Титульный лист

Задание

Аннотация

В работе предлагается решение по разработке и внедрению адаптивного алгоритма для обеспечения увеличения скорости конвергенции протокола граничного шлюза (BGP). Апробация алгоритма производится в открытой платформе для автоматизации развертывания и масштабирования контейнеризированных приложений с применением модуля управления сетевой политикой.

Оглавление

Список используемых сокращений

AS – Autonomous systems/автономные системы

AWS – Amazon Web Servies/Сетевые сервисы Amazon

IGP – Interior gateway protocol/Протокол внутреннего шлюза

MED - Multi-Exit Discriminator

TCP - Transmission Control Protocol/Протокол управления передачей

MAPE – Средняя абсолютная ошибка

RAM – Random access memory/Оперативная память

CPU – Central processing unit, процессор

ЛВС – Локальная вычислительная сеть

ЦОД – Центр обработки данных

BGP – Borderline gateway protocol/Протокол граничного шлюза

EBGP – External borderline gate protocol/Внешний протокол граничного шлюза

IBGP – Internal borderline gate protocol/Внутренний протокол граничного шлюза

OSPF – Open Shortest Path First/Протокол первого кратчайшего пути

BW- Bandwidth/Пропускная способность

CNI – Container networking interface/Сетевой интерфейс контейнера

K8s – Kubernetes

ARIMA - Autoregressive Integrated Moving Average/Авторегрессионная интегрированная модель скользящего среднего

ACF – Autocorrelation function/Автокорреляционная функция

PACF – Partial autocorrelation function/Частичная автокорреляционная функция

Введение

В современном мире всё большую роль играют сервисы, их количество постоянно растет, а в связи с использованием микросервисной архитектуры влияние на увеличение масштаба сетей колоссально. Увеличение количества микросервисов, использующихся в центрах обработки данных (ЦОД), обусловливает существенное повышение необходимости к быстрому и качественному решению задач сетевого управления в условиях постоянно растущей нагрузки. Данный рост приводит к необходимости постоянного масштабирования данных сетей, заключающимся в увеличении количества серверов и маршрутизаторов. Существует потребность в простых и эффективных масштабируемых решениях для облегчения выполнения задач автоматизации, маршрутизации и управления постоянно растущими сетями.

Для реализации сервисов, которыми пользуются миллионы людей, используются такие технологии, как Docker и Kubernetes, в локальных или территориально распределённых вычислительных сетях с использованием технологии BGP для обеспечения маршрутизации между кластерами и контейнерами. Однако, BGP с стандартным набором параметров зачастую не обеспечивает наибольшую сходимость и надежность сети.

Цель данной работы – разработать и внедрить метод автоматизации сетевой доступности в контейнеризированную сеть организации.

Цель проекта определяет список задач работы:

Планирование и описание сетевой инфраструктуры организации (стенда).

Рассмотрение и анализ факторов, влияющих на эффективность BGP в описанной ЛВС.

Разработка и реализация тестового стенда

Разработка адаптивного алгоритма.

Использование программных средств для анализа трафика, позволяющих сделать качественную оценку внедренного решения. (снятие статистической картины с использованием инструментов, не влияющих на утилизацию канала связи).

Имплементация предложенного решения. (интеграция адаптивного алгоритма в сетевой стек).

Объектом исследования является протокол граничного шлюза и принцип работы модулей для обеспечения взаимодействия между контейнерами и кластерами, а в частности протокол BGP в реализации CNI Calico.

Предметом исследования является методика и разработка адаптивного алгоритма маршрутизации для обеспечения наибольшей сходимости протокола BGP в контейнеризированной сети с использованием k8s и CNI Calico.

Значимость работы

Разработка алгоритма направлена на обеспечение более качественной сетевой доступности в кластеризированных сетях, подобный подход позволит более эффективно использовать доступную маршрутную информацию, а в следствии, повысит качество предоставляемых сервисов.

Использование алгоритма, взаимодействующее с доступными инструментами оркестровки, а не модифицируя данные инструменты, предоставляет выбор конечному пользователю между двумя, не влияя на базовый функционал оригинального программного обеспечения.

Вывод

Таким образом, разработка алгоритма по обеспечению автоматизации сетевой доступности является актуальной задачей, направленной на обеспечение доступности узлов и уменьшения задержек при переключении между каналами связи, а также предсказания трафика с целью моментального реагирования на аномалии.

Литературный обзор

1.1 Исследование различных протоколов динамической маршрутизации

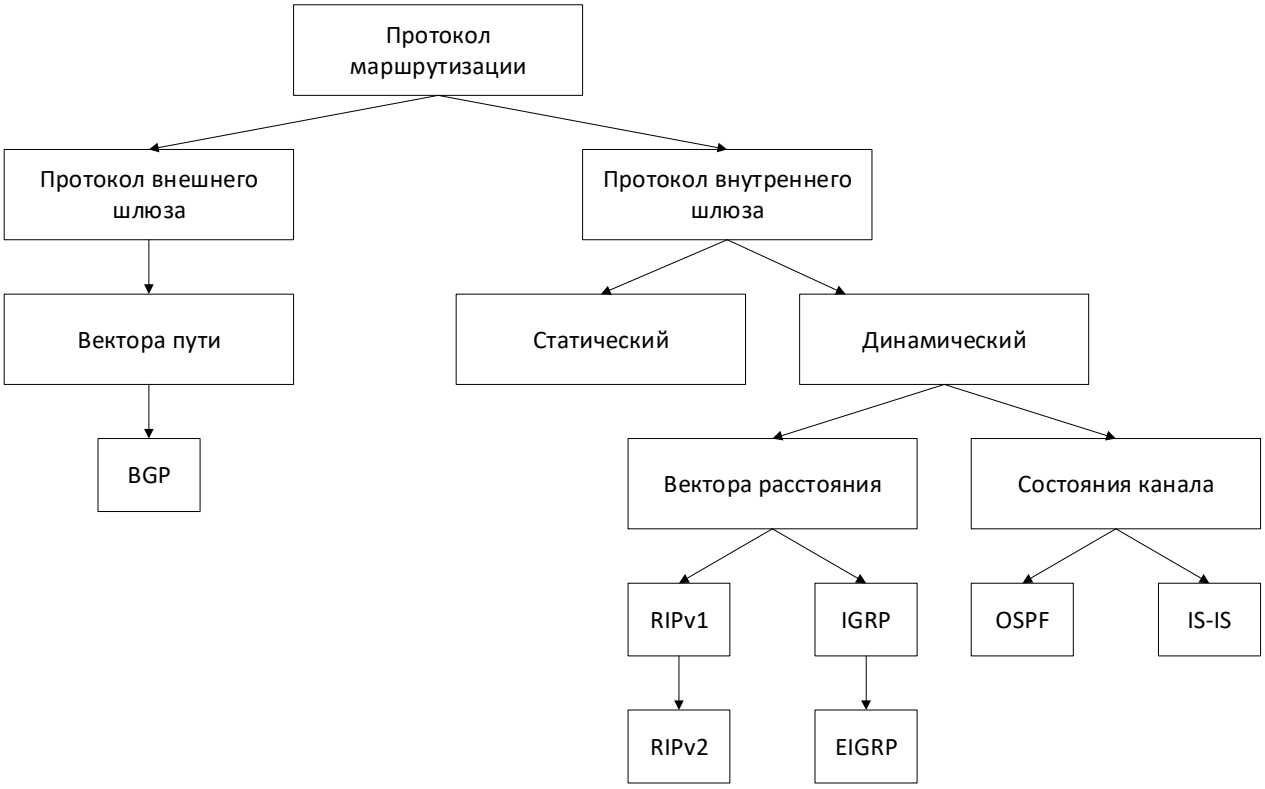

Рисунок 1 - Протоколы маршрутизации

Протоколы маршрутизации делятся на две основные категории: **внутренние (IGP)**, такие как OSPF и RIP, которые используются для обмена маршрутами внутри одной автономной системы (AS), и **внешние (EGP)**, например BGP, управляющие маршрутизацией между разными AS. Основное различие между ними заключается в масштабе и доверии: IGP оптимизирован для быстрой сходимости внутри сети, а EGP фокусируется на политиках и стабильности межсетевых соединений. Среди внутренних протоколов выделяются **векторно-дистанционные** (например, RIP), где маршрутизаторы обмениваются таблицами с информацией о количестве переходов до узлов, и **состояния канала** (например, OSPF), где каждый узел строит полную топологию сети, используя алгоритм Дейкстры. Векторно-дистанционные протоколы проще, но медленнее реагируют на изменения и склонны к петлям,

тогда как протоколы состояния канала обеспечивают высокую скорость сходимости и оптимальность маршрутов за счёт больших вычислительных ресурсов. BGP, как гибридный **протокол вектора пути**, сочетает элементы обоих подходов, учитывая пути через AS для предотвращения петель и реализации сложных политик.

Протоколы динамической маршрутизации семейства IGP использующее алгоритмы поиска кратчайшего пути, в основе работы которых лежат алгоритмы Дейкстры и Беллмана-Форда не применяются в больших масштабируемых сетях из-за:

-Большого количества потребляемых ресурсов узла, в случае алгоритма Дейкстры.

-Невозможности использования циклов отрицательной метрики в сетях, где более 15 переходов, в случае алгоритма Беллмана Форда [1].

Таким образом, одним из наиболее популярных не проприетарных протоколов динамической маршрутизации, применяемых в масштабируемых сетях, является протокол граничного шлюза BGP.

1.2 Исследование протокола граничного шлюза BGP

BGP – border gateway protocol (протокол граничного шлюза) – протокол динамической маршрутизации автономных систем (autonomous systems – AS). Основной функцией протокола является обмен информации о доступности сетей с другими узлами, использующими BGP. Эта информация доступности позволяет составить список автономных систем, через которые она проходит [2].

Под «**автономной системой**» (АС) традиционно понимается условная «зона ответственности» оператора связи с принадлежащими ему маршрутизаторами, находящимися под единым административным управлением и использующими единый согласованный план внутренней маршрутизации, а также согласованную картину адресатов, доступных через данную АС. Имеет большое количество параметров, обеспечивающих быструю и эффективную настройку сетевой политики.

1.2.1 Применение в различных областях

Востребованность обусловлена способностью эффективно масштабироваться и адаптироваться к динамичным средам, что критически важно для выполнения задач высокой доступности, балансировки нагрузки и управления трафиком [3].

Протокол используется в маршрутизации сети интернет, а также распространен в ЦОД и сетях с использованием средств кластеризации [4] [5].

BGP крайне востребован в обеспечении взаимодействия территориально распределенных приложений, развёрнутых на базе контейнерных сред под управлением инструментов оркестровки.

1.2.2 Принцип работы

Процесс определения наилучшего маршрута в BGP осуществляется последовательно через итеративное сравнение доступных путей по строго заданным критериям [2]. Алгоритм работает по принципу приоритетного выбора: первый путь, удовлетворяющий условиям, объявляется оптимальным, а

последующие варианты игнорируются. Основные этапы сравнения, описанные в стандарте RFC и следующие в таком порядке:

1.Проверка достижимости следующего прыжка.

Если следующий прыжок недостижим, маршрут игнорируется.

2.Наибольший локальный приоритет

Маршрут с наибольшим значением Local Preference.

3.Наименьший AS_PATH

Предпочтение маршрута с наименьшей длиной AS_PATH.

4.Lowest Origin Type

Приоритет типов origin в порядке: IGP (0)> EGP (1)> Incomplete (2).

5.Наименьший MED(Multi-exit-descriminator)

Маршрут с наименьшим значением MED (только если оба маршрута пришли от одного соседнего AS).

6.Приоритет EGBP над IBGP

EBGP-маршруты выбираются перед IBGP-маршрутами.

7.Наименьшая метрика IGP до следующего прыжка

Маршрут с наименьшей метрикой IGP (например, OSPF, EIGRP) до Next Hop.

8.Самый старый маршрут (только для EBGP)

Если маршруты пришли через EBGP, выбирается самый «старый» (устойчивый) маршрут.

9.Наименьший идентификатор соседа

Маршрут от BGP-соседа с наименьшим Router ID.

10.Shortest Cluster List (для отраженных маршрутов)

Если используется Route Reflection (отражение маршрутов), предпочитается маршрут с наименьшей длиной Cluster List.

11.Наименьший адрес соседа

Если все предыдущие шаги равны, выбирается маршрут от соседа с меньшим IP-адресом.

1.2.3 Недостатки протокола

Несмотря на широкое применение, BGP обладает рядом фундаментальных ограничений, которые осложняют его использование в динамичных и высоконагруженных средах:

1. **Неустойчивость маршрутов (Route Flapping)**
   BGP подвержен частым изменениям маршрутов из-за нестабильности каналов связи или ошибок конфигурации. Это явление, называемое «флаппингом», приводит к постоянному перерасчету таблиц маршрутизации и генерации избыточного трафика обновлений. Например, кратковременный сбой канала между автономными системами может вызвать каскадное распространение UPDATE-сообщений по всей сети, стандарт RFC2439 стремится решить эту проблему [6].

В статье Минаковой Н.Н. и Мансурова А.В. представлено решение для обнаружения и блокировки аномальной маршрутной информации в протоколе BGP-4, направленное на повышение безопасности межсетевого взаимодействия. Авторы акцентируют внимание на проблеме распространения некорректных маршрутов, вызванных утечками, угонами или ошибочными конфигурациями, которые могут дестабилизировать работу сети. Предложенный подход не требует модификации ПО маршрутизаторов, что упрощает его внедрение без прерывания сервисов оператора связи.

Основу решения составляет модульная архитектура, включающая:

- **Сбор и обработку обновлений BGP-4** через подключение к маршрутизатору контролируемой автономной системы (АС).
- **Анализ маршрутной информации** с использованием данных из баз RADb и IRR для проверки соответствия заявленным политикам связности. Алгоритм включает проверку атрибутов AS_PATH, NLRI и BGP communities, а также верификацию легитимности маршрутов через запросы к объектам ‘route’ и ‘aut-num’.

- **Блокировку аномалий** через автоматическое обновление конфигурации пограничных маршрутизаторов, используя их командные интерфейсы.

Решение реализовано на Python с применением BGP-агента YABGP и документоориентированной СУБД MongoDB для кеширования данных. Эксперименты подтвердили его эффективность в выявлении некорректных маршрутов, включая предотвращение распространения приватных сетей и маршрутов с нарушенными политиками.

2. **Медленная конвергенция**
   BGP характеризуется длительным временем конвергенции, обусловленное:
   - Использованием TCP для надежной доставки сообщений.
   - Последовательным применением 11-шагового алгоритма выбора лучшего пути, а в случае некоторых вендоров, 12 или 13. [2] [7]
   - Отсутствием механизмов мгновенного оповещения всех участников AS об изменениях топологии (в отличие от IGP, например, OSPF).
3. **Зависимость от префиксной гранулярности**
   BGP оперирует префиксами IP-сетей, а не отдельными хостами или каналами. В результате сбой одного физического интерфейса может вызвать массовое обновление тысяч префиксов. Например, отказ магистрального канала между ЦОД Amazon в 2021 г. привел к перерасчету 150 тыс. маршрутов, вызвав частичную недоступность AWS [8].

   Следует отметить еще одно ограничение:
   - Сложность конфигурации: Требуется ручная настройка фильтров, атрибутов и политик, что повышает риск ошибок (около 34% инцидентов связаны с человеческим фактором) [9].

1.2.4 Перспективы развития

Перечисленные недостатки и большая востребованность мотивируют создавать решения, стремящиеся уменьшить участие человека в создании конфигурационных параметров, а также повысить надежность при использовании протокола, например – нейро-нечёткие сети с применением алгоритмов машинного обучения.

Например, В статье А. М. Асланова и М. С. Солодовника [10] исследуется применение нейро-нечетких сетей (ННС) для решения задач адаптивной маршрутизации в условиях динамически изменяющихся компьютерных сетей. Авторы подчеркивают ограничения традиционных методов, таких как матричный алгоритм поиска кратчайших путей, который требует частого пересчета матриц задержек, что приводит к высоким вычислительным затратам и неспособности оперативно реагировать на изменения параметров сети (нагрузка, задержки, топология).

В качестве альтернативы предлагается гибридный подход, объединяющий нечеткую логику и нейронные сети. Нейро-нечеткие сети позволяют:

- Учитывать экспертные знания через фазификацию входных параметров, что повышает гибкость алгоритма.
- Самообучаться на основе данных о задержках и топологии сети, адаптируясь к нелинейным зависимостям.
- Прогнозировать задержки с помощью экстраполяции, что подтверждено экспериментами в среде MatLab. Например, прогноз задержки на 9-й день работы сети на основе предыдущих данных совпал с реальным значением, что демонстрирует высокую точность модели.

Ключевой вывод статьи: применение ННС позволяет маршрутизаторам динамически оптимизировать пути передачи данных, минимизируя задержки даже в условиях нестабильности сети. Это обеспечивает снижение зависимости от ручных настроек, повышает скорость

принятия решений и надежность сети, что делает нейро-нечеткие сети перспективным инструментом для интеллектуальных систем маршрутизации.

В статье Гребенникова А.В., Крюкова Ю.А. и Чернягина Д.В. [11] исследуется применение моделей ARIMA для прогнозирования сетевого трафика с целью оптимизации управления нагрузкой в условиях динамически изменяющихся сетевых условий. Авторы акцентируют внимание на проблеме пиковых нагрузок, ведущих к снижению производительности сетей, и подчеркивают необходимость точных краткосрочных прогнозов для предотвращения перегрузок. В рамках исследования был разработан алгоритм подбора ARIMA-модели с минимальным числом параметров, включающий этапы стационаризации временного ряда, идентификации порядка модели (p, d, q) через анализ автокорреляционных функций, оценки параметров методом максимального правдоподобия и проверки адекватности с использованием Q-статистики и информационного критерия Шварца (SBIC).

Эксперименты, проведенные на реальных данных трафика, показали, что модель ARIMA(4,1,3) демонстрирует наилучшую точность (MAPE < 20%) при прогнозировании на 2 шага вперед, в то время как для более длинных горизонтов (до 10 шагов) эффективной оказалась модель ARIMA(1,1,0). Авторы также подтвердили, что увеличение прогнозного горизонта снижает точность, что согласуется с природой нестационарности сетевого трафика. Результаты работы подчеркивают практическую значимость ARIMA-моделей с оптимизированной параметризацией для задач динамического управления сетевыми ресурсами, включая предотвращение «узких мест» и распределение пропускной способности.

Данное исследование дополняет существующие подходы к прогнозированию трафика, предлагая методологию выбора моделей с балансом между точностью и вычислительной сложностью, что актуально для разработки систем мониторинга и адаптивного управления в современных сетях передачи данных.

1.3 Исследование Kubernetes и средств обеспечения сетевой доступности в нём

В связи с особенностями микросервисной архитектуры, где каждый узел в сети выполняет одну или несколько схожих задач, возникает необходимость в применении средств контейнеризации и оркестровки этих контейнеров. Kubernetes — платформа с открытым исходным кодом, созданная для автоматизации управления контейнеризированными приложениями. Первоначально разработанная компанией Google, поддерживается фондом Cloud Native Computing Foundation (CNCF) и является отраслевым стандартом для оркестрации контейнеров. Ее ключевая задача — упрощение развертывания, масштабирования и поддержки распределенных приложений в кластерных средах [12].

1.3.1 Выбор модуля сетевой доступности

Существует множество решений для обеспечения сетевой доступности в Kubernetes. В статье “ СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ СЕТЕВЫХ ПЛАГИНОВ ОРКЕСТРАТОРА KUBERNETES” [13] автором произведен статистический анализ влияния наиболее популярных CNI (Calico, Flannel, Cilium, Canal) на производительность системы, тестирование проходило по ряду протоколов TCP, UDP, HTTP, FTP и SCP в 10 Гбитной сети. За параметры, между которыми производилось сравнение, взяты:

1. Производительность в Мбит/с при использовании вышеуказанных протоколов
2. Потребление RAM
3. Потребление CPU

Как эталон взято базовое оборудование, без использования специализированных программных решений.

По результатам анализа наибольшую производительность показали Flannel и Calico. Автор предполагает, что совместимость Flannel связана с его компактностью и совместимостью с большим количеством архитектур, а также возможность автоматического определения MTU.

В случае Calico высокая производительность связана с тем, что модуль пропускает только установленный администратором трафик. Неоспоримым преимуществом Calico является безопасность в связи с вышеуказанной особенностью.

Flannel не подходит для целей данной работы, так как он не использует протоколы динамической маршрутизации для обеспечения сетевой доступности.

Calico, который использует BGP для обеспечения взаимодействий между подами выбран в роли CNI.

1.3.2 Модуль(плагин) сетевого интерфейса Calico

Calico — сетевое решение для Kubernetes, обеспечивающее безопасное взаимодействие контейнерных приложений и управление политиками доступа. В отличие от оверлейных сетей, Calico использует **подход уровня 3**, основанный на IP-маршрутизации, что минимизирует сложность инфраструктуры и снижает производительные накладные расходы. Использует BGP для распределения маршрутов.

1.4 Вывод

Предлагается разработка алгоритма для выбора более эффективного маршрута путем динамического варьирования конфигурации протокола BGP по результатам периодического статического среза пропускной способности узла на основании предсказания модели ARIMA. Таким образом, при обнаружении аномалии в трафике, скрипт отправит сигнал об изменении конфигурации, и трафик будет направляться через альтернативный маршрут.

2. Специальный раздел

Экспериментальный стенд спроектирован с применением средств виртуализации, для исследования устойчивости территориально распределенного кластера **Kubernetes** в условиях деградации каналов связи. Архитектура (рис. 2) реализует сценарии отказоустойчивости, где динамическое перераспределение трафика между географически удаленными узлами обеспечивается за счет интеграции **BGP** (Border Gateway Protocol) и **CNI Calico**.

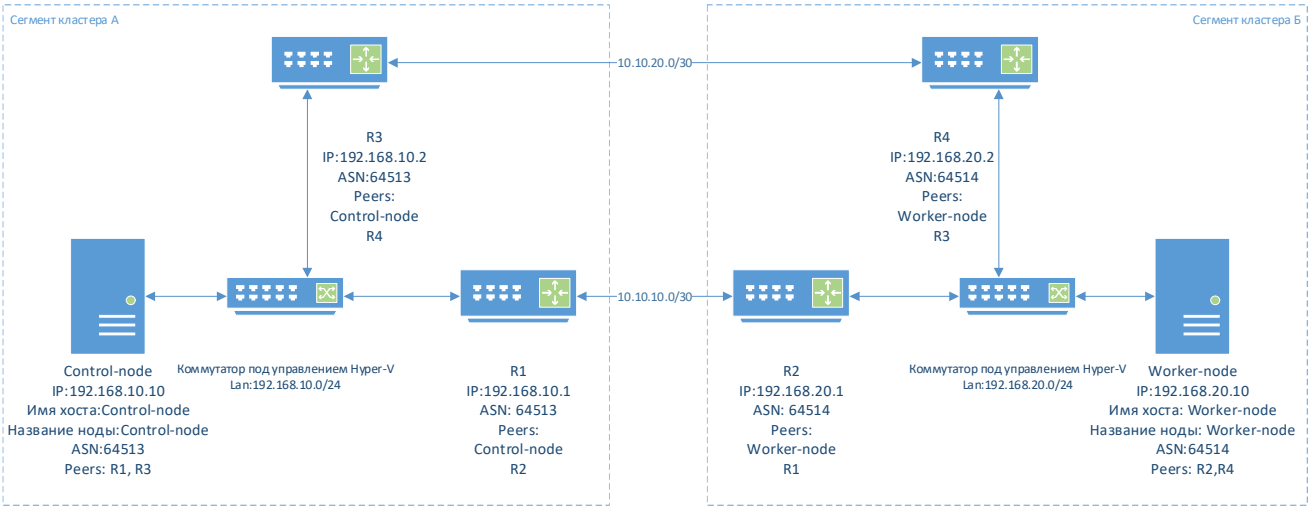

Рисунок 2 - Архитектура сети

На рисунке изображен макет физических рабочих мест на виртуальных образах с применением технологии виртуализации Hyper-V. Перечень виртуальных образов предоставлен ниже:

- Control-node – Узел с ОС Ubuntu 24.10 LTS
- Worker-node – Узел с ОС Ubuntu 24.10 LTS
- R1-R4 – маршрутизаторы с ОС OPNSense

2.1 Цели и функциональные требования

1. **Имитация распределенного кластера Kubernetes**:
    - Control-node — хост с компонентами управления кластером (kube-apiserver, etcd, kube-controller-manager, kube-scheduler).

- Worker-node — узел для развертывания рабочих нагрузок (kubelet, kube-proxy), принимающий деплойменты из системы оркестрации.
- Использование CNI Calico обеспечивает сетевую политику, распределение IP-адресов и интеграцию с BGP для объявления маршрутов между узлами.

2. **Моделирование сбоев каналов связи**:
   - Сегменты **10.10.10.0/30** (R1–R2) и **10.10.20.0/30** (R3–R4) эмулируют каналы двух независимых ISP с возможностью искусственного введения задержек.
   - Деградация канала связи эмулируется путем замедления виртуальной сети средствами гипервизора Hyper-V
3. **Динамическая адаптация маршрутов**:
   - При обнаружении деградации канала алгоритм автоматически корректирует BGP-конфигурацию на хостах Ubuntu, перенаправляя трафик через альтернативный маршрутизатор (например, с R1–R2 на R3–R4).
   - **OPNsense** на маршрутизаторах R1–R4 обеспечивает транзитную маршрутизацию между автономными системами **AS64513** (Control-node, R1, R3) и **AS64514** (Worker-node, R2, R4).

Архитектурные компоненты

1. **Сетевой уровень**:
   - **Транспортные каналы ISP**:
     - Сегмент **10.10.10.0/30** (AS64513 ↔ AS64514) — основной канал, подвергаемый нагрузочному тестированию.
     - Сегмент **10.10.20.0/30** — резервный канал, активируемый при деградации основного.
   - **Внутрикластерная связь**:

- Подсети **192.168.10.0/24** (Control Plane) и **192.168.20.0/24** (Data Plane) изолированы для минимизации коллизий управляющего и пользовательского трафика.

2. **Уровень оркестрации**:
   - **Kubernetes Control Plane**: Развернут на Control-node, координирует состояние кластера, распределяет поды на Worker-node.
   - **Calico**: Каждый узел кластера (Control-node, Worker-node) выступает как BGP-пир, анонсируя свои подсети через маршрутизаторы R1–R4. Это позволяет динамически обновлять таблицы маршрутизации при изменении топологии.

Схема уровня оркестрации предоставлена на рисунке 3.

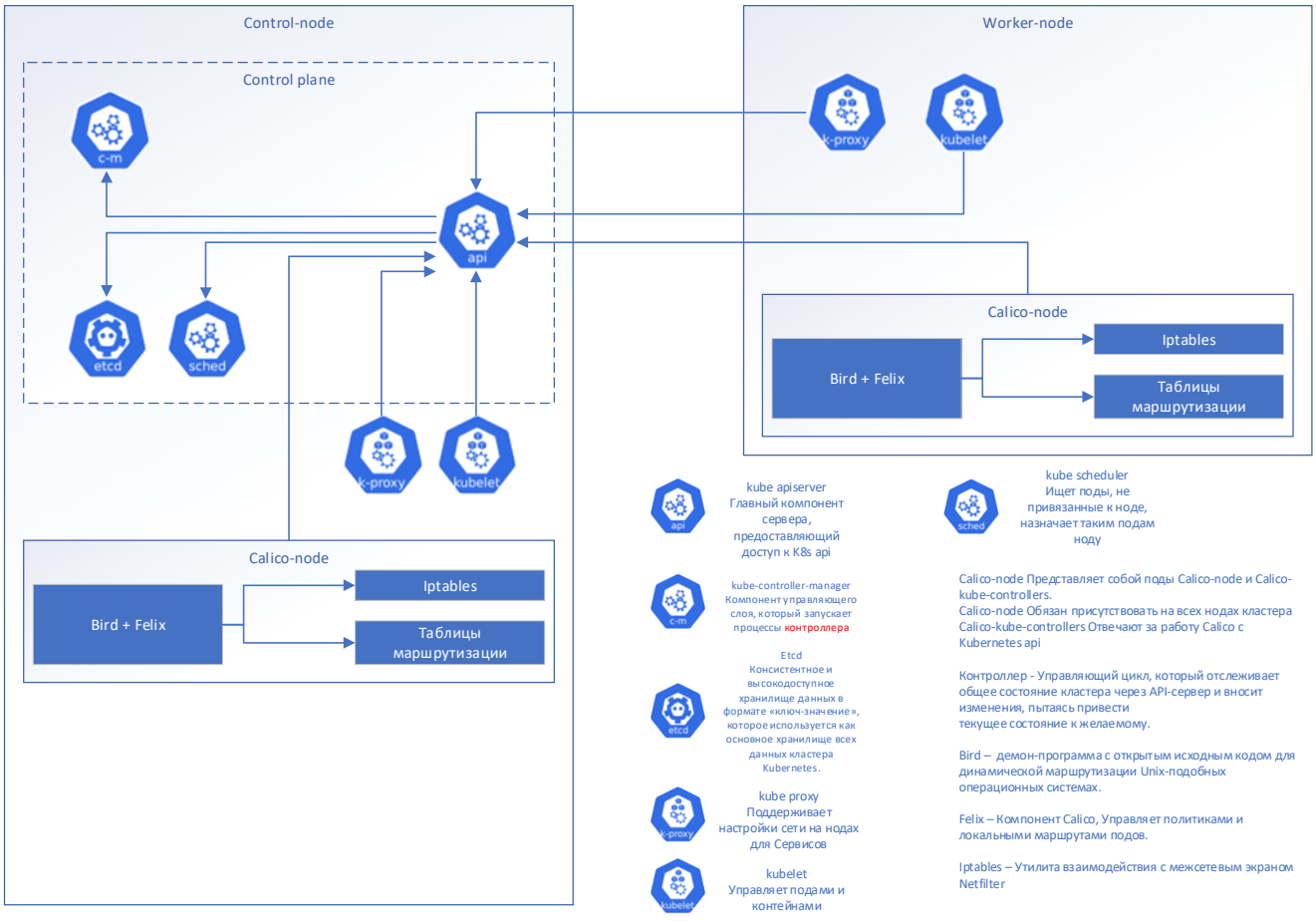

Рисунок 3 – Схема оркестрации

На схеме показаны основные взаимодействия и компоненты на уровне оркестрации в Kubernetes.

Основные компоненты:

- kube apiserver - Главный компонент сервера, предоставляющий доступ к Kubernetes api для обеспечения взаимодействия компонентов кластера и управления
- kube-controller-manager - Компонент управляющего слоя, который запускает процессы контроллера
- etcd – Консистентное и высокодоступное хранилище данных в формате «ключ-значение», которое используется как основное хранилище всех данных кластера Kubernetes
- kube proxy - Поддерживает настройки сети на нодах для Сервисов
- kubelet - Управляет подами и контейнерами

Calico-node Представляет собой поды Calico-node и Calico-kube-controllers.

- Calico-node Обязан присутствовать на всех нодах кластера и отвечает за получение и распространение маршрутной информации.
- Calico-kube-controllers Отвечают за работу Calico с Kubernetes api

Ниже предоставлены таблицы IP адресов узлов и BGP отношений.

Таблица 1 – IP адреса узлов

Имя узла	Интерфейс	Ip адрес	Маска
Control-node	Eth0	192.168.10.10	/24
	Control-IPpool	10.244.0.0	/24
Worker-node	Eth0	192.168.20.10	/24
	Worker-IPpool	10.244.1.0	/24
R1	Eth0	192.168.10.1	/24
	Eth1	10.10.10.1	/30
R2	Eth0	192.168.20.1	/24
	Eth1	10.10.10.2	/30
R3	Eth0	192.168.10.2	/24
	Eth1	10.10.20.1	/30

R4	Eth0	192.168.20.2	/24
	Eth1	10.10.20.2	/30

Таблица 2 – BGPpeers

Узел	BGPpeer	ASN
Control-node	R1	64513
	R3	64513
Worker-node	R2	64514
	R4	64514
R1	Control-Node	64513
	R2	64514
R2	Worker-node	64514
	R1	64513
R3	Control-Node	64513
	R4	64514
R4	Worker-Node	64514
	R3	64513

2.2 Проектирование алгоритма

2.2.1 Цели и задачи

Алгоритм разработан для решения проблемы динамической оптимизации BGP-маршрутизации в программно-конфигурируемых сетях на основе прогнозирования трафика.

Ключевые задачи:

1. **Непрерывный мониторинг** сетевой нагрузки с интервалом 5 секунд
2. **Прогнозирование трафика** на горизонте 5-300 секунд
3. **Автоматическая корректировка** BGP-пиринговых отношений при критических отклонениях
4. **Сохранение целостности** сетевой конфигурации при изменениях

2.2.2 Функциональные требования

Система должна обеспечивать:

- Детерминированную реакцию на аномалии трафика ($\Delta t < 120$ сек)
- Гарантированную работоспособность при потере до 40% входных данных
- Логирование всех состояний для пост-анализа

2.2.3 Применение модели ARIMA в алгоритме

Модель ARIMA (Autoregressive Integrated Moving Average) используется в алгоритме для краткосрочного прогнозирования сетевого трафика, что является ключевым элементом динамической оптимизации BGP-маршрутизации. Принцип работы ARIMA основан на комбинации трех компонентов:

1. **Авторегрессии (AR)**, учитывающей линейную зависимость текущего значения трафика от предыдущих наблюдений;
2. **Интегрирования (I)**, обеспечивающего стационарность ряда через дифференцирование (устранение трендов и сезонности);
3. **Скользящего среднего (MA)**, моделирующего влияние прошлых ошибок прогноза.

Параметры модели подбираются на основе анализа автокорреляционных функций (ACF/PACF) и тестов на стационарность. Это позволяет адаптировать модель к нестационарным данным сетевой нагрузки, характерным для

программно-конфигурируемых сетей. Прогнозы, генерируемые ARIMA, служат основой для принятия решений о корректировке BGP-пиринговых отношений: при отклонении прогнозируемых значений от заданных порогов система инициирует перераспределение маршрутов, минимизируя задержки и потери пакетов. Использование ARIMA обеспечивает баланс между точностью прогнозирования и вычислительной эффективностью, что критично для работы алгоритма в режиме реального времени.

2.3 Вывод

На основе требований и описанных технологий построен макет, имитирующий распределённый кластер Kubernetes, разработаны и апробированы скрипты автоматизации установки и развертывания кластера и его компонентов. Текущая реализация позволяет имитировать критические сценарии (потерю связи, задержки) и фиксировать изменения в таблицах маршрутизации, что необходимо для валидации будущего алгоритма в условиях, приближенных к реальным.

3. Технологический раздел

По описанным в предыдущем разделе требованиям, разработан алгоритм предсказания трафика с целью обнаружений аномалий и реактивного изменения конфигурации BGP с целью выбора нового маршрута, апробируемый на макете, развернутом ранее. Компонентная и логическая схема предоставлена на рисунках 4 и 5 соответственно.

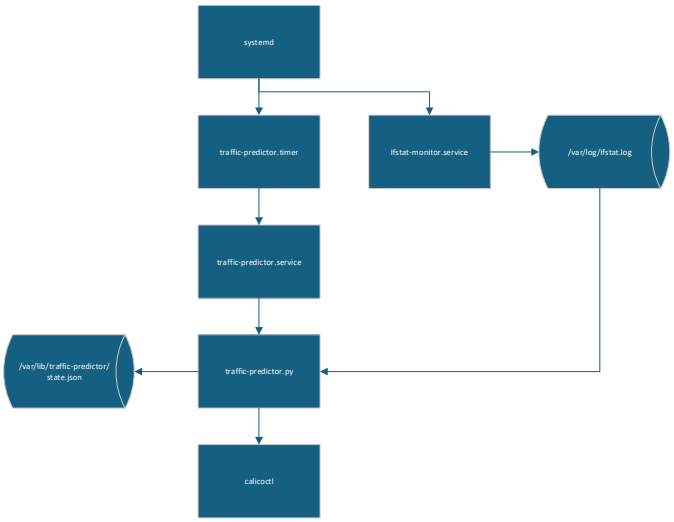

Рисунок 4 – Блок-схема компонентов

Алгоритм представляет собой скрипт, написанный на языке программирования Python. За запуск скрипта отвечает подсистема инициализации и управления службами в Linux systemd.

За запуск скрипта отвечает сервис traffic-predictor.service, который запускается каждые 5 минут через таймер traffic-predictor.timer.

За сбор информации отвечает сервис ifstat-monitor.service, который запускает утилиту ifstat для мониторинга и сбора утилизации пропускной

способности интерфейса хоста. Данная информация записывается в файл лога ifstat.log с интервалом 5 секунд.

Сервис ifstat-monitor.service и таймер traffic-predictor.timer находятся в состоянии enabled, т.е запускаются при загрузке операционной системы.

При запуске oneshot(единичный запуск) сервиса traffic-predictor.service компилируется и исполняется скрипт traffic-predictor.py, который считывает и обрабатывает данные из лог файла ifstat.log, производит предсказание с применением модели ARIMA и на основе полученного результата делается вывод о необходимости изменения конфигурации BGP в CNI Calico.

Если необходимо изменить конфигурацию, скрипт отправляет команду в Kubernetes api с новой конфигурацией.

Если изменения не требуются, то конфигурация не изменяется и продолжается штатная работа системы.

На каждом этапе работы скрипта предусмотрено логирование.

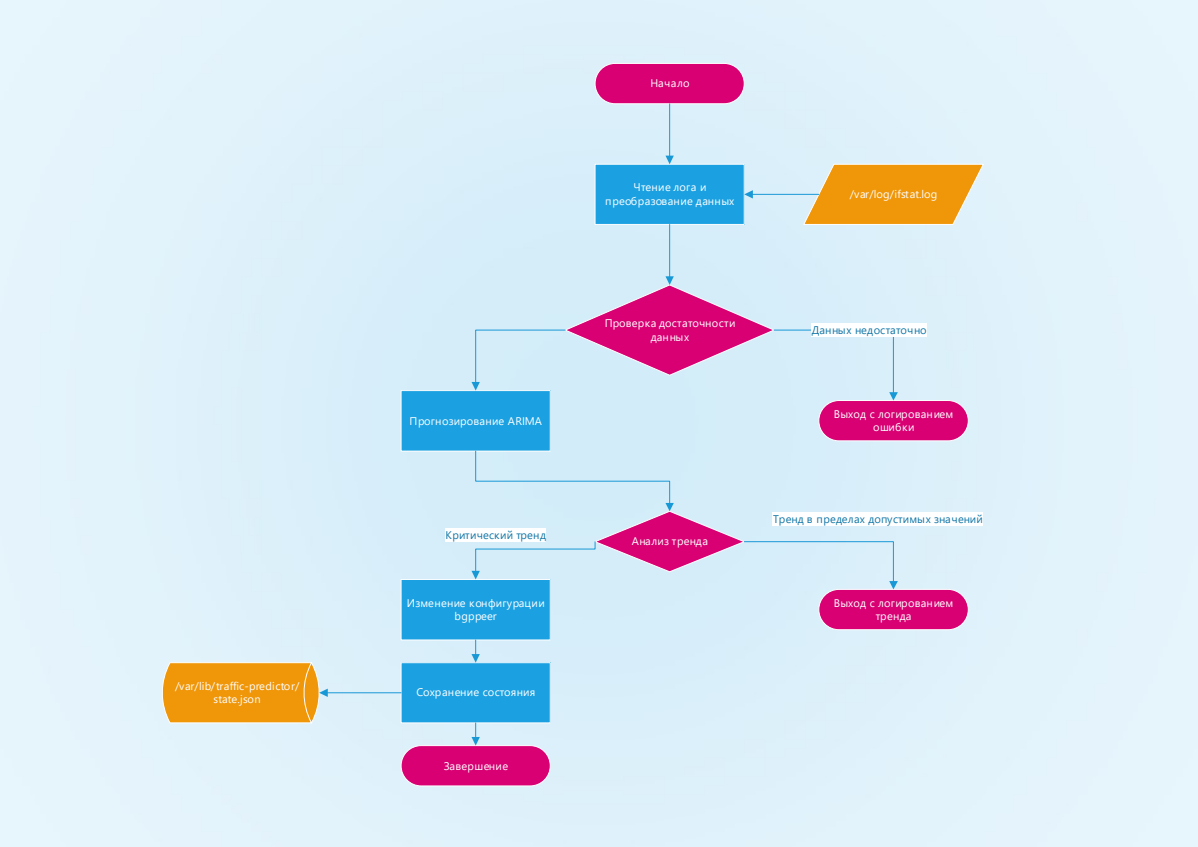

Рисунок 5 – Блок-схема логики скрипта

Для апробации скрипта был запущен трафик между хостами control-node и worker-node путем использования утилиты ping с весом 10000кбит.

Модель успешно предсказывала трафик, после резкого изменения тренда (прекращение передачи ping пакетов) была замечена аномалия и изменилась конфигурация, в следствие чего был использован резервный маршрут.

При резком замедлении канала связи (путем уменьшения пропускной способности интерфейсов маршрутизаторов R1-R2), скрипт обнаружил это и также произвел переключение.

Заключение

В ходе проведённого исследования были проанилизированы текущие решения в области обеспечения сетевой доступности в сетях с применением средств контейнеризации и оркестрации. Развернут стенд, имитирующий подобную сеть и разработан алгоритм, предсказывающий трафик и производящий изменение конфигурации на основе сделанного предсказания.

Список использованных источников

[1] Д. А. Барыбин, *Сравнение алгоритмов Дейкстры и Беллмана-Форда при решении задачи о поиске кратчайшего пути в протоколах маршрутизации,* Д. А. Барыбин, Е. Ю. Кофман, М. С. Шульгин // Символ науки: международный научный журнал. – 2021. – № 6. – С. 27-31. – EDN UIGLRD..

[2] RFC, Border Gateway Protocol 4 (BGP-4).

[3] D. D.G., BGP in the Data Center, O'Relly Media Publ., 2017.

[4] Abhashkumar A., Subramanian K., Andreyev A., Running BGP in data centers at scale., Proc. XVIII USENIX Symposium on NSDI, 2021.

[5] RFC, Use of BGP for Routing in Large-Scale Data Centers.

[6] RFC, BGP Route Flap Damping.

[7] Cisco, BGP Best Path Selection Algorithm, 2023.

[8] AWS, «Сводка перебоя,» декабрь 7 & 10, 2021.

[9] Verizon, «DIBR,» 2022.

[10] А. М. Асланов, М. С. Солодовник, «ИССЛЕДОВАНИЕ ИНТЕЛЛЕКТУАЛЬНОГО ПОДХОДА В МАРШРУТИЗАЦИИ КОМПЬЮТЕРНЫХ СЕТЕЙ,» *Электротехнические и компьютерные системы №16(92),* pp. 93-100, 2014.

[11] «МОДЕЛИРОВАНИЕ СЕТЕВОГО ТРАФИКА И ПРОГНОЗИРОВАНИЕ С ПОМОЩЬЮ МОДЕЛИ ARIMA,» *Системный анализ в науке и образовании,* 2011.

[12] Kubernetes, Официальная документация Kubernetes.

[13] Б. М. Н., «Сравнительный анализ производительности сетевых плагинов оркестратора Kubernetes,» *Молодой ученый,* № 44(543), pp. 4-12, 2024.