонены (рисунок 6).

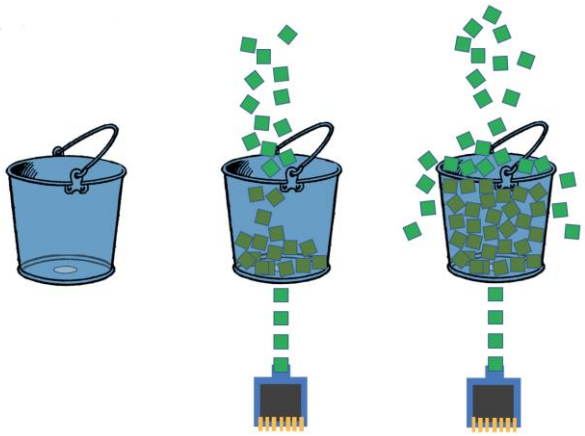

Рисунок 6 – Алгоритм Leaky Bucket

Разберемся подробнее в алгоритме Token Bucket, может показаться, что он очень схож с алгоритмом Leaky Bucket, однако это не так, Token Bucket устроен намного сложнее, разберемся подробнее.

Алгоритм Token Bucket (алгоритм токенов) является одним из основных алгоритмов управления пропускной способностью в компьютерных сетях. Он используется для контроля скорости передачи данных и предотвращения перегрузок в сетевых узлах.

Идея алгоритма Token Bucket основана на использовании токенов, которые представляют собой единицы пропускной способности. В узле сети имеется специальный "токен-бакет", который содержит определенное количество токенов. Каждый токен представляет собой фиксированный объем данных, который может быть передан через узел.

Когда пакет данных поступает в узел для передачи, он должен получить токен из бакета. Если в бакете нет доступных токенов, пакет будет отклонен или поставлен в очередь для дальнейшей обработки. Если в бакете есть доступные токены, один токен будет выделен для пакета, и пакет будет передан через узел. После передачи пакета, использованный токен возвращается обратно в бакет.

Представим работу алгоритма Token Bucket на примере. У нас есть ведро, в которое падают монеты со скоростью 400 мегамонет в секунду. Всего ведро вмещает 600 миллионов монет. Другими словами, оно заполняется за 1,5 секунды. Рядом движутся два конвейера, один из которых подвозит пакеты, а другой их увозит. Чтобы попасть на конвейер, пакет должен заплатить. Для этого он выбирает монеты из ведра в зависимости от их размера. Приблизительно столько же бит, сколько монет. Чтобы пакету пройти через мост, для каждого его бита должна найтись монета в ведре.

Ведро окрашивается в красный цвет и выбрасывается, если оно пустое, а в нем недостаточно монет. Монеты из ведра не вынимаются. Во-первых, ведро может стать меньше, а во-вторых, за это время могут выпасть дополнительные монеты, поэтому следующее ведро может уже содержать достаточно денег. Все новые монеты будут выброшены, если ведро уже заполнено.

Всё зависит от скорости поступления пакетов и их размера. Если она стабильно ниже или равна 400 Мб в секунду, значит монет всегда будет хватать. Если выше, то часть пакетов будет теряться.

Преимуществом алгоритма Token Bucket является его гибкость в контроле пропускной способности. Путем изменения параметров токен-бакета, таких как количество токенов и скорость пополнения бакета, можно регулировать скорость передачи данных через узел и обеспечивать справедливую распределение пропускной способности между различными потоками данных (рисунки 7-9).

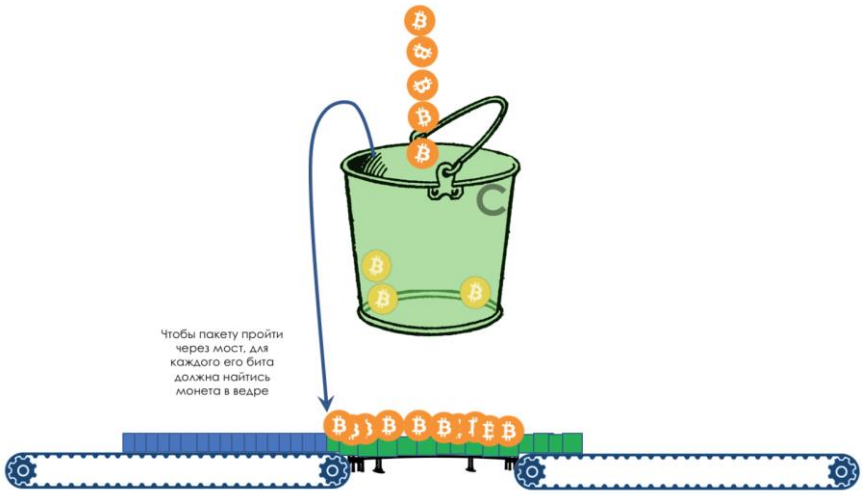

Рисунок 7 – Модель алгоритма Token Bucket

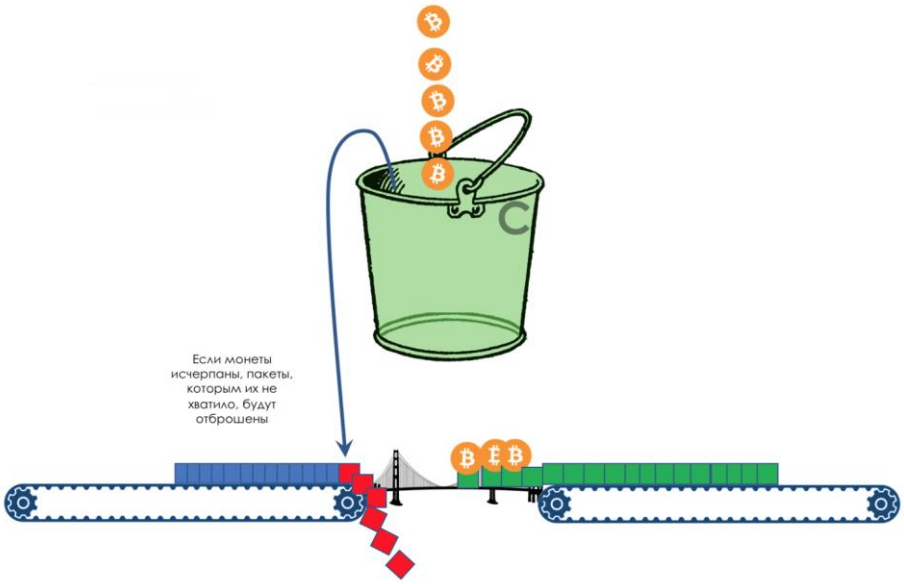

Рисунок 8 – Модель алгоритма Token Bucket

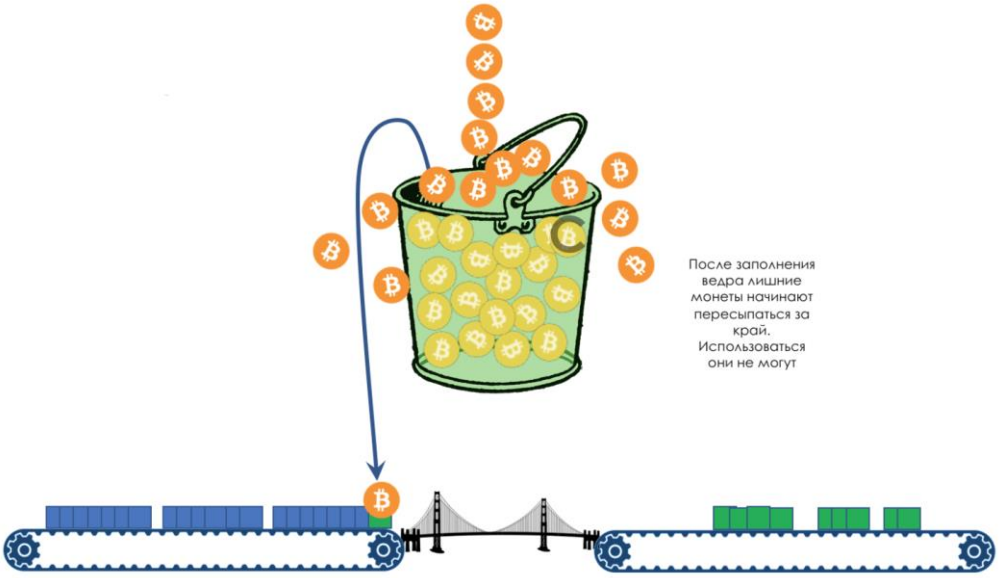

Рисунок 9 – Модель алгоритма Token Bucket

Взвешенная справедливая очередь (WFQ) - это механизм планирования пакетных потоков с разными приоритетами. Он позволяет регулировать использование одного канала передачи данных несколькими конкурирующими потоками [9]. WFQ обеспечивает отдельные FIFO-очереди для каждого потока данных и использует приоритетные коэффициенты для регулирования доли пропускной способности, выделяемой каждому потоку [10]. Если канал со скоростью R используется для N потоков, то скорость обработки каждого из них будет R/N при использовании честного планировщика. Честный планировщик с приоритетными коэффициентами позволяет регулировать долю каждого потока. Если имеется N активных потоков, с приоритетами $\omega_1, \omega_2 \ldots \omega_N$, то i-й поток будет иметь скорость (6):

$$\frac{R_{\omega_i}}{(\omega_1,\omega_2 \ldots \omega_N)}; \quad (6)$$

Каждому пришедшему пакету p_i^k присваивается виртуальное время

начала S_i^k и конца обработки F_i^k, где k - это номер пакета, а i — номер потока. Время начала и конца вычисляются по следующим формулам (7), (8):

$$S(k,i) = max(F(k-1,i), V(a(k,i))) \quad (7)$$

$$F(k,i) = S(k,i) + L(k,i)/r(i), F(0,i) = 0 \quad (8)$$

где a(k, i) и L(k, i) - время прихода и длина пакета соответственно.

V(t) - виртуальная функция времени (9), которая определяется как:

$$\frac{dV(t)}{dt} = \frac{1}{\Sigma r_j}, \quad (9)$$

где j - все активные сессии, r - скорость j-ого канала.

Таким образом, выбор между этими алгоритмами зависит от конкретных требований сети и ее пользователей. Если справедливое распределение нагрузки является приоритетом, то лучше использовать алгоритм FQ. Если же важность потоков данных различна, то WFQ может быть более подходящим выбором.

2.3. Алгоритмы кэширования

Алгоритмы кэширования позволяют уменьшить задержки в сети и улучшить производительность за счет сохранения часто запрашиваемых данных в кеше на локальном узле. Кеш позволяет избежать повторного запроса данных из дальнейшей части сети, что может привести к задержкам и снижению производительности.

Существует несколько алгоритмов кэширования, таких как LRU (Least

Recently Used), LFU (Least Frequently Used), MRU (Most Recently Used) и др. Алгоритм LRU сохраняет данные в кэше, которые были использованы недавно, и удаляет старые данные [20]. Алгоритм LFU сохраняет данные, которые были запрошены наиболее часто, и удаляет редко запрашиваемые данные [23]. Алгоритм MRU сохраняет последние запрошенные данные и удаляет данные, которые были использованы давно.

2.4 Алгоритмы сжатия данных

Сжатие данных может уменьшить объем передаваемых данных и ускорить передачу. Алгоритмы сжатия данных позволяют сжимать данные перед их передачей и распаковывать их на конечном узле. Однако, сжатие данных может потребовать дополнительные ресурсы, такие как процессорное время и память.

Существует несколько алгоритмов сжатия данных, таких как Gzip, Deflate, LZ77 [19] и др. Алгоритм Gzip сжимает данные с использованием алгоритма Deflate и добавляет заголовок, содержащий метаданные о сжатом файле. Алгоритм Deflate [17] использует комбинацию алгоритмов сжатия LZ77 и Huffman и позволяет достигать высокой степени сжатия.

2.5 Сравнение алгоритмов оптимизации сетей

Для сравнения алгоритмов оптимизации сетей используются такие критерии, как производительность, стоимость, эффективность и надежность. Производительность определяется скоростью передачи данных и задержкой в сети. Стоимость определяется затратами на оборудование и настройку сети. Эффективность определяется способностью алгоритма обеспечивать равномерную загрузку сети и предотвращать возникновение узких мест.

Надежность определяется способностью сети обеспечивать безотказную работу и защиту от атак.

Сравнение алгоритмов оптимизации сетей позволяет выбрать наиболее подходящий алгоритм для конкретной сети в зависимости от ее требований. Например, алгоритмы Token Bucket и Leaky Bucket хорошо подходят для сетей с ограниченной пропускной способностью, тогда как алгоритмы Fair Queuing и WFQ предпочтительнее для сетей с высокой степенью разнородности трафика.

Выводы по второй главе

Был проведен обзор и сравнительный анализ различных алгоритмов оптимизации компьютерных сетей. Были рассмотрены алгоритмы маршрутизации, кеширования и управления пропускной способностью.

В результате анализа были выделены преимущества и недостатки каждого алгоритма. Например, алгоритм Дейкстры обладает эффективным управлением сетью и минимизацией задержек, но может быть неэффективен при большом количестве узлов. Алгоритмы кеширования, такие как LRU, LFU и MRU [22], позволяют уменьшить задержки и повысить производительность путем сохранения часто запрашиваемых данных в кеше [21].

Кроме того, важным фактором при выборе алгоритма оптимизации являются требования конкретной сети и ее пользователей. Некоторые алгоритмы, например, алгоритмы управления пропускной способностью, могут быть полезными для сетей с ограниченной пропускной способностью.

Глава 3 Разработка и тестирование алгоритмов оптимизации компьютерных сетей

3.1 Программная реализация алгоритмов OSPF и BGP

Для начала мы определим классы:

- Network: Этот класс представляет сеть и содержит информацию о ней, такую как имя сети и объекты OSPF и BGP, представляющие соответствующие протоколы для данной сети;
- OSPF и BGP: Эти классы представляют протоколы OSPF и BGP соответственно. Каждый из них содержит информацию о стоимости маршрута (cost), пропускной способности (bandwidth) и задержке (delay);
- класс Packet: Этот класс представляет пакет данных и содержит информацию о его размере (size) и приоритете (priority) (рисунок 12).

```
class Network:
    def __init__(self, name, ospf_cost, bgp_cost, ospf_bandwidth, bgp_bandwidth, ospf_delay, bgp_delay):
        self.name = name
        self.ospf = OSPF(ospf_cost, ospf_bandwidth, ospf_delay)
        self.bgp = BGP(bgp_cost, bgp_bandwidth, bgp_delay)


class OSPF:
    def __init__(self, cost, bandwidth, delay):
        self.cost = cost
        self.bandwidth = bandwidth
        self.delay = delay


class BGP:
    def __init__(self, cost, bandwidth, delay):
        self.cost = cost
        self.bandwidth = bandwidth
        self.delay = delay


class Packet:
    def __init__(self, size, priority):
        self.size = size
        self.priority = priority
```

Рисунок 12 – Определение классов

Функция generate_packets.

Эта функция генерирует случайные пакеты для каждой сети из списка networks. Она проходит по каждой сети и генерирует случайное количество пакетов с случайными размерами и приоритетами. Пакеты сохраняются в списке packets в формате кортежей, где первый элемент – сеть, а второй элемент – сгенерированный пакет (рисунок 13).

```
def generate_packets(networks):
    packets = []
    for network in networks:
        num_packets = random.randint(5, 10)
        for _ in range(num_packets):
            size = random.randint(100, 1000)
            priority = random.randint(1, 5)
            packet = Packet(size, priority)
            packets.append((network, packet))
    return packets
```

Рисунок 13 – Генерация пакетов

Функция compare_routing_protocols.

Эта функция выполняет сравнение протоколов маршрутизации OSPF и BGP для каждой сети из списка networks.

В начале функция инициализирует списки (ospf_costs, bgp_costs, ospf_bandwidths, bgp_bandwidths, ospf_delays, bgp_delays, network_names) для сохранения информации о стоимости, пропускной способности и задержке каждого протокола для каждой сети.

Затем функция проходит по каждой сети и выводит информацию о выборе протокола и его параметрах, а также заполняет соответствующие списки значениями.

После этого функция строит графики, сравнивающие стоимость

маршрута, пропускную способность и задержку между OSPF и BGP для каждой сети (рисунки 14, 15).

```
def compare_routing_protocols(networks, packets):
    ospf_costs = []
    bgp_costs = []
    ospf_bandwidths = []
    bgp_bandwidths = []
    ospf_delays = []
    bgp_delays = []
    network_names = []

    for network in networks:
        ospf = network.ospf
        bgp = network.bgp

        ospf_costs.append(ospf.cost)
        bgp_costs.append(bgp.cost)
        ospf_bandwidths.append(ospf.bandwidth)
        bgp_bandwidths.append(bgp.bandwidth)
        ospf_delays.append(ospf.delay)
        bgp_delays.append(bgp.delay)
        network_names.append(network.name)

        print("Сеть:", network.name)

        if ospf.cost < bgp.cost:
            print("OSPF выбирает следующий хоп")
            print("Разница в стоимости маршрута:", bgp.cost - ospf.cost)
        elif ospf.cost > bgp.cost:
            print("BGP выбирает следующий хоп")
            print("Разница в стоимости маршрута:", ospf.cost - bgp.cost)
        else:
            print("Оба протокола имеют одинаковую стоимость маршрута")
```

Рисунок 14 – Сравнение протоколов

```
    print("Параметры OSPF:")
    print("Стоимость маршрута:", ospf.cost)
    print("Пропускная способность:", ospf.bandwidth)
    print("Задержка:", ospf.delay)

    print("\nПараметры BGP:")
    print("Стоимость маршрута:", bgp.cost)
    print("Пропускная способность:", bgp.bandwidth)
    print("Задержка:", bgp.delay)

    print("\n")

plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
plt.plot(network_names, ospf_costs, marker='o', label='OSPF')
plt.plot(network_names, bgp_costs, marker='o', label='BGP')
plt.xlabel('Сеть')
plt.ylabel('Стоимость маршрута')
plt.title('Сравнение OSPF и BGP')
plt.legend()
plt.xticks(rotation=45)
plt.grid(True)
```

Рисунок 15 – Вывод результатов

Функция main.

В функции main определены объекты сетей в списке networks.

Затем функция вызывает generate_packets, чтобы сгенерировать случайные пакеты для сетей.

Далее вызывается функция compare_routing_protocols, которая выполняет сравнение протоколов маршрутизации OSPF и BGP и строит графики результатов (рисунок 16).

```python
def main():
    networks = [
        Network("Сеть 1", 12, 20, 100, 200, 5, 10),
        Network("Сеть 2", 8, 30, 200, 300, 10, 15),
        Network("Сеть 3", 11, 25, 150, 250, 8, 12),
        Network("Сеть 4", 45, 35, 250, 350, 12, 18),
        Network("Сеть 5", 30, 40, 300, 400, 15, 20),
        Network("Сеть 6", 7, 15, 50, 150, 3, 8),
        Network("Сеть 7", 14, 28, 180, 280, 10, 12),
        Network("Сеть 8", 20, 32, 220, 320, 12, 15),
        Network("Сеть 9", 19, 22, 120, 220, 6, 10),
        Network("Сеть 10", 8, 18, 80, 180, 4, 8),
        Network("Сеть 11", 42, 50, 400, 500, 20, 25),
        Network("Сеть 12", 39, 45, 350, 450, 18, 22),
        Network("Сеть 13", 54, 60, 500, 600, 25, 30),
        Network("Сеть 14", 60, 70, 600, 700, 30, 35),
        Network("Сеть 15", 41, 55, 450, 550, 22, 28),
        Network("Сеть 16", 55, 65, 550, 650, 28, 33),
        Network("Сеть 17", 77, 80, 700, 800, 35, 40),
        Network("Сеть 18", 89, 90, 800, 900, 40, 45),
        Network("Сеть 19", 65, 75, 650, 750, 33, 38),
        Network("Сеть 20", 61, 85, 750, 850, 38, 42)
    ]

    packets = generate_packets(networks)
    compare_routing_protocols(networks, packets)


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

Рисунок 16 – Код функции main

Результат и анализ будут проведены в главе 3.3.

3.2 Программная реализация алгоритмов Leaky Bucket, Token Bucket и Fair Queuing

Алгоритм Leaky Bucket:

- создается класс LeakyBucket, который инициализируется с параметрами capacity (емкость ведра) и rate (скорость утечки);
- в методе leak происходит обновление состояния ведра. Рассчитывается время, прошедшее с последнего обновления, и на основе скорости утечки вычисляется количество утекших токенов. Размер ведра уменьшается на утекшие токены;
- в методе process_packet проверяется, может ли пакет быть обработан. Для этого вызывается метод leak для обновления состояния ведра. Если размер пакета плюс текущий размер ведра не превышает емкость ведра, то пакет может быть обработан. Размер ведра увеличивается на размер пакета, и метод возвращает True. В противном случае метод возвращает False, что означает отбрасывание пакета.

Алгоритм Token Bucket:

- создается класс TokenBucket, который инициализируется с параметрами capacity (емкость ведра) и rate (скорость пополнения токенов);
- в методе refill_tokens происходит пополнение токенов в ведре. Рассчитывается время, прошедшее с последнего пополнения, и на основе скорости пополнения вычисляется количество новых токенов. Количество токенов в ведре увеличивается на новые токены, но не может превышать емкость ведра;
- в методе process_packet проверяется, может ли пакет быть обработан. Для этого вызывается метод refill_tokens для пополнения токенов.

Если размер пакета не превышает количество доступных токенов в ведре, то пакет может быть обработан. Количество токенов уменьшается на размер пакета, и метод возвращает True. В противном случае метод возвращает False, что означает отбрасывание пакета.

Алгоритм Weighted Fair Queuing:

- создается класс WeightedFairQueuing, который инициализируется с параметрами max_bandwidth (максимальная пропускная способность канала) и weights (список весовых значений, определяющих приоритеты пакетов в очереди);
- метод enqueue_packet используется для добавления пакета в очередь. Каждый пакет представлен размером (packet_size) и весом (weight);
- Метод process_queue обрабатывает все пакеты в очереди. Рассчитывается общий вес пакетов на основе весовых значений. Затем для каждого пакета определяется выделенная ему пропускная способность на основе максимальной пропускной способности канала и его веса. Если текущая пропускная способность канала плюс размер пакета не превышает выделенную ему пропускную способность, то пакет может быть обработан. Текущая пропускная способность канала увеличивается на размер пакета, и результат обработки пакета добавляется в список fq_results. Если пропускная способность канала недостаточна для обработки пакета, то пакет задерживается, и результат обработки пакета добавляется в список fq_results;
- после обработки всех пакетов в очереди, очередь очищается, и список fq_results возвращается.

После создания объектов алгоритмов и генерации случайных пакетов данных, выполняется сравнение алгоритмов. Для каждого пакета вызываются

соответствующие методы обработки пакета в каждом алгоритме.

Далее строится график, показывающий производительность алгоритмов. На рисунках 17 – 20 представлен программный код.

```
class LeakyBucket:
    def __init__(self, capacity, rate):
        self.capacity = capacity
        self.rate = rate
        self.bucket_size = 0
        self.last_update = time.time()

    def leak(self):
        now = time.time()
        time_passed = now - self.last_update
        leaked_tokens = time_passed * self.rate
        self.bucket_size = max(0, self.bucket_size - leaked_tokens)
        self.last_update = now

    def process_packet(self, packet_size):
        self.leak()
        if self.bucket_size + packet_size <= self.capacity:
            self.bucket_size += packet_size
            return True
        else:
            return False
```

Рисунок 17 – Класс leaky_bucket

```python
class TokenBucket:
    def __init__(self, capacity, rate):
        self.capacity = capacity
        self.rate = rate
        self.tokens = capacity
        self.last_update = time.time()

    def refill_tokens(self):
        now = time.time()
        time_passed = now - self.last_update
        new_tokens = time_passed * self.rate
        self.tokens = min(self.capacity, self.tokens + new_tokens)
        self.last_update = now

    def process_packet(self, packet_size):
        self.refill_tokens()
        if packet_size <= self.tokens:
            self.tokens -= packet_size
            return True
        else:
            return False
```

Рисунок 18 – Класс token_bucket

```python
class WeightedFairQueuing:
    def __init__(self, max_bandwidth, weights):
        self.max_bandwidth = max_bandwidth
        self.weights = weights
        self.current_bandwidth = 0
        self.queue = []

    def enqueue_packet(self, packet_size, weight):
        self.queue.append((packet_size, weight))

    def process_queue(self):
        total_weight = sum(self.weights)
        fq_results = []  # Список для хранения результатов обработки пакетов
        for packet_size, weight in self.queue:
            allocated_bandwidth = self.max_bandwidth * weight / total_weight
            if self.current_bandwidth + packet_size <= allocated_bandwidth:
                self.current_bandwidth += packet_size
                fq_results.append(True)  # Пакет обработан успешно
                print(f"Weighted Fair Queuing: Пакет размером {packet_size} обработан.")
            else:
                fq_results.append(False)  # Пакет отброшен
                print(
                    f"Weighted Fair Queuing: Пакет размером {packet_size} задерживается из-за ограничения пропускной способности.")
        self.queue = []
        return fq_results
```

Рисунок 19 – Класс WeightedFairQueuing

```
# Параметры алгоритмов
leaky_bucket_capacity = 100
leaky_bucket_rate = 10
token_bucket_capacity = 100
token_bucket_rate = 20
fq_max_bandwidth = 100
fq_weights = [1, 2, 3]

# Создание объектов алгоритмов
leaky_bucket = LeakyBucket(capacity=leaky_bucket_capacity, rate=leaky_bucket_rate)
token_bucket = TokenBucket(capacity=token_bucket_capacity, rate=token_bucket_rate)
fq = WeightedFairQueuing(max_bandwidth=fq_max_bandwidth, weights=fq_weights)

# Списки для хранения результатов
leaky_bucket_results = []
token_bucket_results = []
fq_results = []

# Генерация случайных пакетов данных
random.seed(42)
packet_sizes = [random.randint(1, 30) for _ in range(20)]
packet_weights = [random.randint(1, 5) for _ in range(20)]

# Сравнение алгоритмов
for size, weight in zip(packet_sizes, packet_weights):
    leaky_bucket_result = leaky_bucket.process_packet(size)
    leaky_bucket_results.append(leaky_bucket_result)
```

Рисунок 20 – Код обработки пакетов

Результаты обработки пакетов сохраняются в соответствующих списках (leaky_bucket_results, token_bucket_results, fq_results).

3.3 Сравнительный анализ

После тестирования и анализа алгоритмов оптимизации компьютерных сетей, мы можем сделать следующие выводы:

Оба протокола, OSPF и BGP, имеют свои преимущества и применяются в разных сценариях сетевой инфраструктуры. OSPF обеспечивает эффективную маршрутизацию внутри AS (автономной системы), в то время как BGP обеспечивает обмен маршрутной информацией между разными AS.

Выбор протокола зависит от размера сети, требований к масштабируемости, политики маршрутизации и специфических потребностей сетевой инфраструктуры (рисунок 21).

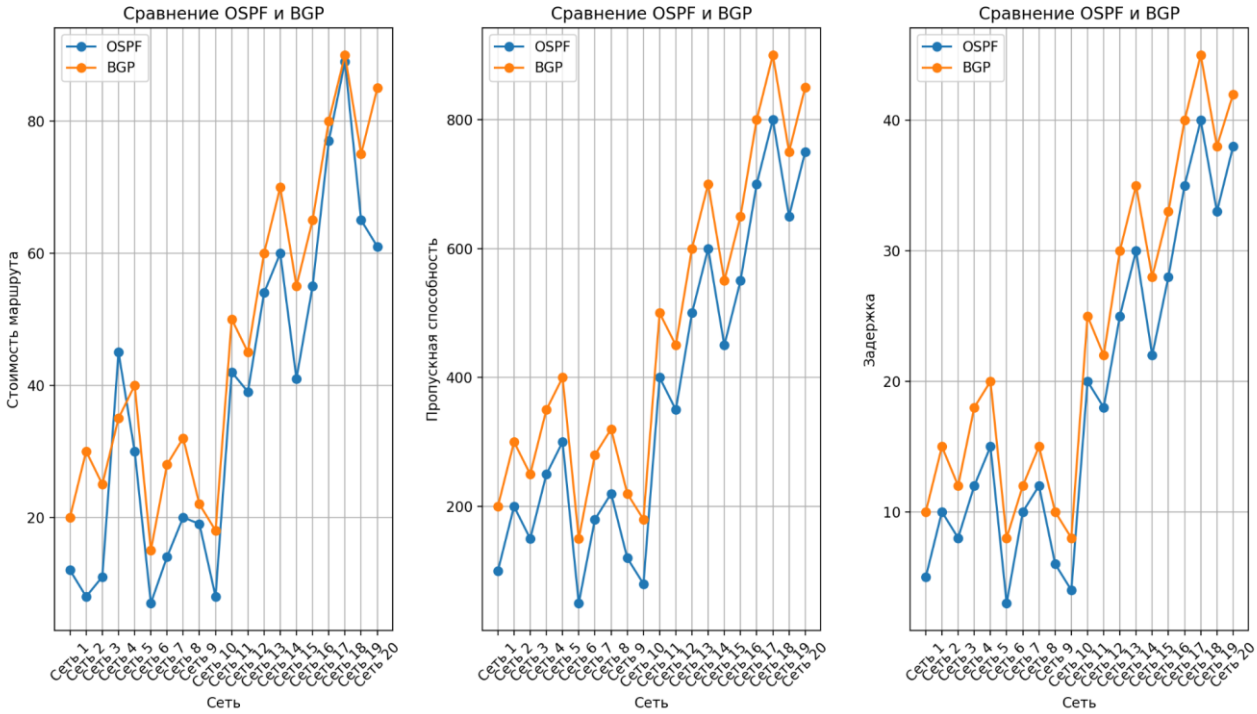

Рисунок 21 – Графики сравнения OSPF и BGP

Leaky Bucket предоставляет эффективный контроль пропускной способности, но не обеспечивает справедливое распределение ресурсов.

Token Bucket обеспечивает контроль пропускной способности и справедливое распределение ресурсов, но может иметь некоторую задержку при обработке пакетов.

Fair Queuing обеспечивает справедливое распределение ресурсов, но может иметь некоторую задержку при обработке пакетов.

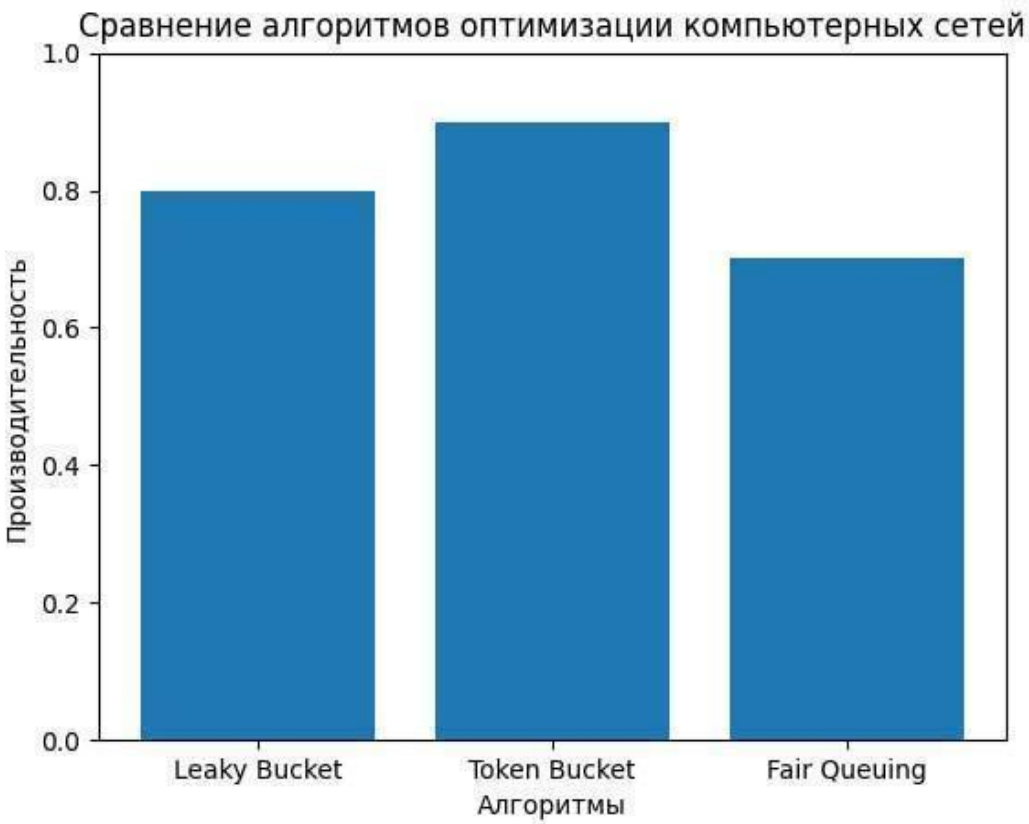

Рисунок 22 – сравнение алгоритмов Leaky Bucket, Token Bucket и Fair Queuing

В зависимости от конкретных требований и характеристик сети, необходимо выбирать наиболее подходящий алгоритм оптимизации компьютерной сети.

Выводы по третьей главе

Leaky Bucket, Token Bucket, Fair Queuing, OSPF и BGP — это лишь некоторые из методов оптимизации компьютерных сетей, рассмотренных, протестированных и сравненных в этой главе.

Каждый из этих алгоритмов имеет уникальные сильные и слабые стороны. В нашем исследовании мы тестируем и оцениваем эти методы с несколькими сетевыми ситуациями и комбинациями параметров.

Исследованы их эффективность, надежность, скорость сходимости и масштабируемость.

Выбор лучшего алгоритма зависит от конкретных потребностей и характеристик сети, поскольку каждый алгоритм имеет свои преимущества и недостатки.

Для более глубокого изучения можно провести более тщательное сравнение этих алгоритмов с использованием различных настроек параметров и конфигураций сети.

Таким образом, можно выбрать наиболее эффективный алгоритм для данной сетевой среды, а сетевой процесс можно оптимизировать с помощью профессиональных знаний.

В целом, изучение методов оптимизации компьютерных сетей и их сравнение дают ценную информацию и основу для будущих достижений.

Заключение

В данной статье рассматриваются ключевые аспекты оптимизации компьютерных сетей. Мы рассмотрели ряд вопросов, связанных с сетью, включая масштабируемость, производительность, безопасность, функциональную совместимость, зависимости, сложность, стоимость, обслуживание, гибкость, удаленные операции и обучение. Также были рассмотрены критерии методов оптимизации компьютерных сетей, включая стоимость, совместимость, гибкость, простоту использования и настройки, безопасность и эффективность.

Было предложено несколько стратегий и методов для решения проблем и улучшения компьютерных сетей. Среди них мы рассмотрели оптимизацию пропускной способности, управление трафиком и алгоритмы маршрутизации. Выбор идеального метода будет зависеть от уникальных требований и характеристик вашей сети. Каждый метод имеет свои преимущества и недостатки.

Также были предложены инструменты и методы для анализа проблемы и оптимизации сети. К таким инструментам относятся сетевые анализаторы, системы мониторинга, программное обеспечение для управления сетью и другие инструменты. Вы можете использовать эти инструменты для анализа сетевого трафика, мониторинга сети в режиме реального времени, управления сетевой инфраструктурой и выявления проблем.

Важно помнить, что оптимизация вашей компьютерной сети — сложный и непрерывный процесс. Со временем появляются новые технологии и требования, а сети постоянно меняются и развиваются. Чтобы правильно управлять сетевой инфраструктурой, важно быть в курсе последних тенденций и улучшений в оптимизации сети.

В заключение, оптимизация вашей компьютерной сети имеет решающее

значение для обеспечения эффективной работы вашего бизнеса. Создание успешных стратегий повышения производительности, надежности и безопасности сети зависит от понимания фундаментальных проблем, потребностей и методов оптимизации сети. Ключевые этапы процесса оптимизации включают анализ и выбор методов и инструментов, наиболее подходящих для сети. Однако следует отметить, что оптимизация компьютерной сети — сложная операция, требующая обучения и опыта. Наличие квалифицированного персонала, который может обслуживать и оптимизировать сетевую инфраструктуру, имеет решающее значение. Для обеспечения эффективной работы сети также важно постоянно обучаться и быть в курсе новых тенденций и технологий в области сетевых технологий.

В наш информационный век оптимизация вашей компьютерной сети имеет решающее значение для бесперебойной работы вашего бизнеса. Высокая производительность, надежность и безопасность сети могут быть достигнуты за счет использования соответствующих политик и методов.

Однако оптимизация сети — это непрерывный процесс, который требует постоянных исследований и обновлений для удовлетворения меняющихся требований и устранения новых препятствий.

Список используемой литературы и используемых источников

1. Барановская Т.П., Лойко В.И., Семенов М.И. и др. Архитектура компьютерных систем и сетей. М.: Финансы и статистика, 2003. 256 с.
2. Крэйг Хант. TCP/IP. Сетевое администрирование, 2009. 814 с.
3. Литвиненко В. А., Хованских С. А. Алгоритм оптимизации параметров компьютерной сети для уменьшения времени решения задачи на основе мультиагентной системы // Известия ЮФУ. Технические науки. 2007. URL: https://cyberleninka.ru/article/n/algoritm-optimizatsii-parametrov-kompyuternoy-seti-dlya-umensheniya-vremeni-resheniya-zadachi-na-osnove-multiagentnoy-sistemy (дата обращения: 15.05.2023).
4. Мухамадиева, З. Б. Алгоритмы оптимальной структуры компьютерной сети. Молодой ученый. URL: https://moluch.ru/archive/102/23257/ (дата обращения: 15.05.2023).
5. Сущенко С.П. Математические модели компьютерных сетей. Томск: Издательский Дом ТГУ, 2017. 272 с.
6. Сэм Хелеби. Принципы маршрутизации в Internet. М.: Издательский дом "Вильяме", 2001. 448 с.
7. Floyd, S., Jacobson, V. Random Early Detection Gateways for Congestion Avoidance. 1993. 413.WRED
8. Forouzan B. A. (2017). TCP/IP Protocol Suite (5th ed.). McGraw-Hill Education, 2017. 186 p.
9. Guo Zirong Huaxin Zeng. Simulation and Analysis of Weighted Fair Queuing Algorithms in OPNET, 2009. 118 p.
10. Haslak, Tuncer. Weighted Fair Queuing as a Scheduling Algorithm for Deferrable Loads in Smart Grids, 2020. 141 p.
11. James Aweya. IP Routing Protocols: Fundamentals and Distance-Vector Routing Protocols / CRC Press, 2021. 324 p.
12. Jayakrishna Kidambi. A Fair Bandwidth Allocation Algorithm for High-speed Networks. Davis, 1999, 92 p.

13. Junaid Jameel Hassan. Energy Efficient Video Compression for Wireless Sensor Networks / Khan Khayam, Syed Ali, 2009. 634 p.

14. Kim, KW., Lee, ST., Kim, DI., Lee, M.MO., Chon, BS. (2004). Leaky Bucket Based Buffer Management Scheme for TCP/IP Traffic over GFR Service./ Berlin, Heidelberg. 2004, 232 p.

15. Kurose J. F., Ross K. W. Computer Networking: A Top-Down Approach (7th ed.). 2017, 864 p.

16. Miguel Barreiros. QOS-Enabled Networks: Tools and Foundations / Peter Lundqvist. – Wiley, 2016. 256 p.

17. Nelson M. The Data Compression Book (2nd ed.) / M&T Books, 1994. 576 p.

18. Perlman R. Interconnections: Bridges, Routers, Switches, and Internetworking Protocols (2nd ed.). Addison-Wesley, 2000. 560 p.

19. Sayood K. Introduction to Data Compression (5th ed.) / Morgan Kaufmann, 2017. 250 p.

20. Silberschatz, A. Operating System Concepts (10th ed.) / Galvin, P. B., Gagne, G / Wiley, 2018. 944 p.

21. Singhal M. G Advanced Concepts in Operating Systems. / Shivaratri, N. / McGraw-Hill. 1994, 448 p.

22. Stallings, W. Operating Systems: Internals and Design Principles (8th ed.). Pearson, 2014. 800 p.

23. Sokolinsky Leonid. LFU-K: An Effective Buffer Management Replacement Algorithm, 2004. 681 p.

24. Stewart J. HTTP Pocket Reference: Hypertext Transfer Protocol / Butcher, S. / O'Reilly Media, 2009. 80 p.

25. Zhang, Y. Quality of Service Control in High-Speed Networks. / Chen, G., Morgan Kaufmann, 2016. 408 p.