ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Санкт-Петербургский государственный университет телекоммуникаций
им. проф. М.А. Бонч-Бруевича»

На правах рукописи

МУХИЗИ САМУЭЛЬ

# РАЗРАБОТКА МОДЕЛЕЙ И МЕТОДОВ СЕГМЕНТАЦИИ РЕСУРСОВ В ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЯХ

Специальность 05.12.13 – Системы, сети и устройства телекоммуникаций

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель
доктор технических наук, доцент
Киричек Руслан Валентинович

Санкт-Петербург – 2019

ОГЛАВЛЕНИЕ

**ВВЕДЕНИЕ..... 5**

**Глава 1. Анализ концепции программно-конфигурируемых сетей в сетях связи пятого поколения 5G/IMT-2020..... 11**

1.1. Перспективы развития сетей связи 5G/IMT-2020..... 11

1.2. Необходимость программирования сетей связи 5G/IMT-2020 ..... 12

1.3. Виртуализация сетей в сетях связи 5G/IMT-2020..... 13

1.4. Анализ концепции программно-конфигурируемых сетей и протокола OpenFlow ..... 15

1.4.1. Развитие программно-конфигурируемых сетей..... 15

1.4.2. Архитектурная концепция и принципы построения программно-конфигурируемых сетей ..... 18

1.4.3. Архитектура программно-конфигурируемых сетей ..... 19

1.4.4. Контроллер программно-конфигурируемой сети ..... 21

1.4.5. Протокол OpenFlow..... 23

1.4.6. Протокол OF-CONFIG ..... 25

1.4.7. Протоколы NB-API ..... 27

1.4.8. Порты OpenFlow..... 28

1.4.9. Канал OpenFlow..... 29

1.5. Виртуализация сетевых функций ..... 30

1.6. Концепция сетевой сегментации в сетях связи..... 33

1.6.1. Аналитический обзор международной деятельности по исследованию сетевой сегментации ..... 36

1.6.2. Деятельность по разработке стандартов по сетевой сегментации ..... 36

Выводы по главе 1..... 39

**Глава 2. Разработка моделей оценки показателей эффективности функционирования программно-конфигурируемых сетей..... 40**

2.1. Коммутатор OpenFlow..... 40

2.2. Принципы функционирования программно-конфигурируемой сети..... 40

2.3. Оценка производительности ПКС-контроллеров..... 41

2.4. Тестирование контроллера программно-конфигурируемой сети ..... 42

на базе разработанной методики испытаний..... 42

2.4.1. Цель эксперимента..... 42

2.4.2. Задачи эксперимента..... 42

2.4.3. Структура лабораторного стенда и модельной сети..... 42

2.5. Алгоритмы тестирования контроллера ПКС..... 44

2.5.1. Алгоритм тестирования ПКС-контроллера утилитой Cbench..... 44

2.5.2. Алгоритм тестирования ПКС-контроллера программным обеспечением ProLan Qutester Plus..... 44

2.5.3. Алгоритм тестирования ПКС-контроллера программным обеспечением OCCT ..... 44

2.5.4. Алгоритм тестирования ПКС-контроллера программным обеспечением Mininet ..... 45

2.6. Результаты тестирования..... 45

2.6.1. Результаты тестирования ПКС-контроллера утилитой Cbench..... 45

2.6.2. Результаты тестирования ПКС-контроллера программным обеспечением ProLan QuTester Plus ..... 47
2.6.3. Результаты тестирования ПКС-контроллера программным обеспечением OCCT ..... 48
2.6.4. Результаты тестирования ПКС-контроллера программным обеспечением Mininet ..... 51
2.6.5. Выводы по результатам тестирования ..... 55
2.7. Разработка моделей ПКС для исследования эффективности функционирования ..... 57
2.7.1. Общее представление системы ..... 57
2.7.2. Имитационная модель программно-конфигурируемой сети ..... 58
2.7.3. Аналитический модель программно-конфигурируемых сетей ..... 59
2.7.4. Результаты моделирования ..... 63
Выводы по главе 2 ..... 65
**Глава 3. Метод кластеризации ресурсов программно-конфигурируемых сетей для динамического распределения контроллеров ..... 67**
3.1. Проблема распределения контроллеров программно-конфигурируемых сетей ..... 67
3.2. Программно-конфигурируемые сети с распределенными контроллерами ..... 69
3.3. Разработка алгоритма динамического распределения ПКС-контроллеров ..... 72
3.4. Результаты моделирования ..... 76
3.5. Разработка модели классификации и приоритизации трафика в программно-конфигурируемых сетях ..... 79
3.5.1. Характеристики сетевого трафика ..... 79
3.5.2. Модифицированный алгоритм кластеризации k-means ..... 80
3.5.3. Модель классификации и приоритизации трафика ПКС ..... 84
Выводы по главе 3 ..... 87
**Глава 4. Разработка модели идентификации и приоритизации трафика Интернета вещей на основе сегментации ресурсов в программно-конфигурируемых сетях ..... 88**
4.1. Концепция сетевой сегментации ..... 88
4.2. Сравнительный анализ моделей и методов сетевой сегментации ..... 89
4.2.1. Единая эталонная модель ..... 89
4.2.2. Модифицированная эталонная модель ..... 93
4.2.3. Модель сегмента с одной S/D парой ..... 94
4.2.4. Модель сегмента со многими S/D парами ..... 96
4.2.5. Контент-ориентированная модель ..... 98
4.2.6. Номинальная модель ..... 100
4.2.7. Г-надежная модель ..... 102
4.2.8. Легкая надежная модель ..... 103
4.2.9. Сравнительный анализ моделей сетевой сегментации ..... 104
4.3. Разработка модели идентификации и приоритизации трафика Интернета вещей на основе сегментации ресурсов в программно-конфигурируемых сетях . 107
4.3.1. Модель контроля параметров качества обслуживания для приоритизации приложений Интернета Вещей в программно-конфигурируемых сетях ..... 107
4.3.2. Архитектура высокого уровня модели и общие описания взаимодействия элементов ..... 108

4.3.3. Функциональные элементы модели ..... 110
4.3.4. Организация контроля QoS для приоритизации приложений ..... 112
4.3.5. Моделирования сегмента модельной сети ..... 115
Выводы по главе 4 ..... 118
**ЗАКЛЮЧЕНИЕ ..... 119**
**СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ ..... 122**
**СПИСОК ЛИТЕРАТУРЫ ..... 123**
**Приложение А. МЕТОДИКА ИСПЫТАНИЙ КОНТРОЛЛЕРА ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ ..... 135**
**Приложение Б. ЛИСТИНГ ПРИЛОЖЕНИЯ ТЕСТИРОВАНИЯ КОНТРОЛЛЕРА ПОВЕРХ MININET ..... 143**
**Приложение В. ДОКУМЕНТЫ, ПОДТВЕРЖДАЮЩИЕ ВНЕДРЕНИЕ ОСНОВНЫХ РЕЗУЛЬТАТОВ ДИССЕРТАЦИОННОЙ РАБОТЫ ..... 147**

# ВВЕДЕНИЕ

**Актуальность темы исследования.** Ожидается, что сети связи пятого поколения (5G/IMT-2020) будут обеспечивать поставку сквозных (E2E) услуг с гарантированным качеством обслуживания для огромного количества подключаемых устройств интернета вещей, поддержку разнообразных вариантов использования и приложений (в том числе умные дома, промышленная автоматизация, умные транспортные системы и системы электронного здравоохранения). Управление и администрирование сетей обычно требуют много времени и навыков для гибкого перепрограммирования сети, в основном, из-за архитектурной сложности сети, а также из-за ограничения используемых протоколов в традиционных сетевых архитектурах. Эти ограничения на реконфигурацию или перепрограммируемость сети противоречат концепции эволюции сетей пятого поколения, которые должны будут поддерживать широкий спектр подключаемых устройств (приложений) и их использование с учетом различных характеристик по мобильности, безопасности, задержкам, надежности и т. п. Увеличение потребностей в использовании сетевых технологий для различных бизнес-задач, распределенных вычислений, появление крупных дата-центров и начало виртуализации Интернета, значительный рост числа мобильных устройств и контента, виртуализация серверов и распространение облачных сервисов являются основными причинами, повлекшими за собой необходимость переосмысления традиционных сетевых архитектур. В частности, ожидается, что трафик Интернета вещей будет экспоненциально расти, поскольку к 2020 году планируется, что будет подключено более 32 миллиарда как физических, так и виртуальных объектов, генерирующих более 44 зеттабайт данных в год. Таким образом, необходимо разработать сетевую архитектуру, способную передавать трафик Интернета вещей с заданным качеством обслуживания и приоритетом.

В качестве эффективного решения данных проблем выступают программно-конфигурируемые сети (ПКС), работающие на базе протокола OpenFlow. ПКС позволяют оптимизировать управление сетью более эффективным и гибким способом. Концепция ПКС пересматривает структуру традиционных сетей связи,

разделяя уровень управления и уровень передачи данных для упрощения администрирования и контроля сети, а также внедрения новых услуг. Согласно концепции ПКС, функции контроля и управления сетью вынесены из традиционного сетевого оборудования (коммутаторы, маршрутизаторы и т. п) в специальный сервер, называемый контроллером.

ПКС-контроллер является ключевым элементом в архитектуре сети; контроллер – это сетевая операционная система с приложениями. Контроллер выполняет функции управления элементами сетевой инфраструктуры и потоками данных в сети. Динамические характеристики сети в значительной степени определяются его производительностью. Характеристиками контроллера определяется производительность, масштабируемость, надежность телекоммуникационной сети, построенной на ее основе. Предлагаемый подход заключается в создании выделенных логических сегментов сетей, называемых слайсами. Ввиду того, что на русский язык термин слайс переводится как сегмент, поэтому далее по тексту будет использоваться термин сетевая сегментация. Сетевая сегментация подразумевает разделение физической сети на несколько виртуальных сетей для конкретных приложений, развертываемых по запросу на основе требований к выделенным сетевым параметрам. Сетевая сегментация осуществляется на базе технологии виртуализации сетевых функций и ПКС. Ввиду насущной необходимости разработки моделей и методов сегментации ресурсов в ПКС для предоставление новых сервисов и услуг в сетях связи пятого поколения тема диссертационной работы является актуальной.

**Степень разработанности темы.** В последние годы появились довольно большое количество работ российских и зарубежных авторов, посвященных исследованию программно-конфигурируемых сетей и организации эффективного распределения нагрузки в программно-конфигурируемых сетях.

На сегодняшний день в научных школах, возглавляемых российскими и зарубежными учеными А. Е. Кучерявым, Р. Л. Смелянским., К. Е. Самуйловым, С. Н. Степановым, А. А. В. Росляковым, В. Г. Карташевским, Е. А. Кучерявым, М. А. Р. В. Киричком, А.С.А. Мутханной, А.В. Шалимовым и зарубежными А

Tootoonchian, R. Sherwood, O. Salman, Y. Zhao и др. ведутся работы по исследованию программно-конфигурируемых сетей. В частности, выявлен ряд проблем производительности ПКС-контроллеров и предложено немало подходов по улучшению показателей сетей. Несмотря на то, что в целом указанная область исследуется довольно активно, ряд вопросов остается нерешенными. Необходимо отметить объективно малое количество работ, посвященных методам сегментации ресурсов в ПКС, сочетающим оперативность реакций на происходящие изменения на уровне сетевых приложений, а также методам динамического распределения трафика по запросу, способным функционировать в режиме близко к реальным времени и обеспечивать требуемое качество обслуживания.

**Цель работы и задачи исследования.** Целью диссертационной работы является разработка моделей и методов сетевой сегментации в программно-конфигурируемых сетях и исследование характеристик предложенных моделей.

Для достижения поставленной цели решаются следующие задачи:

– проанализировать и выявить области эффективного применения и развертывания ПКС для сетей связи пятого поколения 5G/IMT-2020;

– разработать модели ПКС для оценки задержки пакетов и параметров качества облуживания для эффективного функционирования ПКС;

– разработать метод кластеризации ресурсов программно-конфигурируемых сетей, позволяющий осуществлять распределение контроллеров ПКС и балансировку трафика, в зависимости от приложений;

– проанализировать применимость методов балансировки трафика, основанных на сетевой сегментации, для автоматизации процесса динамического распределения ресурсов ПКС по приложениям;

– разработать самоорганизующейся модели сети для балансировки трафика, способной гибко реагировать на изменения сетевых требований в режиме, близком к реальному времени.

**Объект исследования –** программно-конфигурируемые сети.

**Предмет исследования** – модели и методы сегментации ресурсов программно-конфигурируемых сетей.

**Методологические и теоретические основы исследования.** Проводимые исследования базируются на теории массового обслуживания, математической статистике, методах моделирования и натурных экспериментах. Моделирование фрагмента программно-конфигурируемой сети проведено на основе пакета имитационного моделирования имитационного моделирования AnyLogic и CloudSim.

**Научная новизна исследования**.

1. Разработана модель программно-конфигурируемой сети для оценки задержки, отличающаяся от известных тем, что элементы ПКС представлены в виде кластеров.

2. Разработан метод классификации трафика в программно-конфигурируемых сетях на основе алгоритма k-means, отличающийся от известных тем, что в алгоритме учитываются параметры трафика и методы его обнаружения на базе предварительного обучения.

3. Разработана модель сегментации ресурсов в программно-конфигурируемых сетях, отличающаяся от известных тем, что при сегментации ресурсов учитываются требования для приложений сетей связи пятого поколения.

**Теоретическая и практическая значимость работы.** Теоретическая значимость работы состоит в разработке модели сегментации ресурсов в программно-конфигурируемых сетях за счет распределения контроллеров ПКС и балансировки трафика, в зависимости от приложений, с использованием подходов кластеризации ПКС-контроллеров. Модель позволяет упростить процесс представления сетевых услуг в сетях 5G/IMT-2020 по сравнению с традиционными моделями реализациями, позволяя осуществить динамическое распределение ресурсов сети по требованиям: передачу трафика с учетом требований к обслуживанию, управление трафиком с учетом физического состояния каналов, объединение ресурсов облака и пропускной способности сети.

Практическая значимость диссертационной работы подтверждается актом внедрения и состоит в разработке модельной сети для тестирования ПКС и методике тестирования контроллеров ПКС.

**Основные положения и результаты, выносимые на защиту:**

1. Модель программно-конфигурируемой сети, позволяющая оценить эффективность её работы в условиях высокой нагрузки и рационально планировать размещение элементов сети на этапе развертывания и масштабирования.

2. Метод кластеризации ресурсов программно-конфигурируемых сетей для динамического распределения контроллеров, модель классификации и приоритизации трафика в программно-конфигурируемых сетях.

3. Модель идентификации и приоритизации трафика Интернета Вещей на основе сегментации ресурсов в программно-конфигурируемых сетях для осуществления динамического распределение ресурсов.

**Степень достоверности и апробация результатов.** Достоверность полученных автором научных и практических результатов определяется обоснованным выбором исходных данных при постановке частных задач исследования, основных допущений и ограничений, принятых в процессе математического моделирования, соответствием расчетов с результатами экспериментальных исследований, проведенных лично автором, согласованностью с данными, полученными другими авторами и апробацией результатов исследований на международных, всероссийских и ведомственных научно-технических конференциях и конгрессах. Основные теоретические и практические результаты работы реализованы в учебном процессе кафедры Сетей связи и передачи данных Санкт-Петербургского государственного университета телекоммуникаций им. проф. М. А. Бонч-Бруевича при чтении лекций, проведении практических занятий и лабораторных работ. Кроме того, научные результаты, полученные Мухизи Самуэлем, были использованы при подготовке вкладов СПбГУТ в Сектор Стандартизации Телекоммуникаций Международного Союза Электросвязи.

**Апробация результатов исследования.** Основные результаты диссертационной работы докладывались и обсуждались на IV Международной научно-технической и научно-методической конференции «Актуальные проблемы инфотелекоммуникаций в науке и образовании» (Санкт-Петербург, 2015),

Международной научной конференции «Молодежная научная школа по прикладной теории вероятностей и телекоммуникационным технологиям» (Москва, 2017), 71–73 Всероссийской научно-технической конференции, посвященной Дню радио (Санкт-Петербург 2016-2018), International Conference on Wired/Wireless Internet Communication (Санкт-Петербург, 2017); III Международной конференции молодых ученых «Интернете вещей и его приложения» INTHITEN ( Санкт-Петербург 2017), X Международном конгрессе по ультрасовременным системам телекоммуникаций и управления (Санкт-Петербург, 2018), XXI International Conference on Distributed Computer and Communication Networks (Москва 2018).

**Публикации по теме диссертации**. По теме диссертации опубликовано 10 работ, из них: 3 в рецензируемых научных изданиях; 3 в изданиях, индексируемых в международных базах данных; 4 в других изданиях и материалах конференций.

**Личный вклад автора.** Основные результаты теоретических и экспериментальных исследований получены автором самостоятельно. В работах, опубликованных в соавторстве, соискателю принадлежит основная роль при постановке и решении задач, а также обобщении полученных результатов.

**Соответствие специальности.** Диссертационная работа соответствует пунктам 3, 4, 14 паспорта специальности 05.12.13 – «Системы, сети и устройства телекоммуникаций».

**Структура и объем диссертации.** Диссертация состоит из введения, четырех глав, выводов, заключения, списка используемых источников и трёх приложений. Общий объем диссертации составляет 148 страниц, из них основного текста 134. Работа содержит 55 рисунков, 8 таблиц и список из 109 литературных источников.

# Глава 1. Анализ концепции программно-конфигурируемых сетей в сетях связи пятого поколения 5G/IMT-2020

## 1.1. Перспективы развития сетей связи 5G/IMT-2020

В последние годы эволюция мобильной связи и ее интеграция в повседневную жизнь всего общества оказывает очевидное влияние на экономическое и социальное развитие [1; 4]. Сети связи 5G/IMT-2020 будут сталкиваться с большим количеством новых подключаемых приложений [52]. Управление и эксплуатация сетей должны основываться на подключенных к сети приложениях [5]. Сети, основанные на приложениях, состоят из взаимосвязанных устройств, различных модулей, машин, датчиков и исполнительных механизмов, и множества клиентов, подключённых к Интернету, генерирующих большие потоки трафика [6; 63]. Проектирование технологии сетей связи 5G/IMT-2020 должно отвечать на потребность разработки новых моделей сети, способных быть открытыми, более гибкими и масштабируемыми [59; 78]. Будущие сети связи должны упрощать процесс настройки сети по требованиям конкретных предоставляемых услуг (Рисунок 1).

Сети связи 5G/IMT-2020 будут обеспечить конвергентную сетевую связь в сетях с различными технологиями и обеспечить открытую систему связи для взаимодействия со спутниковыми системами, сотовыми сетями, облачными системами, информационными центрами, домашними шлюзами и многими другими открытыми сетями и устройствами. Кроме того, сети связи 5G/IMT-2020 будут автономными и программируемыми. Соответственно, безопасность, отказоустойчивость, надежность и целостность данных станут первоочередной задачей при проектировании будущих сетей. Помимо этого, сети связи 5G/IMT-2020 должны обеспечить мобильность пользователей для осуществления возможности подключения в любом месте, в любое время и в любых сочетаниях [25].

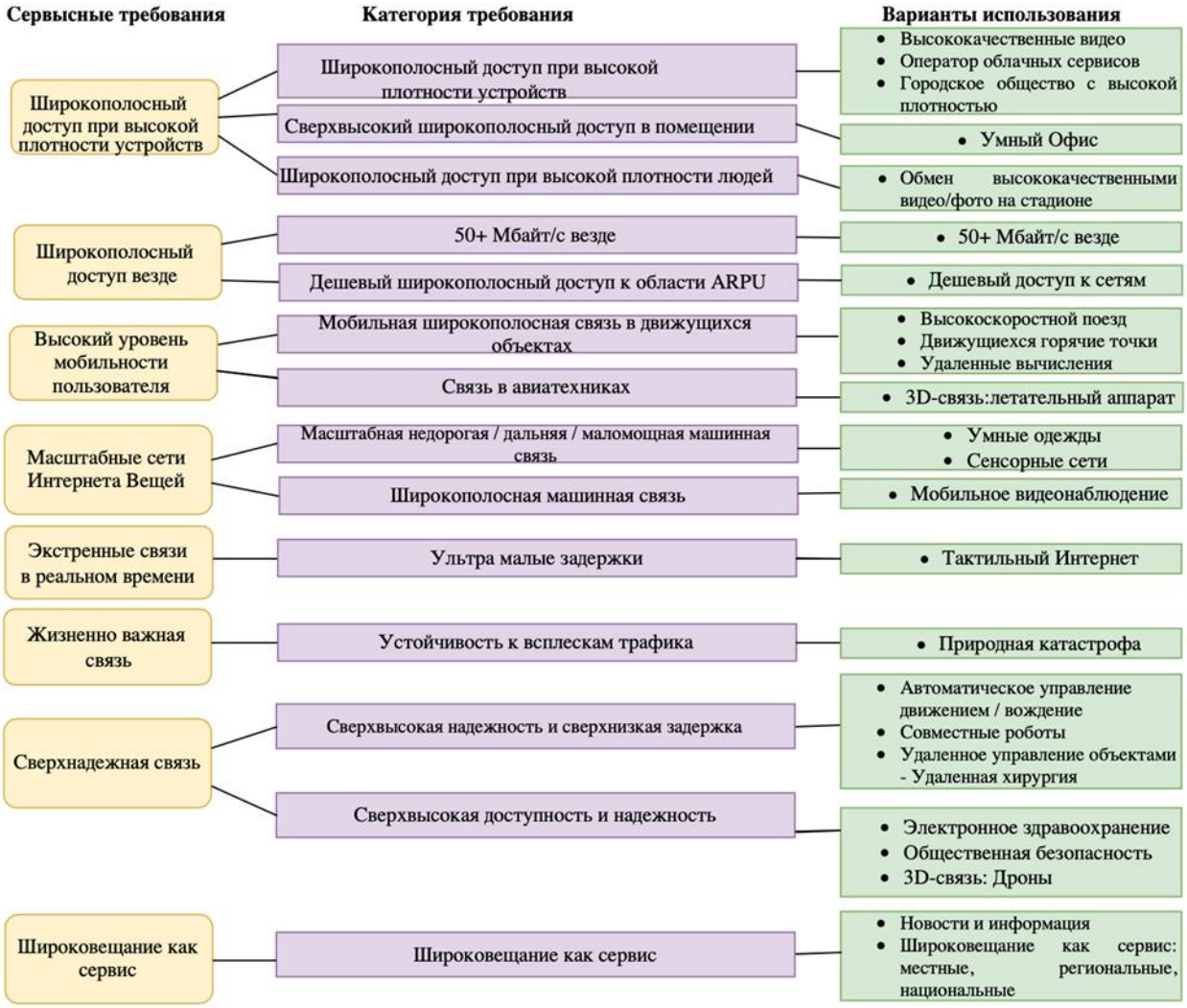


**Рисунок 1** – Концептуальный сценарий реализации сетей 5G/IMT-2020

## 1.2. Необходимость программирования сетей связи 5G/IMT-2020

Для достижения требуемой гибкости сначала необходимо, чтобы существующие ресурсы в инфраструктуре могли быть адаптированы и изменены динамически [73]. Кроме того, нужны методы и механизмы для разработки приложений и сервисов поверх гибкой инфраструктуры в разных технологических областях. Для достижения этих требований необходима архитектура управления с высоким уровнем программируемости. В частности, архитектура управления не должна быть привязана к конкретному варианту использования или сценарию, и должна позволить сетевому оператору программировать настраиваемые алгоритмы на уровне управления для оптимизации сети радиодоступа RAN, транспортной сети и сетевых ресурсов облачных платформ. Дополнительно необходимы следующие функции:

1) **модульность**. Архитектура управления должна следовать модульной архитектуре с четко определенными функциями управления и интерфейсами. Интерфейсы и архитектурные строительные блоки также должны поддерживать стекирование рекурсивным образом, чтобы позволить внедрение системы, адаптированной к конкретным сценариям;

2) **виртуализация**. Архитектура должна иметь возможность разделить физические и виртуальные ресурсы инфраструктуры на отдельные группы (или фрагменты или сечения) и распределить их по разным клиентам. Здесь клиенты могут быть контроллерами более высокого уровня с функциями обслуживания различных приложений. Выделенные фрагменты сети должны быть изолированы друг от друга по причине безопасности и производительности (например, для предотвращения негативного воздействия перегруженного фрагмента сети на другие фрагменты);

3) **масштабируемость**. Управление ресурсами в каждом из доменов является сложной задачей, так как обычно мы имеем дело с большим количеством сетевых элементов, а также с параметрами и процедурами управления. Таким образом, совместный контроль над доменами может легко стать неразрешимым, чего следует избегать при правильном проектировании архитектуры управления. Также необходимы подходящие методы абстракции, чтобы ограничить сложность в более высоких слоях и сделать общую проблему оптимизации управляемой. Для удовлетворения этих требований в разработке общей архитектуры управления, применяются принципы организации сети, основанные на технологии ПКС и виртуализации сетевых функций.

### 1.3. Виртуализация сетей в сетях связи 5G/IMT-2020

Организация сетей с программируемыми параметрами и виртуализацией сетевых функций (NFV) представляют собой будущее электросвязи, в котором виртуализированная инфраструктура и услуги обеспечивают беспрецедентную гибкость, интеллектуализацию и открытость [25; 28; 36; 65; 77; 84; 85].

В течение последних пяти лет технологии программно-конфигурируемых сетей и виртуализации сетевых функции совершенствовались благодаря уникальному взаимодействию организаций по стандартизации с сообществами разработчиков программного обеспечения с открытым исходным кодом, которые вместе меняют методы принятия новой технологии [7; 100; 102].

Инновационные отраслевые группы, такие как Рабочая группа ISG ETSI по NFV и организация ONF (Open Networking Foundation) создали эталонные архитектуры, обосновали сценарии использования и изменили требования к составным элементам с открытым исходным кодом, которые являются неотъемлемой частью NFV и ПКС.

Технологии программно-конфигурируемых сетей и виртуализации сетевых функций, стали важнейшим ключом для сетей 5G/IMT-2020, способствовавшим разработке широкого круга приложений, в том числе приложения для подвижной широкополосной связи, Интернета вещей, связи между подвижными терминалами (M2M) и т. д. [58, 66].

Для того, чтобы обеспечить возможность использования такого широкого круга приложений для конечных пользователей, модель управления и контроля SDN/NFV должна стать намного более масштабируемой, интеллектуальной, гибкой и открытой, чем когда-либо раньше (Рисунок 2).

По этим причинам многие наиболее прогрессивные и инициативные операторы и поставщики сетевых услуг в отрасли электросвязи взяли на себя задачу изменения жизненного цикла процесса предоставления услуг. Для этого необходимо беспрецедентное взаимодействие организаций по стандартизации, операторов, отраслевых организаций и сообщества разработчиков программного обеспечения с открытым исходным кодом.

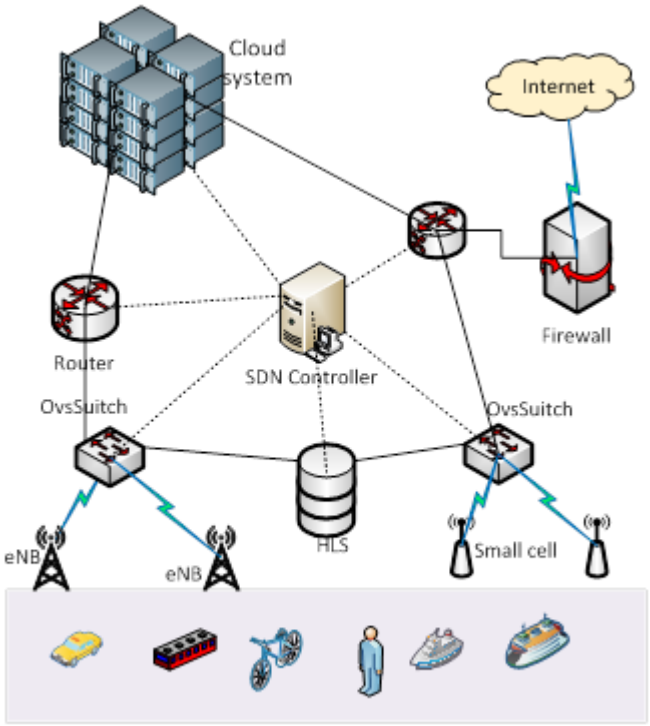


**Рисунок 2** – Общая архитектура SDN в сетей подвижной связи

## 1.4. Анализ концепции программно-конфигурируемых сетей и протокола OpenFlow

### *1.4.1. Развитие программно-конфигурируемых сетей*

Популярность программно-конфигурируемых сетей стала ощутимо расти в последние несколько лет, однако многие идеи, лежащие в основе этой технологии, постепенно развивались уже в течении двадцати и более лет. С одной стороны, технология ПКС пересматривает структуру ранних телефонных сетей, которые четко разделяли уровень управления и уровень передачи данных для упрощения администрирования сети и внедрения новых услуг. С другой стороны, технология ПКС сетей имеет сходство с концепцией традиционных сетей связи, которая исследовалась в конце 20-го века и сформировала видение программируемых сетей, хотя и с большим акцентом на программируемые уровни передачи данных. Сети с центральным управлением поддерживали многие из тех же принципов, что и ПКС сейчас, однако не имели четкого представления о применении и пути развития. После многочисленных исследований, видение программируемых сетей стало более отчетливым, что выразилось в идее разделения уровня управления и передачи данных и позволило приложениям и сетевым службам напрямую

контролировать абстрагированную сетевую инфраструктуру через интерфейсы прикладного программирования (API – Application Programming Interface) [18; 23; 70; 104].

До появления протокола OpenFlow, идеи, лежащие в основе программно-конфигурируемых сетей, сталкивались с трудностями из-за разногласий в видении полностью программируемых сетей и реализуемости, которая позволила внедрить ПКС в уже существующую сетевую инфраструктуру. Появление протокола OpenFlow в 2009 году разрешило дискуссию, протокол предоставлял больше контроля над сетевыми устройствами и поддерживался уже существующими коммутационными устройствами. Несмотря на то, что эксплуатация прежнего технического оснащения несколько ограничивала гибкость, протокол OpenFlow быстро устанавливался и встраивался, обеспечивая технологии ПКС активное развитие. Далее последовало производство ПКС-контроллеров, что, в свою очередь, стало стимулом к созданию OpenFlow API для разработки приложений управления.

Успех протокола заключался в технических условиях, созданных в отрасли. Еще до появления OpenFlow производители коммутационных чипсетов, такие как Broadcom, начали внедрять свои API, чтобы программисты могли контролировать определенные сетевые характеристики сетевого оборужования. В частности, благодаря тому, что коммутаторы уже поддерживали средства детального контроля доступа и мониторинг потока, внедрение начального набора возможностей протокола OpenFlow было простым и понятным, производителям не нужно было улучшать устройство для их совместимости с протоколом.

Первоначальным объектом для тестирования протокола OpenFlow стали сети университетских городков, которые были идеальной площадкой, отвечающей запросам сетевого исследовательского сообщества. В конце 2000-х годов в Стэнфорде были приложены все усилия для развертывания испытательных стендов протокола OpenFlow и проведены демонстрации возможностей протокола как в одной сети кампуса, так и в магистральных сетях, охватывающей несколько кампусов. После успеха тестирования в Стэнфорде, протокол OpenFlow

постепенно начал приживаться и в других сферах, например в таких, как частные глобальные сети и центры обработки данных, где существовала четкая потребность управления сетевым трафиком в больших объемах.

Важным этапом в развитии технологии программно-конфигурируемых сетей стало создание некоммерческой организации Open Networking Foundation (ONF) в 2011 году, куда вошли крупнейшие всемирно известные компании в области информационных технологий, такие как Deutsche Telekom, Facebook, Google, Microsoft, Verizon, и Yahoo! и др. [87]. По состоянию на 2017 год в состав организации входило порядка 150 компаний, среди которых как крупнейшие телеком-операторы и производители сетевого устройства, так малые инновационные предприятия – стартапы. ONF отвечает также за разработку программ сертификации устройства на соответствие стандартам. Задачей данного проекта является содействие в популяризации программно-конфигурируемых сетей посредством разработки открытых стандартов.

Несмотря на то, что технология ПКС достигла крупных успехов на раннем этапе своего развития, в данной области требуется больше исследований для улучшения уже существующей инфраструктуры и изучения потенциала с целью решения большего количества задач. Недавние исследовательские работы рассматривают задачи функциональности ПКС-контроллера и взаимосвязи между ПКС-контроллером и остальными информационными технологиями, например, корпоративные сети, сети связи 5G/IMT-2020 и т. п.

По мере увеличения интереса к функциональности ПКС, большое внимание на построение архитектуры сместилось с гибкой передачи пакетов на динамическую виртуализацию ресурсов и оркестрацию сервисов. В результате полученная архитектура применима ко всем видам приложений в сетях предприятия, операторов, центров обработки данных и сетях кампусов, от конечного потребителя до владельца аппаратного обеспечения, как для абсолютно новых, так и для развивающихся существующих сетевых реализаций, что адаптирует ПКС внутри и между различными сетевыми доменами (например, внутри и между операторскими, внутри и между центрами обработки данных).

### *1.4.2. Архитектурная концепция и принципы построения программно-конфигурируемых сетей*

Традиционные компьютерные сети состоят из взаимосвязанных сетевых устройств, такие как коммутаторы и маршрутизаторы (Рисунок 3). В каждом устройстве находится механизм передачи на уровне передачи и система управления, которая включает в себя операционную систему и приложения. В этой модели сетевые устройства имеют закрытую архитектуру и нет возможности добавить новые функции. Привязка сетевых устройств к выбранному сетевому производителю не гарантирует поддержку будущих приложений и сервисов.

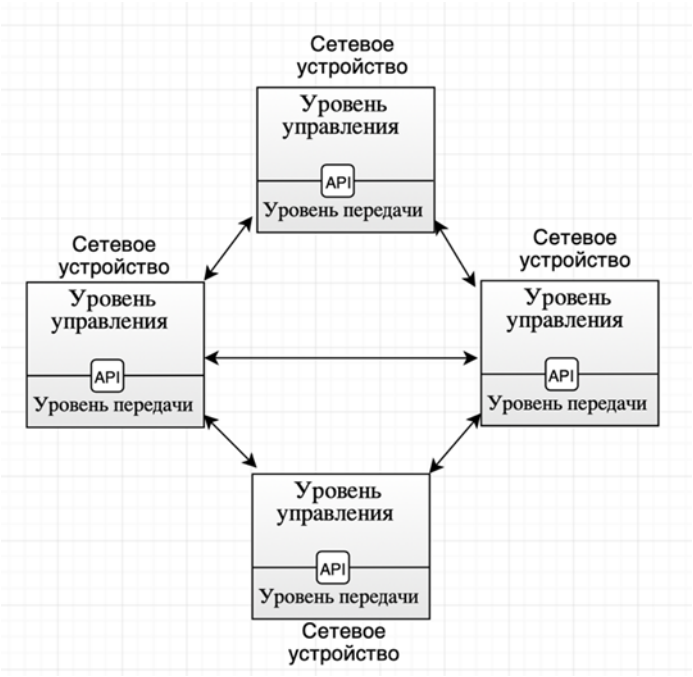


**Рисунок 3** – Структура взаимодействия устройств в традиционной сети

Архитектура программно-конфигурируемых сетей определена в ONF TR-502 [86], она основана на следующих трех принципах:

1) разделение функции управления и передачи трафика. Это принцип свободы развёртывания, жизненного цикла и развития систем управления и передачи трафика;

2) логически-централизованное управление. Логически централизованное управление означает, что управление выглядит снаружи (клиент/приложение) как единое целое, что гарантирует, что ресурсы могут использоваться более эффективно по сравнению с локально ограниченной перспективой;

3) программируемость сетевых сервисов. Интерфейсы между компонентами ПКС раскрывают абстракции ресурсов и сетевое состояние, а также обеспечивают обмен информацией и операции по программированию и управлению сетью. Приложения могут задать требования и запрашивать изменения в своих сетевых службах, а также программно реагировать на состояния сети.

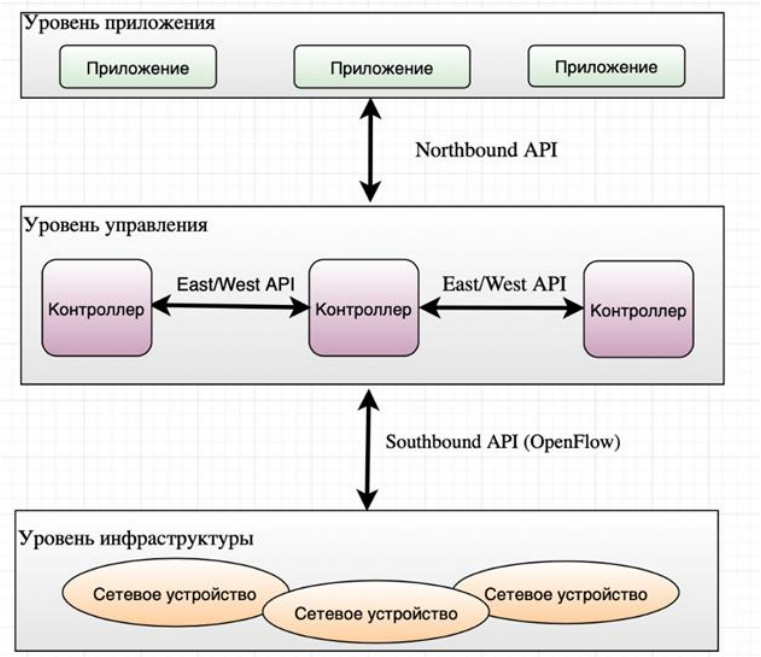


**Рисунок 4** – Общая архитектура ПКС

### *1.4.3. Архитектура программно-конфигурируемых сетей*

Как показано на Рисунке 4, архитектура ПКС содержит шесть основных компонентов.

1. **Уровень приложений.** На верхнем уровне, называемом уровнем приложений или прикладным уровнем, реализуются различные службы, такие как системы обнаружения и предотвращения вторжений (IDS/IPS), функция обеспечения качества обслуживания (QoS), управление доступом, прокси-сервер и другие, которые определяют поведение сети.

2. **Уровень управления.** Уровень управления абстрагирует топологию сети для уровня приложений посредством northbound API (северного API-интерфейсы). На этом уровне основным элементом является ПКС-контроллер, который отвечает за координацию одного или нескольких сетевых устройств, находящихся на уровне инфраструктуры. Контроллер формирует правила обработки данных для этих

устройств, отслеживает их состояние и осуществляет сбор сетевой информации через southbound API (южный API-интерфейс). Протокол OpenFlow является примером реализации такого API. На этом уровне также присутствуют такие элементы, как eastbound и westbound API (восточный и западный API-интерфейсы), которые обеспечивают связь между объектами уровня управления – контроллерами и позволяют им обмениваться информацией, касающейся обработки трафика на уровне инфраструктуры.

3. **Уровень инфраструктуры.** Нижний уровень, который известен как уровень инфраструктуры или уровень передачи данных, обеспечивает обработку и пересылку пакетов на основании полученных инструкций от уровня управления. На этом уровне находятся сетевые устройства, управляемые контроллером, например, такие, как коммутаторы.

4. **Northbound interfaces.** Северные API-интерфейсы представляют собой программируемый виртуальный сетевой интерфейс для приложений и систему управления поверх стека ПКС. Именно эти интерфейсы наиболее важные API-интерфейсы архитектуры ПКС, так как они позволяют приложениям использовать сетевые сервисы и динамически настроить сеть. Сетевые сервисы, которые могут быть развернуты и оптимизированы с помощью северных API-интерфейсов, включают сервисы безопасности, балансировку нагрузки, управление трафиком, качеством обслуживания и т. п.

5. **Southbound interfaces.** Южные API-интерфейсы обеспечивают эффективное управление сетью и позволяют контроллеру динамически организовывать сетевые ресурсы в соответствии с требованиями и потребностями в реальном времени. Южные API-интерфейсы в отличии от северных взаимодействуют с сетевыми устройствами (коммутаторы, маршрутизатор). Из различных протоколов южного API-интерфейсов наиболее используемый – OpenFlow.

6. **East/West interfaces**. Восточные/Западней API-интерфейсы обеспечивают связь между объектами уровня управления – контроллерами и позволяют им

обмениваться информацией, касающейся обработки трафика на уровне инфраструктуры.

### *1.4.4. Контроллер программно-конфигурируемой сети*

Контроллер является наиболее важным компонентом архитектуры ПКС, узлом, который централизует сетевые функции и отслеживает глобальное состояние сети. Контроллер программно-конфигурируемых сетей может быть представлен как аппаратно-программное решение, так и в качестве программной реализации [95]. На данный момент, контроллеры на рынке чаще представлены в программной реализации. Извлекая уровень управления из сетевого оборудования и запуская его как программное обеспечение, контроллер упрощает автоматическое управление сетью, а также интеграцию и администрирование бизнес-приложений. Отметим, что согласно OFN архитектуры ПКС не определяет внутренний дизайн контроллера. Первые версии архитектуры ПКС представляют план управления, состоящий из одного централизованного контроллера. Позже были предложены распределенные архитектуры с использованием нескольких контроллеров для повышения производительности и масштабируемости сети.

Среди основных характеристик контролера программно-конфигурируемых сетей выделяют следующие:

1) производительность – число потоков, обрабатываемых контроллером в единицу времени [потоки/с];

2) время обработки – количество времени, затрачиваемое контроллером на обработку запроса от коммутатора [с];

3) надежность – число отказов при заданном профиле нагрузки;

4) ресурсоемкость – утилизация контроллером оперативной памяти физического сервера, и нагрузка на ядра процессора;

5) масштабируемость – поддержка контроллером многопоточности.

ПКС-контроллер обычно содержит набор «подключаемых» модулей, которые могут выполнять различные сетевые задачи. Основные модули включают в себя:

– модуль обнаружения каналов;

– модуль топологии;

– модуль памяти;

– модуль выработки стратегии;

– модуль таблицы потоков;

– модуль управления данными.

Также могут внедряться другие модули, которые поддерживают более сложные функции.

За предоставление услуги маршрутизации отвечают два модуля: модуль топологии и модуль обнаружения каналов. Модуль обнаружения каналов отвечает за обнаружение и поддержание статуса физических соединений в сети. Процедура запускается контроллером, когда какой-либо неизвестный тип трафика попадает в домен OpenFlow. Таким образом, информация, собранная модулем обнаружения каналов, используется для создания базы данных соседей на контроллере. По этой БД модуль топологии строит, и после, поддерживает обновление информации о топологии и вычисляет маршруты в сети.

На сегодняшний день, существует немалое количество контроллеров, большинство из которых с открытым исходным кодом и поддерживают протокол OpenFlow. Эти контроллеры различаются языками программирования, поддерживаемой версией OpenFlow, методами, используемыми в качестве многопоточности и производительностью в качестве битрейта. Таблица 1 суммирует характеристики наиболее распространенных ПКС-контроллеров [95].

**Таблица 1** – Характеристики наиболее распространенных ПКС-контроллеров

| *Контроллер* | *ЯП* | *GUI* | *Southbound API* | *Платформ* | *Партнеры* |
|---|---|---|---|---|---|
| ONOS | Java | Web | OF 1.0, 1.3, 1.5, NETCONF, SNMP | Linux, MAC OS, Windows | ON.LAB, At&T, Ciena, Cisco, Ericsson, Fujitsu, Huawei, Intel, Nec, Sk Telecom |
| OpenDaylight | Java | Web | OF 1.0, 1.3–1.5, NETCONF/ YANG, OVSDB, PCEP, BGP/LS, LISP, SNMP | Linux, MAC OS, Windows | Linux Foundation |

Продолжение таблицы 1

| *Контроллер* | *ЯП* | *GUI* | *Southbound API* | *Платформ* | *Партнеры* |
|---|---|---|---|---|---|
| Floodlight | Java | Web/ Java | OF 1.0–1.4 | Linux, MAC OS, Windows | Big Switch Networks |
| NOX | C++ | Python + QT4 | OF 1.0 | Linux | Nicira |
| POX | Python | Python + QT4 | OF 1.0 | Linux, MAC OS, Windows | Nicira |
| RunOS | C++ | Web | OF 1.3 | Linux | ЦПИКС |
| RYU | Python | Web | OF 1.0, 1.2–1.5 NETCONF, OF-Config | Linux | Nippo Telegraph, Telephone Corporation |
| Beacon | Java | Web | OF 1.0 | Linux, MAC OS, Windows | Standford University |
| Maestro | Java | - | OF 1.0 | Linux, MAC OS, Windows | RICE, NSF |
| OpenMUL | C | Web | OF 1.0, 1.3, 1.4, OVSDB, OF-Config | Linux | Kulcloud |

В рамках экспериментов и тестирования ПКС моделей в данной диссертации был использован контроллер OpenDaylight, разработанный консорциумом OpenDaylight Project [88].

### *1.4.5. Протокол OpenFlow*

Протокол OpenFlow является открытым стандартом. Он был предложен ONF для стандартизации взаимодействия контроллера с сетевыми устройствами в архитектуре ПКС. Следуя эволюционному пути развития сетей связи, протокол OpenFlow, на данном этапе развития и практической реализации концепции ПКС, внедряется в виде программного модуля в аппаратные реализации Ethernet коммутаторов, маршрутизаторов и беспроводных узлов доступов, в качестве расширения возможностей и их применения в качестве устройств ПКС [85].

Стандарт OpenFlow в настоящее время де-факто принят большинством производителей сетевого оборудования. На телекоммуникационном рынке в

настоящее время доступны коммутаторы с поддержкой OpenFlow таких производителей как Cisco, Juniper, Brocade, Huawei, Zelax, B4N и др.

Сообщения протокола OpenFlow принято делить на три типа:

1. Сообщения ***контроллер-коммутатор*** – инициализируются на контроллере, служат для управления коммутатором и контролем за событиями, происходящими на нем. К данному типу сообщений относятся следующие сообщения:

– *Features*: данное сообщение служит для запроса контроллером возможностей коммутатора; коммутатор в свою очередь отвечает на такой запрос ответом features, в котором обозначает свои возможности. Такой процесс происходит при открытии OpenFlow канала;

– *Configuration*: настоящим сообщением контроллер запрашивает и устанавливает параметры настройки коммутатора;

– *Modify-State*: эти сообщения отсылаются сетевой ОС для управления состоянием коммутаторов. Задача сообщений добавление, удаление правил и изменение OpenFlow таблиц, настройка портов коммутатора;

– *Read-State*: данные сообщения отвечают за сбор статистики коммутаторов;

– *Packet-out*: сообщения Packet-out используются контроллером для отправки пакетов из определенного порта на коммутаторе и пересылке пакетов, полученных с помощью сообщения Packet-in, которые содержат целый пакет или идентификатор ID буфера, ссылающегося на пакет, загруженный в коммутатор. Сообщение должно содержать список действий, которые применяются в указанном порядке: если список действий пуст, то пакет сбрасывается;

– *Barrier*: сообщения Barrier обеспечивают установление зависимостей между сообщениями или оповещения о завершенных операциях. Используются при необходимости обработки сообщений в определенном порядке;

– *Role-Request*: данный запрос служит для изменения приоритета контроллера на коммутаторе (для повышения роли со Slave до Master);

– *Asynchronous-Configuration*: с помощью данного сообщения контроллер устанавливает фильтр на асинхронные сообщения от коммутаторов.

2. Вторым типом сообщений являются ***асинхронные сообщения***, которые инициализируются OpenFlow-коммутаторами, предназначены для извещения контроллера о событиях на сети, например, сбои, ошибки, изменения состояния. К данному типу сообщений относятся следующие сообщения:

– *Packet-in*: данное сообщение коммутатор инициирует и отсылает контроллеру в случае, если пришедший пакет не имеет нужного правила в таблице коммутации. Для всех пакетов, пересылаемых в виртуальный порт, сообщение Packet-in отправляется на контроллер;

– *Flow-Removed*: сообщение для удаления правил, которые не используются и неактивны;

– *Port-status*: генерируются коммутатором на контроллер в случае изменения настроек порта;

– *Error*: этим сообщением контроллер извещает о произошедших на нем ошибках или сбоях.

3. Третьим типом сообщений являются ***симметричные сообщения***, которые рассылаются как коммутаторами, так и контроллером. К данному типу сообщений относятся следующие сообщения:

– *Hello*: сообщения, обмен которыми между коммутатором и контроллером происходит при установлении соединения;

– *Echo*: сообщения вида запрос/ответ могут инициироваться и контроллером, и коммутатором, при условии, что обязательно будет получен ответ. Также могут служить для измерения задержек или пропускной способности соединения контроллер-коммутатор, а также проверки эффективности соединения;

– *Experimenter*: сообщения Experimenter предназначены для обеспечения дополнительной функциональности, при проведении экспериментов в пространстве типов сообщений OpenFlow.

### *1.4.6. Протокол OF-CONFIG*

Данный протокол разработан для решения задач более высокого, по сравнению с протоколом OpenFlow, уровня [83]. В основном это задачи по

построению сетевой среды в целом, конфигурации коммутаторов и принятию решений, например, о закрытии или открытии отдельных портов. Протоколом OF-CONFIG (OpenFlow Management and Configuration Protocol) предусмотрены следующие абстракции (Рисунок 5):

– логический коммутатор OpenFlow – абстракция узла передачи данных OpenFlow. Протокол OF-CONFIG позволяет осуществить конфигурацию логического коммутатора OpenFlow так, чтобы контроллер OpenFlow мог взаимодействовать с ним и управлять им по протоколу OpenFlow;

– OpenFlow совместимый коммутатор – физический или логический сетевой элемент, ресурсы которого (порты, очереди и пр.) выделены одним или несколькими логическими коммутаторами OpenFlow. Протокол OF-CONFIG позволяет динамически назначать ресурсы OpenFlow-совместимого коммутатора размещенным на нем логическим коммутаторам OpenFlow;

– точка конфигурации OpenFlow – источник сообщений OF-CONFIG для OpenFlow-совместимых коммутаторов. Сущность точки конфигурации OpenFlow и взаимодействие точек конфигурации с контроллерами OpenFlow в настоящий момент спецификациями ONF не регламентируются.

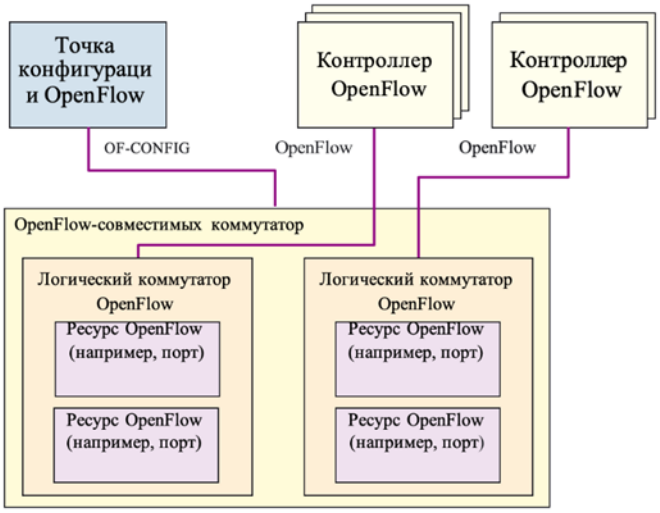


**Рисунок 5** – Основные абстракции протокола OF-CONFIG

В качестве базового протокола для OF-CONFIG спецификацией 1.1.1 определен протокол NETCONF [82].

Протокол OF-CONFIG используется для решения следующих задач:

– назначение одного или более контроллеров OpenFlow коммутатору;

– конфигурирование портов и очередей;

– удаленное изменение свойств портов;

– конфигурирование сертификатов для безопасного взаимодействия логических коммутаторов OpenFlow и контроллеров OpenFlow;

– запрос возможностей логических коммутаторов OpenFlow;

– конфигурирование ограниченного набора туннелей (IP-in-GRE, NV-GRE, VxLAN);

– инициализация логических коммутаторов OpenFlow;

– назначение ресурсов OpenFlow-совместимого коммутатора одному и более логическому коммутатору OpenFlow;

– поддержка согласованных моделей участка передачи (Negotiable Datapath Model, NDM).

Предполагается, что следующие версии OF-CONFIG будут предусматривать также такие возможности, как обнаружение коммутаторов и топологии, конфигурирование характеристик, обработка триггеров, связанных с событиями, инициализация сети OpenFlow, поддержка большего числа конфигурируемых туннелей.

Спецификацией протокола OF-CONFIG определена модель передаваемых протоколом данных, которая имеет существенное значение, так как стандартизированная модель данных сети связи является необходимым условием успешного практического применения SDN. Регламентированная модель описания сети обеспечивает согласованное представление о состоянии сети SDN, всех ее составляющих и возможность независимой разработки программных средств для работы с сетью SDN посредством протоколов консорциума ONF при сохранении совместимости.

### *1.4.7. Протоколы NB-API*

До середины 2013 г. ONF целенаправленно не разрабатывал и не накладывал требований на NB-API (Northbound Application programming Interface, Северный

программный интерфейс), реализуемый контроллерами для организации взаимодействия с сетевыми приложениями. Решения, создаваемые для реализации функций контроллеров, предлагают собственные API, конкурирующие в рыночных условиях.

Сложившаяся тенденция в практических решениях SDN предлагает два типа реализации NB-API интерфейса:

1) REST (или RESTful – Representational State Transfer) – архитектурное решение взаимодействия компонент распределенного приложения в сети, на основе модели «клиент – сервер», в основе обычно используется протокол HTTP и форматы передачи данных – *.xml, *.json;

2) программный интерфейс взаимодействия ПО (например, Java API платформы Java EE или Java OSGI API), который непосредственно зависит от реализации контроллера программно-конфигурируемой сети.

Обычно, как в первом (REST API), так и во втором случае, существует документация на предоставляемый программный интерфейс, а также возможные ответы системы (в данном случае контроллера) на соответствующие формируемые запросы. Тем самым, контроллер Программируемой сети, с точки зрения администратора или стороннего разработчика приложений под контроллер, выглядит как «черный ящик» с известным набором функций и реакций на них.

### *1.4.8. Порты OpenFlow*

Пакеты OpenFlow принимаются на входной порт и обрабатываются конвейером, который может перенаправить их на выходной порт, используя выходные действия, которые определяют: как пакет возвращается обратно в сеть. Порты OpenFlow являются сетевыми интерфейсами для передачи пакетов между обработкой OpenFlow и остальной частью сети.

Коммутатор OpenFlow должен поддерживать три типа портов: физические, логические и зарезервированные;

1) физические порты – порты, соответствующие аппаратным интерфейсам коммутатора;

2) логические порты – порты, которые не соответствуют непосредственно аппаратным интерфейсам коммутатора. Обработка, выполняемая логическим портом, зависит от реализации и должна быть прозрачной для обработки OpenFlow;

3) зарезервированные порты – порты, определяющие общие действия пересылки, такие как отправка контроллеру, рассылка или пересылка с использованием методов, отличных от OpenFlow. Включают в себя обязательные порты: ALL, CONTROLLER, TABLE, IN_PORT, ANY, UNSET и опциональные: LOCAL, NORMAL, FLOOD.

### *1.4.9. Канал OpenFlow*

Канал OpenFlow используется для обмена сообщениями между коммутатором OpenFlow и контроллером OpenFlow. Контроллер может управлять несколькими каналами, каждый из которых предназначен для разных коммутаторов. В свою очередь, коммутатор также может иметь каналы для нескольких контроллеров.

Канал OpenFlow обычно создается как одно сетевое соединение между коммутатором и контроллером, используя протокол TLS или обычный TCP, что гарантирует поддержку соединений OpenFlow в широком диапазоне сетей и условий. Однако, канал может состоять из нескольких вспомогательных сетевых соединений на основе протоколов TLS, TCP, DTLS и UDP.

Каждое соединение поддерживается отдельно, если соединение с определенным контроллером или коммутатором прервано, это не приводит к завершению соединения с другими контроллерами или коммутаторами. Несмотря на это, если основное соединение завершено или прервано, все соответствующие ему вспомогательные соединения также завершаются.

При прерывании соединения из-за сетевых условий или таймаута, коммутатор или контроллер, в зависимости от того, кто был инициатором соединения, предпринимает попытки подключения к другой стороне до тех пор, пока не будет установлено новое соединение или пока не будет удален URI соединения из конфигурации. В случае, когда коммутатор теряет контакт с одним

или несколькими контроллерами, он также отправляет сообщение Controller-Status (статус контроллера) всем оставшимся подключенным контроллерам и при восстановлении соединения, отправляет им обновленное сообщение статуса контроллера.

В случае, когда коммутатор теряет контакт со всеми контроллерами, он должен немедленно ввести либо «fail secure mode» (пакеты и сообщения, предназначенные для контроллера, удаляются), либо «fail standalone mode» (обрабатывает все пакеты, используя зарезервированный порт, действует как обычный Ethernet-коммутатор), в зависимости от реализации и конфигурации коммутатора.

## 1.5. Виртуализация сетевых функций

Согласно рекомендации Международного Союза Электросвязи, виртуализация сети – технология, которая позволяет создавать логически изолированные участки сети в рамках совместно используемых физических сетей таким образом, что в этой совместно используемой сети одновременно могут сосуществовать многие виртуальные сети (LINP – Logically Isolated Network Partition, логически изолированный участок сети) [54]. При этом, говоря про виртуализацию сети, стоит отметить такое понятие, как виртуальный ресурс (Virtual resource) – это абстракция физического или логического ресурса, которая может иметь характеристики, отличающиеся от характеристики этого физического или логического ресурса, и ее функциональные возможности могут быть не связаны с функциональными возможностями самого физического или логического ресурса.

Таким образом, виртуализация сетевых функций – технология виртуализации физических сетевых элементов (физических ресурсов) телекоммуникационной сети путем исполнения сетевых функций программными модулями, работающими на стандартных серверах и виртуальных машинах (VM – Virtual Machines) в них. При этом данные программные модули могут

взаимодействовать между собой для предоставления услуг связи, чем ранее занимались аппаратные решения.

Различные типы сетевого оборудования могут быть расположены на стандартных промышленных серверах больших вычислительных мощностей, коммутаторов и систем хранения, которые могут быть расположены в центрах обработки данных (ЦОД), сетевых узлах и в помещениях конечных пользователей. «Виртуализация» включает в себя реализацию сетевых функций в программном обеспечении, которые могут работать в масштабе промышленного сервера и которые могут быть перемещены в различные места в сети по мере требований, без необходимости установки нового оборудования, Рисунок 6 иллюстрирует архитектуру высокого уровня NFV по определению ETSI [34].

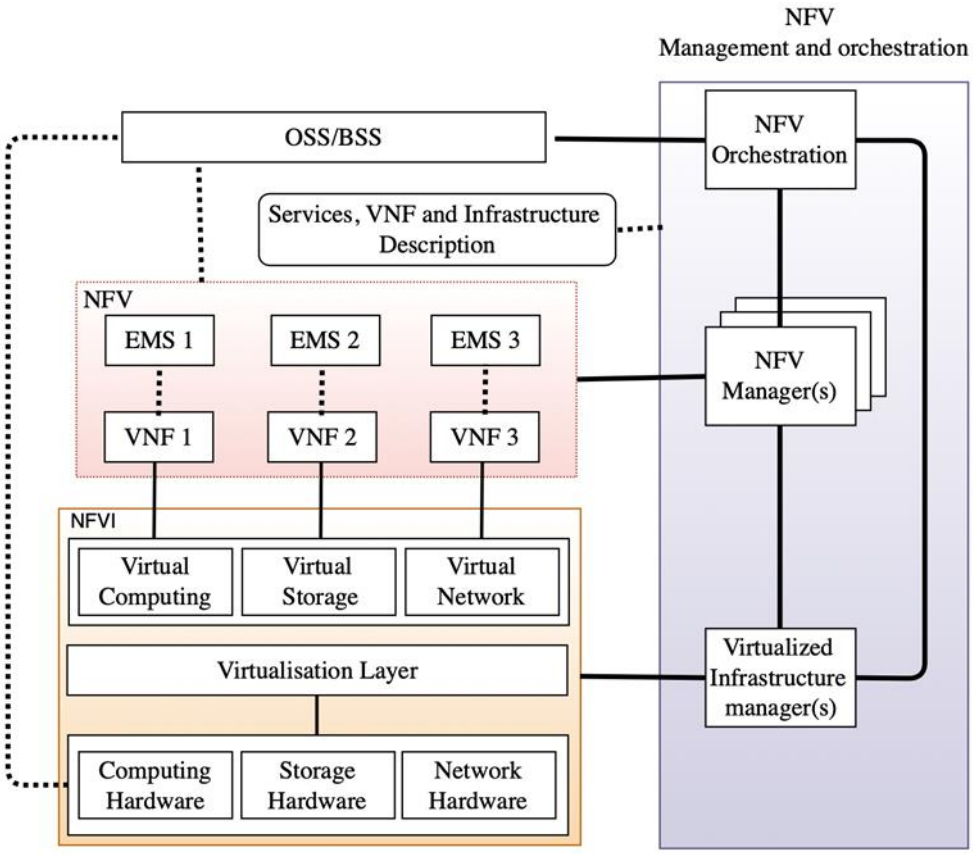


**Рисунок 6** – Архитектура высокого уровня NFV

Архитектура высокого уровня NFV состоит из следующих трех основных функциональных блоков:

1) Virtualized Network Function (VNF) – Виртуальная сетевая функция VNF является программной реализацией сетевой функции, которая обычно реализована

в виде АПК-решения. Например, такие сетевые функции, как: Mobility Management Entity (MME), Serving Gateway (SGW), Packet Data Network Gateway (PGW), Dynamic Host Configuration Protocol server (DHCP-сервер), firewalls, DPI и другие, могут быть реализованы в качестве программной реализации, которые работают на сервере(-ах). Стоит также заметить, что виртуальная сетевая функция может быть развернута как централизовано на одной VM (Virtual Machine, виртуальной машине), так распределено на нескольких виртуальных машинах VM's.VNF также можно разделить на несколько подфункций, называемых компонентами VNF (VNFC) под управлением подсистемы EMS (Elemental Management Systems – EMSs);

2) NFV Infrastructure (NFVI) – Инфраструктура NFV. Инфраструктура NFV включает в себя все аппаратное и программное обеспечение, необходимое для развертывания, эксплуатации и мониторинга виртуальных сетевых функции. С этой целью NFVI имеет уровень виртуализации, необходимый для абстрагирования аппаратных ресурсов (обработка, хранение и сетевое подключение). Это обеспечивает независимость программного обеспечения VNF от физических ресурсов. Уровень виртуализации обычно состоит из серверных (Xen, KVM, VMware и т. Д.) и сетевых (VXLAN, NVGRE, OpenFlow и т. Д.) гипервизоров. С точки зрения непосредственно самих виртуальных сетевых функций, аппаратная часть архитектуры NFV представляется как единая вычислительная платформа, имеющая способность масштабироваться;

3) NFV Management and Orchestration (MANO) – подсистема управления виртуальными сетевыми функциями и оркестрации MANO состоит из трех компонентов:

(1) диспетчер виртуализированной инфраструктуры (VIM), который управляет и контролирует взаимодействие VNF с физическими ресурсами, находящимися под его контролем (например, распределение, освобождение и учет);

(2) подсистема управления виртуальными сетевыми функциями (VNFM), которая отвечает за управление жизненным циклом VNF (например, инициализация, приостановка и завершение);

(3) оркестратор NFV (NFVO), который является подсистемой управления и ответственен за организацию и управление инфраструктурой NFV, программными ресурсами. При этом, одной из главных его функций является – организация и управление услугами на NFVI.

Также стоит отметить элемент системы поддержки операций и системы поддержки бизнеса (OSS/BSS). Этот элемент включает в себя сторонние системы управления и помогает MANO в выполнении сетевых политик, либо автоматически, либо вручную.

## 1.6. Концепция сетевой сегментации в сетях связи

Число устройств и пользователей мобильных связей неуклонно растет, появляются все новые и новые сетевые сервисы, требования абонентов к скорости интернет-доступа вообще и мобильного интернет-доступа в частности становятся жесткими и постоянно увеличиваются, абоненты стали более требовательнее к качеству обслуживания в целом [20; 93]. Разработчики программного и аппаратного обеспечения сетей связи и операторы связи, желая удовлетворить новые вызовы, преобразовывают архитектуру сетей и нормы, стандарты, регламенты взаимодействия. Так, в последнее время появились сети пятого поколения (5G/IMT-2020 – 5th Generation Mobile Network), которые являются серьезным продолжением эволюции сетей четвертого поколения (LTE, 4G) [14; 53; 55].

В сетях связи 5G/IMT-2020 важнейшей концепцией является сетевая сегментация [17; 12; 42; 70; 74; 90; 94]. Сетевая сегментация является одной из главных компонентов новых технологий эффективного использования сетевых инфраструктурных ресурсов. Согласно [42] сетевая сегментация определяется как концепция параллельного развертывания нескольких логических, автономных и независимых сетей на общей инфраструктурной платформе. В принципе, сетевая сегментация – это специфический способ виртуализации сетевых

инфраструктурных ресурсов; это инструмент для совместного использования сетевых ресурсов и предоставления настраиваемых сетевых архитектур для различных вариантов использования, которые используют одну и ту же базовую физическую инфраструктуру [80]. Каждая такая виртуальная сеть представляет собой сетевой ссегмент, имеющий свой тип трафика, и в котором может использоваться своя технология передачи данных. Гибкость сетевой сегменации позволяет удовлетворить самые разнообразные и даже противоречивые требования абонентов [70; 94]. Благодаря сетевой сегментации операторы связи могут проводить изоляцию сетевых ресурсов «по требованию» на базе совместно используемой физической инфраструктуры [31; 38; 90; 95].

Таким образом, в сети на базе сегментов, каждый сегмент представляет собой отдельную логическую сеть, настроенную на определенные (минимум один) сервисы [11; 31; 17; 68]. Сетевой сегмент относится к управляемым разделам физических и/или виртуальных сетевых ресурсов; к физическим, виртуальным и сервисным сетевым функциям; которые могут выступать в качестве независимого экземпляра сети и/или как сетевое облако. К сетевым ресурсам относятся ресурсы связи (каналы связи, телекоммуникационное оборудование и т.д.), вычислительные ресурсы и ресурсы хранения.

Согласно [54], реализации сетевой сегментации в сетях связи 5G/IMT-2020 – преимущественно для сетевых администраторов и дизайнеров, благодаря в основном следующим наборам функций:

1) изоляция сетевых сегментов: полная изоляция сетевого сегмента дает возможности управлять параллельными независимыми сетевыми сегментами. В результате чего сетевые сбои, перегрузки или другие угрозы безопасности в одном сегменте не влияют на работу других в масштабах сети. Более того, каждый сетевой сегмент должен иметь независимые функции обеспечения безопасности, которые приостанавливают неавторизованные попытки доступа на чтение или запись информации о конфигурации, управлении или учету;

2) гибкая виртуализация сетевых функций: в отличие от традиционных сотовых связей, в которых все сервисы состоят из одних и тех же функций, с

помощью технологии сегментации каждая служба может зависеть от других функций;

3) упрощенные сервисные цепочки в домене виртуализации: технология NFV позволяет использовать сетевые функции в независимости от их физического размещения. Однако оптимальное размещение сетевых функций улучшает возможности сети;

4) прозрачное управление сетевыми сегментами: благодаря уровню абстрагирования физических сетевых ресурсов, технология сегментации в сетях связи позволяет устанавливать сетевые сегменты для сетей разных доменов.


**Рисунок 7** – Сетевые виртуальные функции в сетях связи 5G/IMT-2020 и соответствующие им требования для сетевых сегментов

Один сетевой сегмент специально ориентирован на обеспечение сверхмалой задержки и высокой надежности (например, автономные транспортные средства). Другой сетевой сегмент, например, предназначен для устройств, с низким энергопотреблением (например, датчики), а также с малой емкостью батареи, а другой, к примеру, для сервисов, требующих сверхвысокие скорости.

### *1.6.1. Аналитический обзор международной деятельности по исследованию сетевой сегментации*

Процесс стандартизации сетевой сегментации, на данный момент, все еще находится на начальной стадии, и все работы в основном сосредоточены на вертикальной сетевой сегментации. Существует множество исследований, посвященных сетевой сегментации, в рамках различных исследовательских проектов, включая NGMN, 5G NORMA, WWRF, 3GPP и 5GPPP, IETF, GSMA, ONF и т. д. Во всех этих проектах сетевая сегментация считается одной из основных технологий для сетей связи 5G/IMT-2020. Большинство исследований и разработок по сетевой сегментации рассматривают, в основном, вопросы концепции, архитектуры системы, требования и влияния сегментации на сетевую архитектуру.

В качестве исходных документов для анализа составили документы Международных организаций по стандартизации и программ таких как ITU-T (International Telecommunication Union – Telecommunication Standardization Sector) [53–55], ETSI (European Telecommunications Standards Institute) [32–35], NGMN (Next Generation Mobile Networks) [79–81], 3GPP (3rd Generation Partnership Project) [11–13], 5G PPP (5G Infrastructure Public Private Partnership) [14; 15], IEEE (Institute of Electrical and Electronics Engineers) [49], IETF (Internet Engineering Task Force) [50; 51], GSMA (GSM Association) [44; 45], ONF [83–88] и др., в которых даны теоретические концепции, методы и модели сетевой сегментации.

### *1.6.2. Деятельность по разработке стандартов по сетевой сегментации*

Рассмотрим различные исследовательские группы и организации, имеющие отношение к разработке документов по сетевой сегментации:

**3GPP.** 3GPP занимается стандартизацией сетевой сегментации применительно сетям мобильной связи [13] и содержит различные рабочие группы, связанные с сетевой сегментацией: 3GPP SA1, в которой рассматриваются варианты использования и требования сетевой сегментации, 3GPP RAN1/2/3 рассматривает возможности распознавания RAN сенментов, а в 3GPP SA2 рассматривает фундаментальные архитектуры поддержки сетевой сегментации. Рабочая группа 3GPP SA5 занимается вопросами создания и управления сетевых

сегментов в 3GPP и за координацию с другими соответствующими группами по стандартизации. Рабочая группа 3GPP SA3 отвечает за возможности обеспечения безопасности сетевых сегментов, которые требуют координацию с ETSI ISG NFV для изоляции сетевых сегментов.

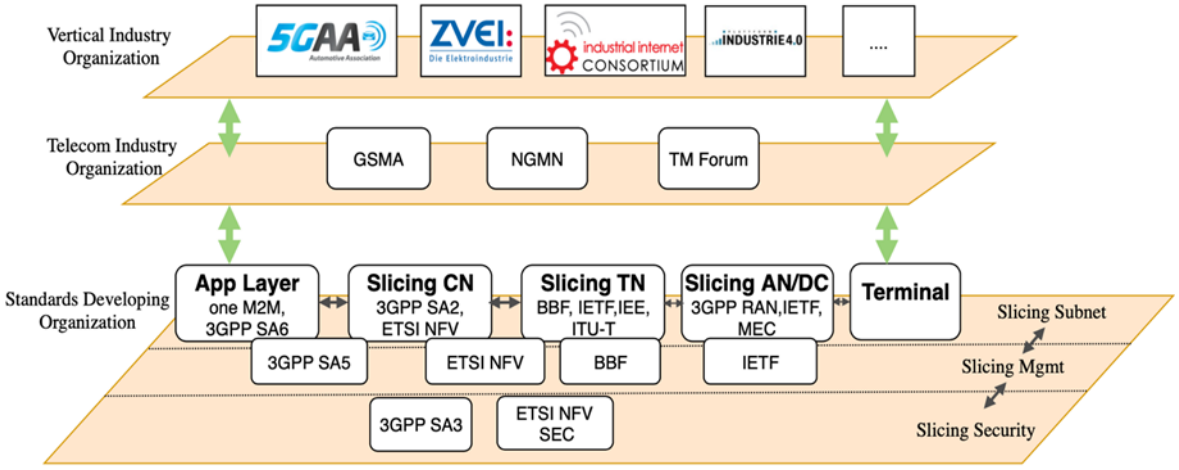


**Рисунок 8 –** Организации по разработке стандартов сетевой сегемнтации и некоторые другие отраслевые организации

**NGMN.** Альянс NGMN определил основное описание концепции сетевой сегментации [79; 80]. 5G/IMT-2020 сетевой сегмент по NGMN – это коллекция набора 5G/IMT-2020 сетевых функций и специальных настроек технологии радиодоступа, которые объединяются для конкретного варианта использования или бизнес-модели, используя концепции виртуализации сетевых функций (NFV) и программно-конфигурируемых сетей (SDN). Концепция сетевой сегментации организована многоуровневым образом, дифференцируя разные уровни сетевого экземпляра: уровень службы, включающий конечного пользователя бизнес-услуг; уровень сетевого сегмента (network slice instance – NSI) как набор функций, образующих конкретную логическую сеть; и уровень ресурсов, состоящий из физических и логических ресурсов. В этом многоуровневом представлении NSI могут потенциально использоваться несколькими экземплярами службы.

**ITU-T.** Сектор стандартизации международного союза электросвязи (ITU-T) рассматривает роль сетевых сегментов в рамках рассмотрения концепции 5G/IMT-2020. ITU-T разработкой рекомендаций по этому направлению занимаются исследовательские комиссии ИК13 и ИК15 и рассматривают вопросы управления

и транспорта; интеграция с 5G/IMT-2020. В [54] ITU-T определил концепцию LINP, состоящую из нескольких виртуальных ресурсов, которая фактически является реализацией сетевой сегментации. LINPs изолированы от других LINPs, имеющих собственную программируемую плоскость управления и плоскость передачи данных. В Рекомендации ITU-T IMT-2020 [55] определяются вертикальный и горизонтальный сетевый сегменты с независимым управлением каждой плоскости.

**ONF, BBF, ETSI-NFV.** ETSI-NFV анализирует варианты использования сетевой сегментации, определенные в других организациях по стандартизации и отраслевых форумах по поддержке сетевой сегментации с ETSI-NFV Architecture Framework [32], и дает описание возможности использования сетевых сегментов в концепциях SDN и NFV. Две основные рабочие группы по изучению концепции сетевой сегментации: ETSI-NFV ISG отвечает за предоставление технических решений по сетевым ресурсам, такие как вычисления и хранение для сетевых сегментов и ETSI ISG ZSM, целью которой является решение проблемы управления сетевыми сегментами. В группе ETSI MEC новое подразделение «Поддержка MEC для сетевой сегментации» призван определить необходимую поддержку сетевой сегментации, оценить пробелы в функциях и возможностях MEC, и определить новые требования. BBF [22] также занимается вопросом сетевой сегментации, дополняя предыдущие функции управления, определяя новые и дополнительные функции, такие как управление сетевыми сегментами сетей доступа (Access Network Slice Management – ANSM), управление сетевыми сегментами базовой сети (Core Network Slice Management – CNSM) и управление сетевыми сегментами транспортной сети (Transport Network Slice Management – TNSM). Каждый из них предназначен для управления жизненным циклом сетевого сегмента каждой конкретной сетевой иерархии (сеть доступа, базовая или транспортная). ONF [84; 86] рассматривает возможности SDN для сетевой сегментации.

**IETF.** IETF рассматривает сетевую сегментацию с точки зрения спецификации терминологии, архитектуры, вариантов использования, постановки задачи и других аспектов, связанных с сетевой сегментацией [50, 51]. Чтобы

отобразить пробелы между технологически независимыми требованиями к сервису сетевой сегментации и конкретными технологическими реализациями, создается независимая от технологии информационная модель.

**GSMA.** В [44] GSMA определяет концепцию сетевой сегментации с точки зрения модели бизнес-клиентов, согласно которой, сетевая сегментация является воплощением концепции запуска нескольких логических сетей как практически независимых бизнес-операций на общей физической инфраструктуре эффективным и экономичным способом. GSMA определяет, что настраиваемые сетевые возможности включают скорость передачи данных, качество, задержку, надежность, безопасность и услуги. Также упоминается, что разные операторы могут совместно использовать один и тот же сетевой сегмент в описании GSMA.

## Выводы по главе 1

1. Проведен анализ основных требований к будущим сетям связи, в частности, к сетям пятого поколения 5G/IMT-2020.

2. Показано, что одна из задач будущих сетей связи состоит в представлении сетевых услуг по запросу приложений или пользователей.

3. Проведен анализ архитектур построения программно-конфигурируемых сетей, обозначены концепции и разработки, способствовавшие её появлению и последующему развитию.

4. Проведен анализ сетевой сегментации как архитектурного решения организации сетевой инфраструктуры для оптимизации сетевых ресурсов, сетевых функции и сетевых сервисов.

# Глава 2. Разработка моделей оценки показателей эффективности функционирования программно-конфигурируемых сетей

## 2.1. Коммутатор OpenFlow

Как было рассмотрено ранее, OpenFlow – открытый стандарт, в котором описываются требования, предъявляемые к коммутатору, поддерживающему протокол OpenFlow для удаленного управления [83]. Согласно спецификации стандарта, OpenFlow [85], взаимодействие контроллера с коммутатором осуществляется посредством протокола OpenFlow. Каждый коммутатор должен содержать одну или более таблиц потоков (flow tables), групповую таблицу (group table) и поддерживать канал (OpenFlow channel) для связи с удаленным контроллером. Каждая таблица потоков в коммутаторе содержит набор записей (flow entries) о потоках или правила. Каждая такая запись состоит из полей-признаков (match fields), счетчиков (counters) и набора инструкций (instructions). Управление данными в OpenFlow коммутаторе осуществляется не на уровне отдельных сообщений, а на уровне их потоков. Правила в коммутаторе устанавливаются с участием контроллера только для первого пакета, а затем все остальные пакеты потока его используют.

## 2.2. Принципы функционирования программно-конфигурируемой сети

При развертывании ПКС контроллер обычно управляет несколькими OpenFlow коммутаторами, соединяющими группу хостов. Типичная архитектура ПКС показана на Рисунке 9. OpenFlow коммутатор проверяет таблицу потоков на прибытие пакета. Если пакет найден, коммутатор применяет действие к пакету, указанное в записи таблицы, как правило, перенаправляя его на указанный интерфейс. В противном случае пакет должен принадлежать к новом потоку, и коммутатор пересылает его ПКС-контроллеру в сообщение packet-in. Затем контроллер определяет соответствующее правило потока и посылает его коммутатору в сообщение *packet-out* или *flow_mod*. Как следствие, ПКС-контроллер получает поток сообщение *packet-in* от каждого OpenFlow коммутатора (Рисунок 10).

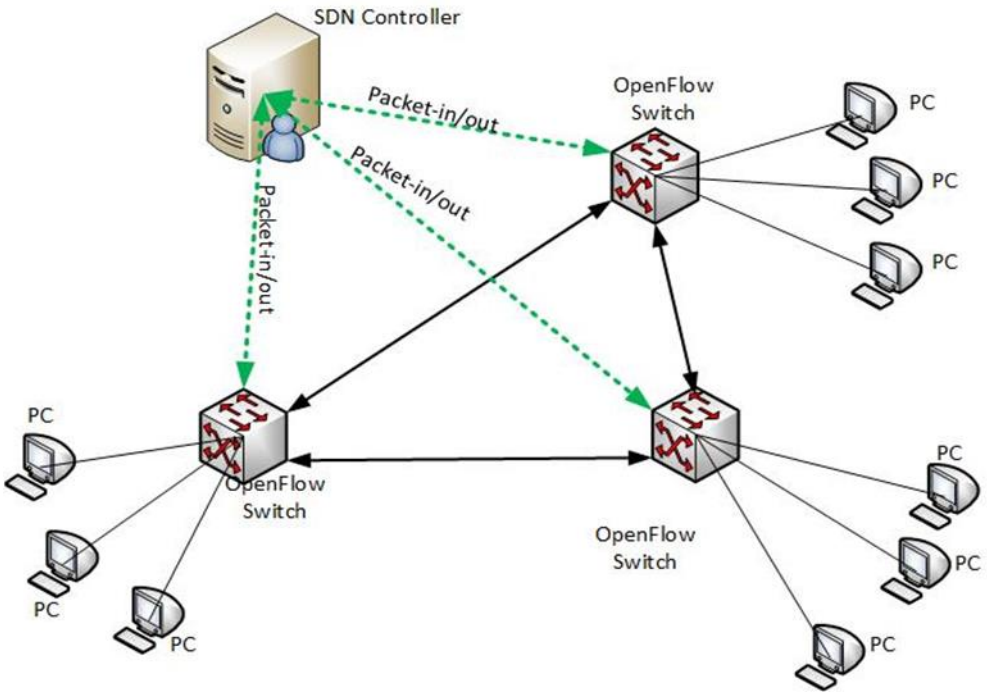


**Рисунок 9** – Архитектура программно-конфигурируемых сетей

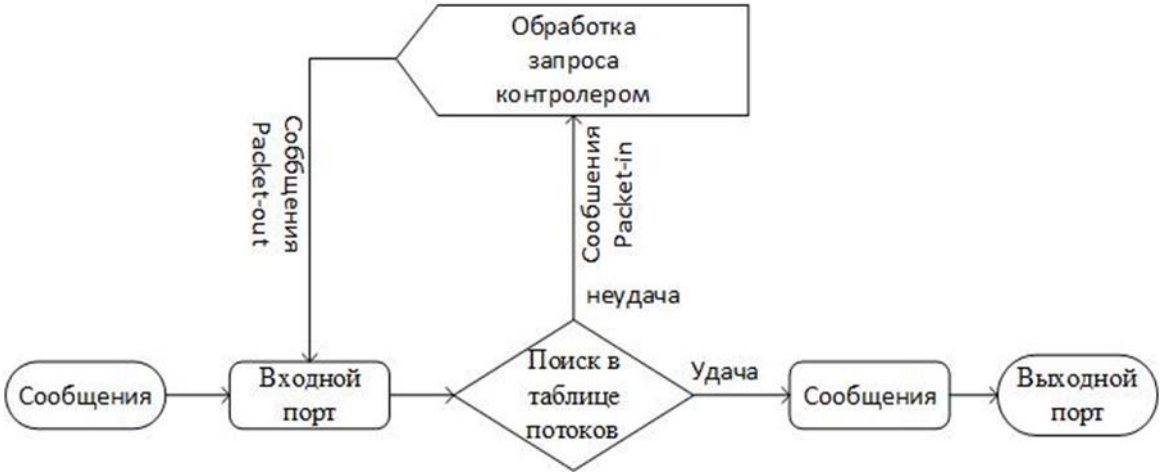


**Рисунок 10** – Процесс передачи сообщения через OpenFlow коммутатор

В современных сетях связи одновременно передается информация разных видов (видео и аудиоинформация, сжатое видео и аудиоинформация, а также данные, менее чувствительные к задержкам) с разными показателями качества обслуживания, на ее работу существенно влияют методы управления трафиком [8; 102].

## 2.3. Оценка производительности ПКС-контроллеров

В частности, при развертывании ПКС с одним контроллером, контроллер остается фундаментальной частью архитектуры и критической точкой для успешной работы, поскольку все решения, касающиеся работы сети, принимаются

единственным контроллером. Соответственно много исследовательских работ сосредоточено на улучшение производительности контроллеров и их способности управлять очень большим количеством сетевых устройств [2; 29; 74]. Два важных показателя используются для характеристики производительности контроллера: время, необходимое для установки нового правила в коммутаторах (задержка), и количество запросов, обрабатываемых в секунду (пропускная способность). К примеру, NOX - первый контроллер, разработанный для управления ПКС, имеет пропускную способность 30000 потоков в секунду и задержку 10 мс [96]. Впоследствии применение нескольких контроллеров (более 30) были предложены промышленностью и лабораториями для создание глобальных сетей на основе ПКС.

## 2.4. Тестирование контроллера программно-конфигурируемой сети на базе разработанной методики испытаний

### *2.4.1. Цель эксперимента*

Цель эксперимента – исследовать поведение ПКС-контроллера и его аппаратных характеристик, при тестировании контроллера различным программным обеспечением; оценить эффективность проведения такого тестирования.

### *2.4.2. Задачи эксперимента*

Для достижения поставленной цели решаются следующие задачи:

1) установить и запустить контроллер ПКС, на базе аппаратной платформы;

2) установить необходимое программное обеспечение для проведения тестирования;

3) провести тестирование – измерение характеристик контроллера ПКС;

4) обработать полученные результаты и сделать выводы.

### *2.4.3. Структура лабораторного стенда и модельной сети*

Для проведения эксперимента по тестированию взаимодействия программного и аппаратного обеспечения контроллера программно-

конфигурируемой сети, на базе лаборатории «Программируемых сетей» кафедры Сетей связи и передачи данных (СС и ПД), была выполнена сборка и настройка лабораторного стенда. Структурная схема лабораторного стенда представлена на Рисунке 11.

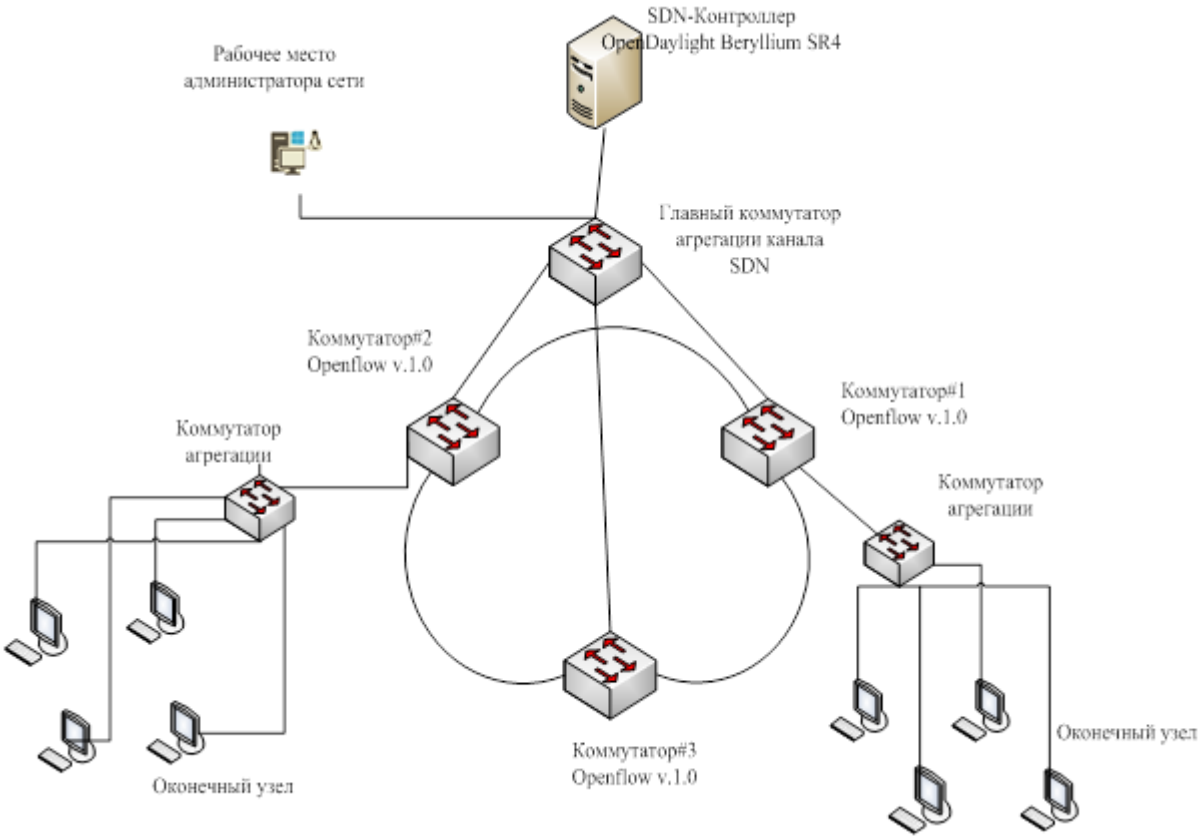


**Рисунок 11** – Структурная схема лабораторного стенда и модельной сети

Лабораторный стенд и модельная сеть состояли из:

– ПКС-контроллера, с операционной системой, которая является главным управляющим элементом Коммутаторов с поддержкой протокола OpenFlow v1.0 (OpenFlow-коммутаторы), которые выполняют функции коммутации, передачи трафика и подключения оконечных узлов;

– коммутаторов агрегации, которые осуществляют подключение оконечных узлов к ПКС;

– главного коммутатора агрегации, к которому подключаются OpenFlow-коммутаторы;

– оконечных узлов, на которые устанавливается программное обеспечение для тестирования ПКС-контроллера;

– рабочего места администратора сети, для мониторинга и исправления нестабильно работающих элементов сети и самой сети;

– соединительных кабелей стандарта Gigabit Ethernet 1000BASE-TX.

## 2.5. Алгоритмы тестирования контроллера ПКС

В качестве тестируемого ПКС-контроллера и его аппаратной платформы были выбраны:

– контроллер OpenDaylight Beryllium SR 4, который является программным обеспечением;

– программно-аппаратный комплекс с ОС Ubuntu 16.04 LTS, ЦП Intel Xeon(R) CPU E3-1220 v2 @ 3.10 (Ivy) ГГц (4 ядра), RAM=15,6 Гб, HDD=1Тб MotherBoard ASUSTek COMPUTER INC.: P8Z77-V PRO.

### *2.5.1. Алгоритм тестирования ПКС-контроллера утилитой Cbench*

1. На оконечный узел устанавливается ОС Ubuntu 16.04.LTS, программное обеспечение Open vSwitch и утилита Cbench.

2. Согласно пунктам 1–4 «Этапа 1. Разработка сценариев испытаний контроллера», запускается тестирование.

3. После прохождения тестирования, проводится обработка и анализ полученных результатов.

### *2.5.2. Алгоритм тестирования ПКС-контроллера программным обеспечением ProLan Qutester Plus*

1. На оконечный узел устанавливается программное обеспечение ProLan Qutester Plus.

2. Согласно пунктам 1–4 «Этап 1. Разработка сценариев испытаний контроллера, пункт 5», запускается тестирование.

3. После прохождения тестирования, проводится обработка и анализ полученных результатов.

### *2.5.3. Алгоритм тестирования ПКС-контроллера программным обеспечением ОССТ*

1. На оконечный узел устанавливается программное обеспечение OCCT. Выбирается метод тестирования CPU: OCCT. Задаются параметры:

– тип тестирования – «Авто»,

– длительность – 20 часов,

– версия теста – 64 бит,

– режим тестирования – средний набор данных,

– Number of Threads – «Авто».

2. Согласно пунктам 1–4 «Этапа 1. Разработка сценариев испытаний контроллера, пункт 6», запускается тестирование.

3. После прохождения тестирования, проводится обработка и анализ полученных результатов.

#### *2.5.4. Алгоритм тестирования ПКС-контроллера программным обеспечением Mininet*

1. На оконечный узел устанавливается программное обеспечение Mininet и утилита для тестирования, разработанная в лаборатории. Программа работает по принципу изменения соединений OpenFlow коммутаторов между собой. Используя возможности программного обеспечения Mininet возмоно создавать нагрузку на контроллер изменением топологии заданной сети и измерять его ключевые характеристики.

2. Согласно пунктам 1–4 «Этапа 1. Разработка сценариев испытаний контроллера, пункт 7», запускается тестирование.

3. После прохождения тестирования, проводится обработка и анализ полученных результатов.

### 2.6. Результаты тестирования

#### *2.6.1. Результаты тестирования ПКС-контроллера утилитой Cbench*

Измерение пропускной способности контроллера в режиме нагрузочного тестирования, в зависимости от количества управляемых контроллером коммутаторов (1, 2, 4, 8, 16, 32, 64, 128), при фиксированном количестве оконечных узлов (1000000). Тестирование проводится 10 раз, с периодом одного теста 90

секунд. Тестирование проводилось при 4 активных ядрах центрального процессора (ЦП). Результаты представлены на Рисунке 12.


**Рисунок 12** – Зависимость пропускной способности от числа коммутаторов, при постоянном количестве оконечных узлов (1000000)

Измерение пропускной способности в режиме нагрузочного тестирования, в зависимости от количества подключенных оконечных узлов (10, 100, 1000, 10000, 100000, 1000000), при фиксированном количестве коммутаторов (32). Тестирование проводится 10 раз, с периодом одного теста 90 секунд. Тестирование проводилось при 4 активных ядрах ЦП. Результаты представлены на Рисунке 13.

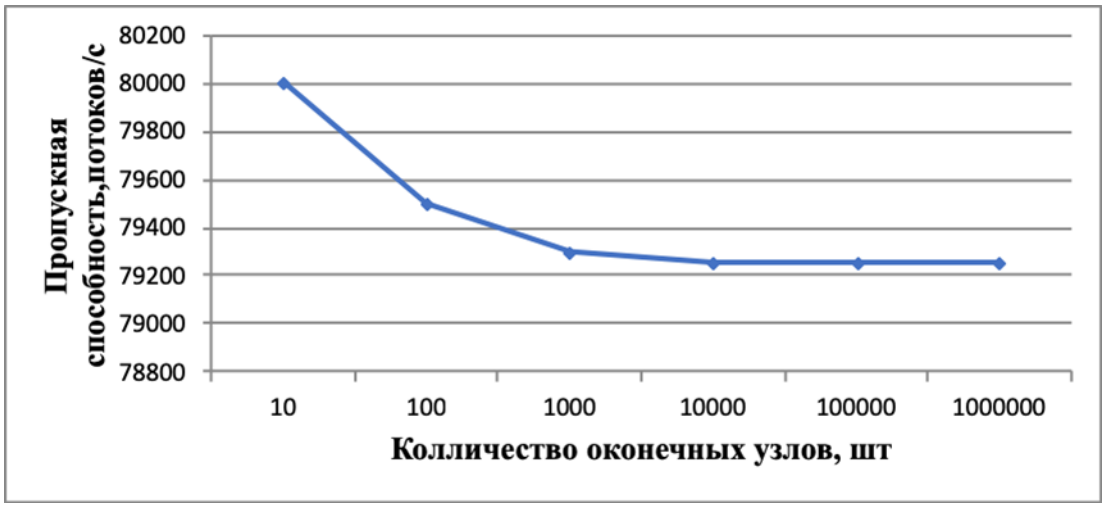


**Рисунок 13** – Зависимость пропускной способности от числа оконечных узлов, при постоянном количестве коммутаторов

Измерение времени обработки одного запроса (задержки) контроллера в режиме нагрузочного тестирования, в зависимости от количества управляемых контроллером коммутаторов (1, 2, 4, 8, 16, 32, 64, 128). Тестирование проводится 10 раз, с периодом одного теста 90 секунд. Тестирование проводилось при 4 активных ядрах ЦП. Результаты представлены на Рисунке 14.

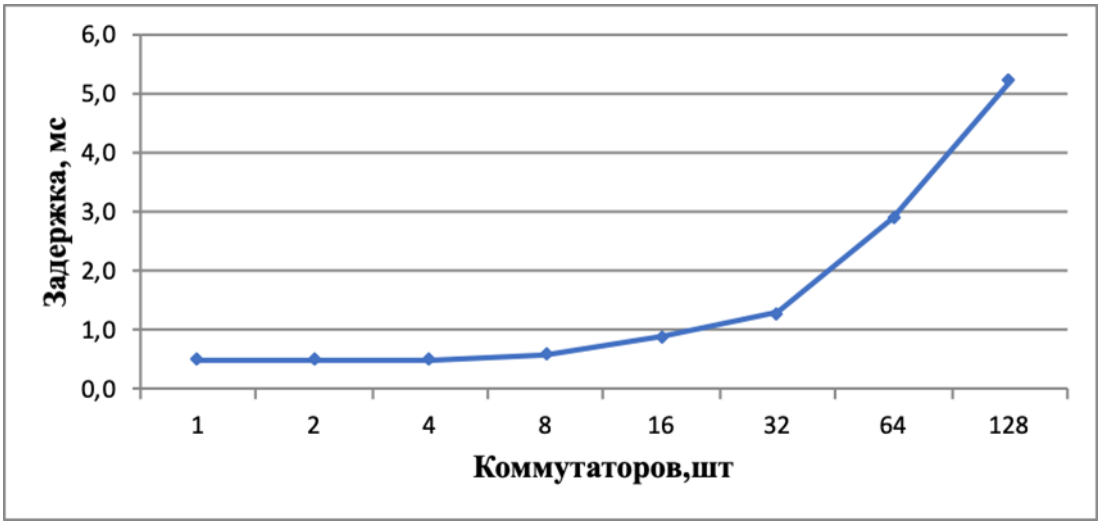


**Рисунок 14** – Зависимость времени обработки одного запроса (задержки) контроллера в режиме нагрузочного тестирования от количества управляемых контроллером коммутаторов

Измерение времени обработки одного запроса (задержки) контроллера в режиме нагрузочного тестирования в зависимости от количества подключенных оконечных узлов (10, 100, 1000, 10000, 100000, 1000000), при фиксированном количестве управляемых контроллером коммутаторов (32). Тестирование проводится 10 раз, с периодом одного теста 90 секунд. Тестирование проводилось при 4 активных ядрах ЦП. Результаты представлены на Рисунке 15.

### *2.6.2. Результаты тестирования ПКС-контроллера программным обеспечением ProLan QuTester Plus*

Исследование масштабируемости контроллера. Измерение производительности контроллера в режиме стрессового тестирования, при фиксированном количестве управляемых контроллером коммутаторов (256) и оконечных узлов (1000000). На каждой конфигурации проводилось 10 тестов продолжительностью 90 секунд каждый, после чего высчитывалось среднее

значение измеряемого параметра. Тестирование проводилось при 4 активных ядрах ЦП. Результат тестирования приведен в Таблице 2.

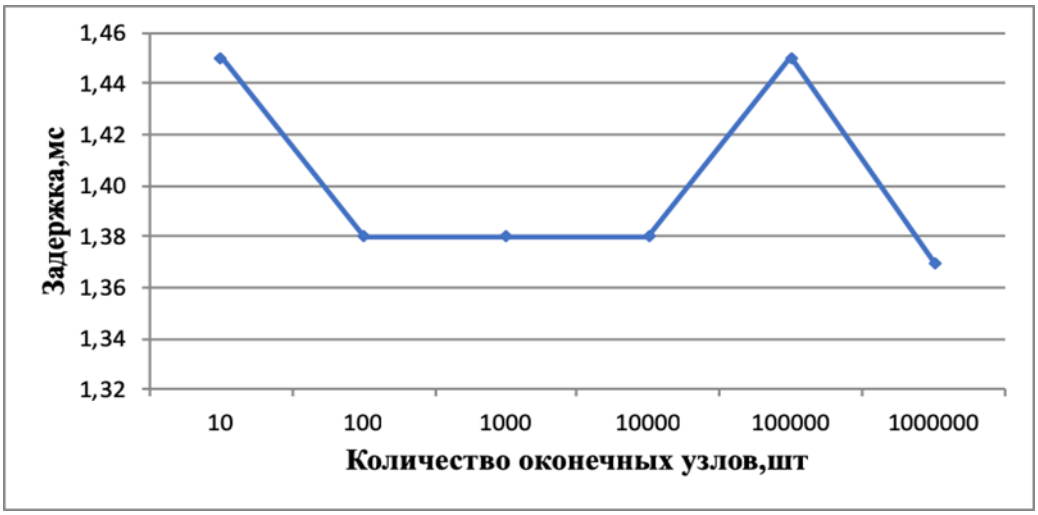


**Рисунок 15** – Зависимость времени обработки одного запроса (задержки)

**Таблица 2** – Результат исследования масштабируемости контроллера

| *Оценка качества (цвет «светофора»)* | *Измеряемый параметр (eth.100)* | | *Пороговое значение (eth.100)* | |
|---|---|---|---|---|
| | Read Rate (Kbyte/sec) | Write Rate (Kbyte/sec) | Read Rate (Kbyte/sec) | Write Rate (Kbyte/sec) |
| Хорошо («зеленый») | 2320 | 1802 | 2298 | 1789 |
| Хорошо («зеленый») | 2315 | 1798 | 2298 | 1789 |
| Хорошо («зеленый») | 2325 | 1795 | 2298 | 1789 |
| Допустимо («мигающий желтый») | 2326 | 1536 | 2298 | 1130–1789 |
| Хорошо («зеленый») | 2295 | 1793 | 2298 | 1789 |
| Допустимо («мигающий желтый») | 2290 | 1530 | 1789–2298 | 1789 |
| Хорошо («зеленый») | 2329 | 1799 | 2298 | 1789 |
| Хорошо («зеленый») | 2330 | 1802 | 2298 | 1789 |
| Хорошо («зеленый») | 2325 | 1806 | 2298 | 1789 |
| Хорошо («зеленый») | 2320 | 1801 | 2298 | 1789 |

### *2.6.3. Результаты тестирования ПКС-контроллера программным обеспечением ОССТ*

Тестирование стабильности работы контроллера. Измерение производительности, объема занимаемой оперативной памяти, температуры каждого ядра ЦП контроллера в режиме нагрузочного тестирования, при

фиксированном количестве управляемых ПКС-контроллером коммутаторов (32) и оконечных узлов (1000000), на временном интервале t = 24 часа (86400 секунд). Тестирование проводилось при 4 активных ядрах ЦП. Результаты тестирования приведены на рисунках ниже.

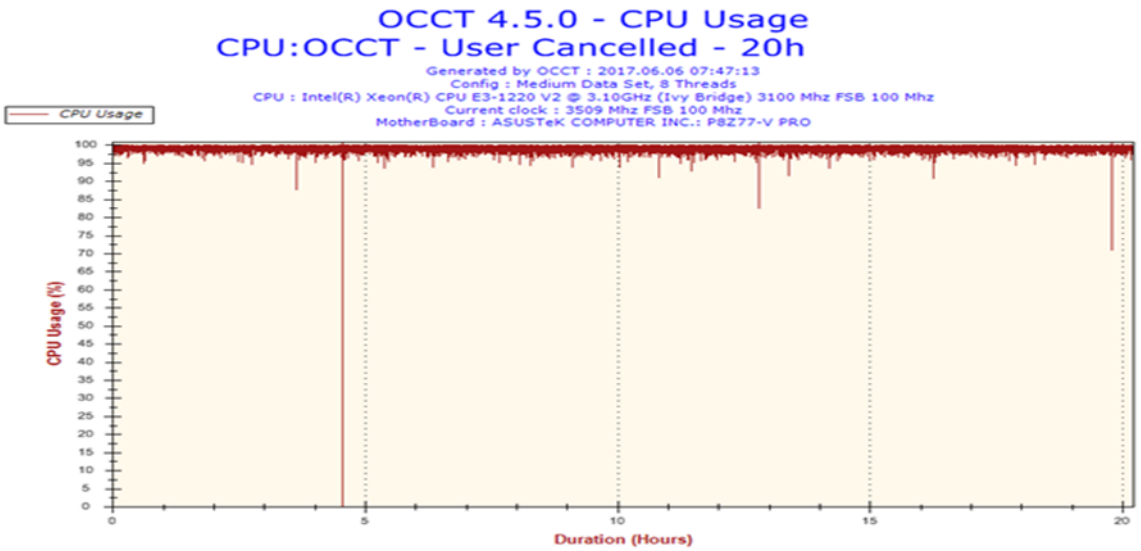


**Рисунок 16** – Тестирование производительности центрального процессора ПКС-контроллера

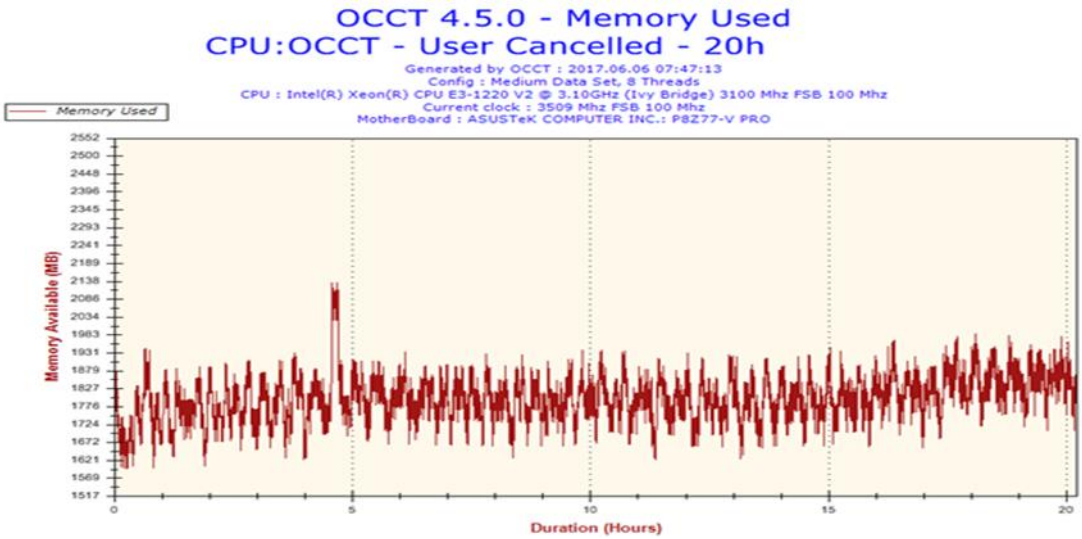


**Рисунок 17** – Использование оперативной памяти ПКС-контроллера

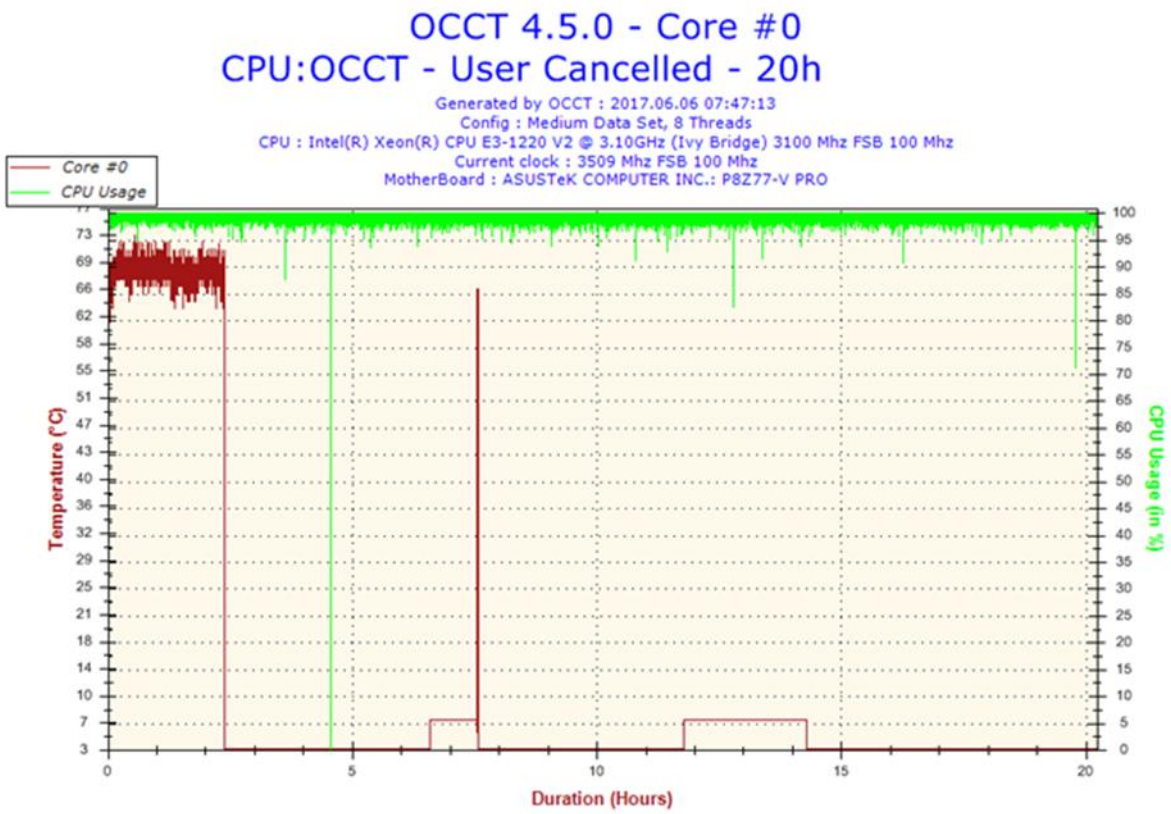


**Рисунок 18** – Температура и производительность на первом ядре (core#0) центрального процессора ПКС-контроллера

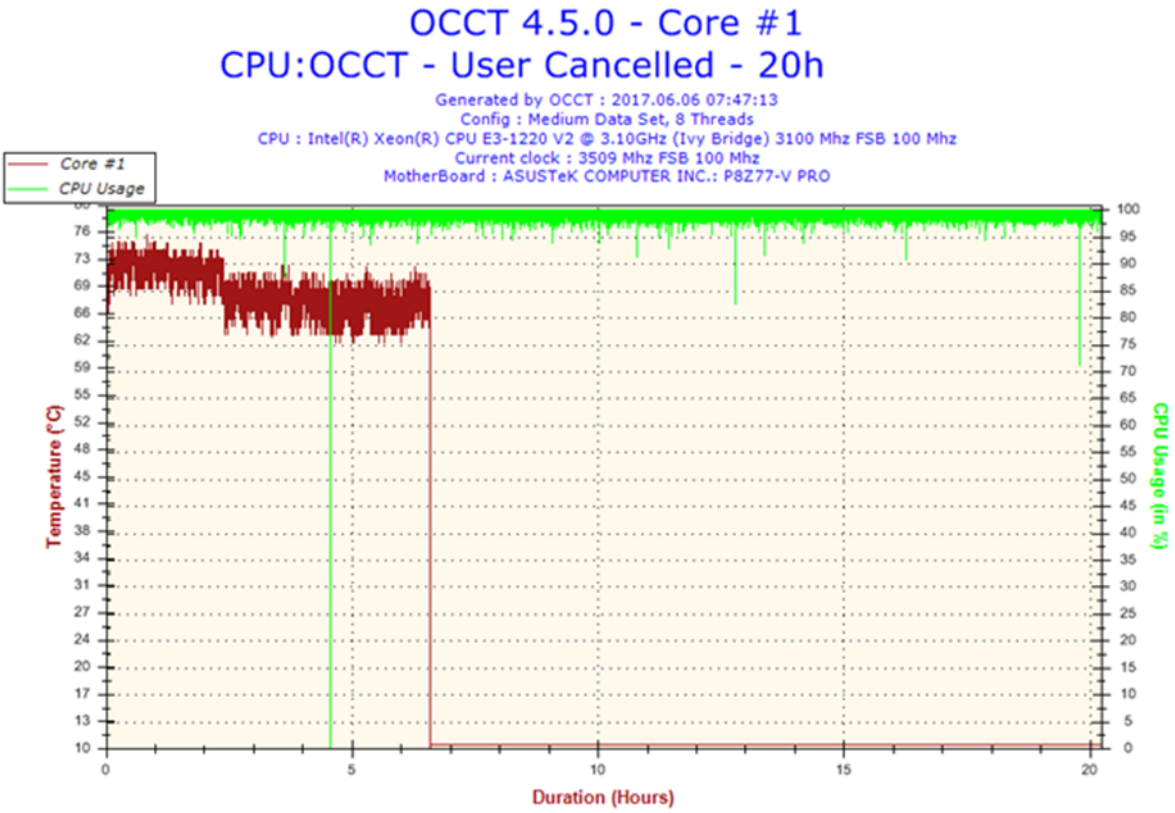


**Рисунок 19** – Температура и производительность на втором ядре (core#1) центрального процессора ПКС-контроллера

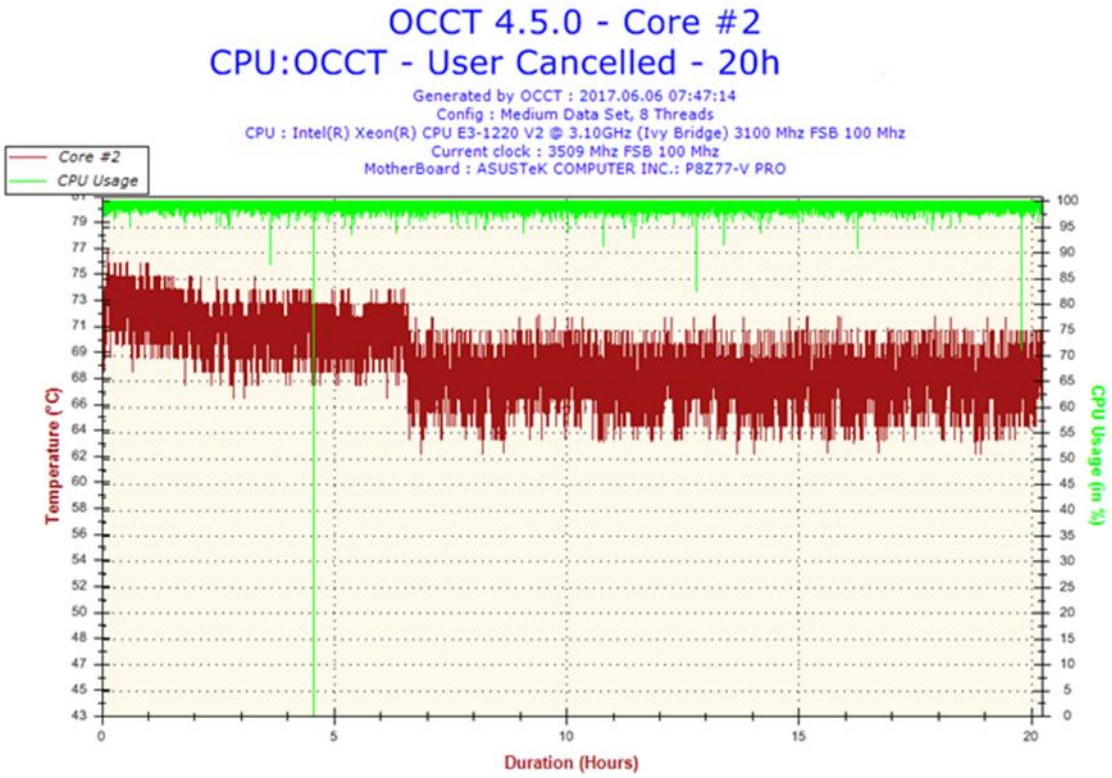


**Рисунок 20** – Температура и производительность на третьем ядре(core#2) центрального процессора ПКС-контроллера

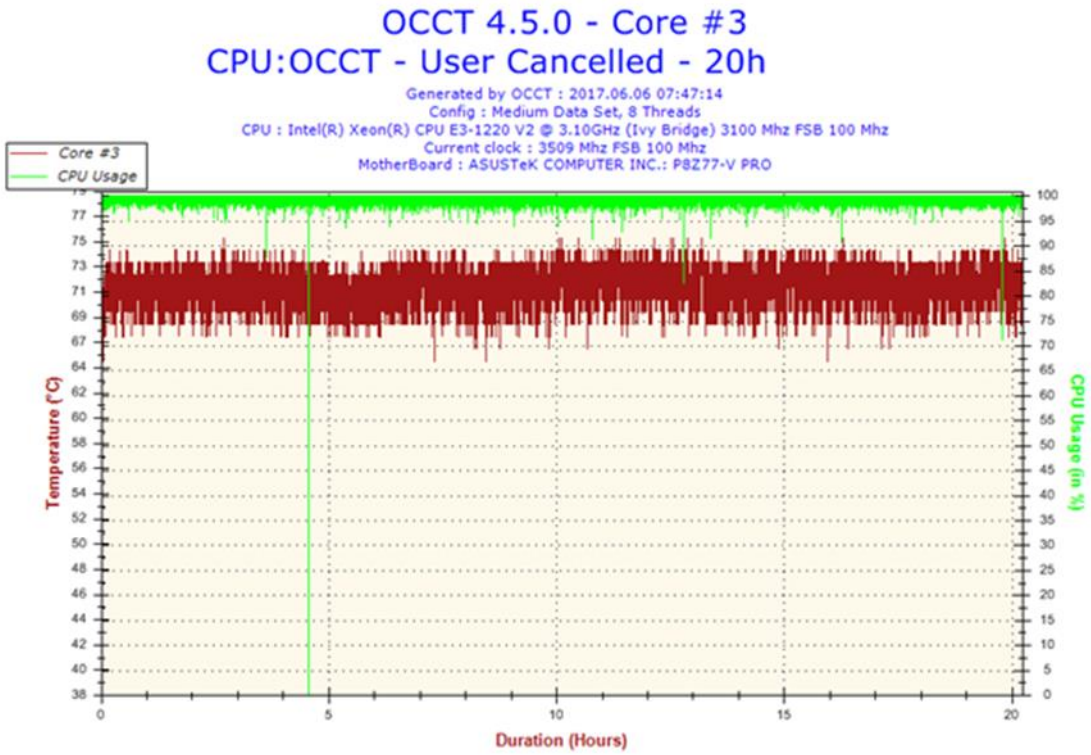


**Рисунок 21** – Температура и производительность на четвертом ядре (core#3) центрального процессора ПКС-контроллера

### *2.6.4. Результаты тестирования ПКС-контроллера программным обеспечением Mininet*

Выполнялось измерение производительности контроллера, в зависимости от времени изменения заданной топологии сети на контроллере (10; 1; 0,1 секунды), при соответствующем параметре количества измерений.

На каждой топологии сети проводилось 10 тестов продолжительностью 90 секунд каждый, после чего высчитывалось среднее значение исследуемого параметра. Тестирование проводилось при 4 активных ядрах центрального процессора. Результаты тестирования топологии сети, представленной на Рисунке 22 на контроллере, находятся на рисунках ниже.

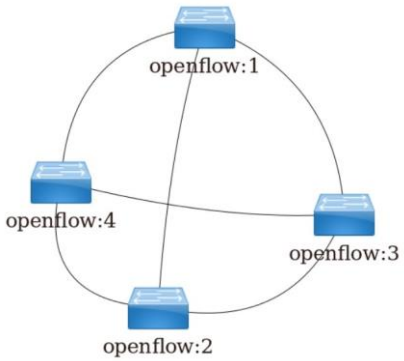


**Рисунок 22** – Топология сети на контроллере

- Время изменения соединений двух коммутаторов t = 10 секунд. На Рисунках 23–24 представлены результаты зависимости нагрузки на процессор ПКС-контроллера (%) и объем занимаемой оперативной памяти (%) от времени проведения тестирования (секунды).


**Рисунок 23** – Зависимость нагрузки на процессор от времени тестирования

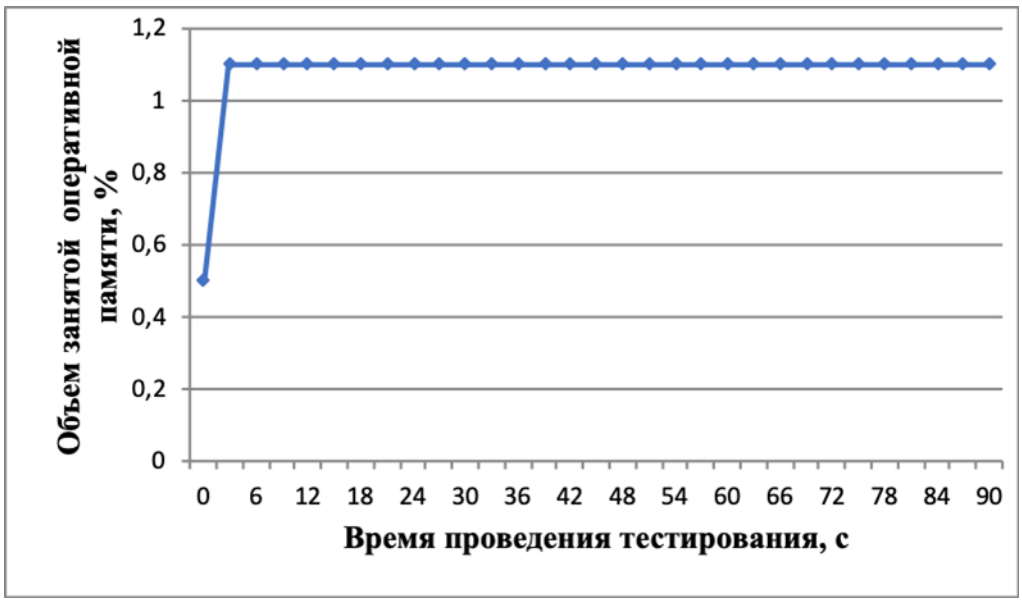


**Рисунок 24** – Зависимость объема оперативной памяти от времени тестирования

- Время изменения соединений двух коммутаторов t = 1 секунда. На Рисунках 25–26 представлены результаты зависимости нагрузки на процессор ПКС-контроллера (%) и объем занимаемой оперативной памяти (%) от времени проведения тестирования (секунды).


**Рисунок 25** – Зависимость нагрузки на процессор от времени тестирования


**Рисунок 26** – Зависимость объема оперативной памяти от времени тестирования

- Время изменения соединений двух коммутаторов t = 0,1 секунды. На Рисунках 27–28 представлены результаты зависимости нагрузки на процессор ПКС-контроллера (%) и объем занимаемой оперативной памяти (%) от времени проведения тестирования (секунды).


**Рисунок 27** – Зависимость нагрузки на процессор от времени тестирования


**Рисунок 28** – Зависимость объема оперативной памяти от времени тестирования

### *2.6.5. Выводы по результатам тестирования*

1. *Выводы по результатам тестирования ПКС-контроллера утилитой Cbench*

Проведен анализ полученных результатов тестирования ПКС-контроллера утилитой Cbench. Было выявлено, что производительность контроллера OpenDaylight Beryllium SR4 масштабируется в зависимости от размера сети. При этом увеличение управляемых контроллером коммутаторов и оконечных узлов оказывает негативное влияние на такие характеристики контроллера ПКС, как задержка и пропускная способность. Так, максимум пропускной способности контроллера составляет N = 101000 потоков в секунду, а минимум задержки t = 0,5 миллисекунды, а минимум пропускной способности контроллера составляет N = 580000 потоков в секунду, а максимум задержки t = 5,2 миллисекунды.

2. *Выводы по результаты тестирования ПКС-контроллера программным обеспечением ProLan QuTester Plus*

Анализ результатов тестирования ПКС-контроллера программным обеспечением ProLan QuTester Plus показал, что контроллер OpenDaylight Beryllium SR4 пригоден для масштабирования. Присутствует возможность построения стабильно работающей сети, включающей в себя 256 коммутаторов, поддерживающих протокол Openflow, за каждым из которых может находиться до N = 1000000 хостов включительно. Усредненные показатели сетевой скорости

чтения данных – Read Rate = 2317,5 (кбайт/с) = 18,54 (Мбит/с) и сетевой скорости записи данных – Write Rate = 1746,2 (кбайт/с) = 13,9696 (Мбит/с) также подтверждают возможность масштабирования контроллера.

3. *Выводы по результатам тестирования ПКС-контроллера программным обеспечением ОССТ*

Проведение тестирования контроллера на стабильность его работы является одним из основополагающих тестов данной методики. Программная и аппаратная части контроллера стабильно работали на протяжении 20 часов. Однако были замечены определенные скачки в производительности ПКС-контроллера, которые происходили в процессе тестирования. Так, максимальное колебание производительности произошло в районе 5-го часа тестирования, до уровня в 0%.

Анализируя другие ситуации снижения производительности в районе 4-го, 13-го и 19-го часов до уровня 87%, 77% и 70% соответственно, можно сделать вывод, что определенная нестабильность работы контроллера присутствует. Однако, наблюдая общий процент стабильной работы контроллера ПКС, стремящийся к 100%, можно принять случаи спада в производительности ПКС-контроллера за погрешность измерений. Такая характеристика контроллера, как оперативная память RAM в проведенном стрессовом тестировании колебалась в пределах 1570–1930 Мбайт = 1,57–1,93 Гбайт в пике своем, достигая 2066 Мбайт = 2,06 Гб, что составляет 10,06–12,37% и 13,2% объема оперативной памяти ПКС-контроллера.

При тестировании также измерялась температура всех ядер процессора. При анализе показателей наблюдалась нестабильная работа первого и второго ядер (core#0 и core#1) процессора и, наоборот, стабильная работа третьего и четвертого ядер (core#2 и core#3). В полном периоде тестирования максимальная температура процессора была зафиксирована в 77 °C.

4. *Выводы по результатам тестирования ПКС-контроллера программным обеспечением Mininet*

Анализ результатов тестирования программным обеспечением Mininet показывает, что ПО Mininet и разработанное нами на этой платформе ПО могут

использоваться при тестировании ПКС-контроллеров. В результате тестирования зафиксирована: максимальная производительность процессора – 325%, максимальный объем занятой оперативной памяти – 16,25%. Минимальная производительность процессора и занимаемый объем оперативной памяти, составляют, соответственно, 50% и 0,5%.

## 2.7. Разработка моделей ПКС для исследования эффективности функционирования

Для создания модели сети, а также динамического моделирования её работы, анализа и оптимизации её характеристик, управления трафиком, несомненно, нужно использовать один из мощнейших инструментов исследования сложных систем – имитационное моделирование [5].

### *2.7.1. Общее представление системы*

В настоящее время для балансировки нагрузки в сетях связи чаше всего используются аппаратные решения. Однако согласно текущим прогнозам повсеместное использование ПКС для представления услуг связи изменит процессы и методы балансировки нагрузки [46; 102].

На Рисунке 29 показан кластер контроллеров, подключенных к планировщику задач. Каждый из контроллеров является элементом кластера, но работает независимо от других. Поступающие от коммутаторов запросы поступают в буфер устройства планирования, которое изменяет адрес получателя и перенаправляет запрос к выбранному контроллеру. После получения запроса контроллер отправляет ответ на адрес контрольного планировщика, который, в свою очередь изменяет адрес отправителя на его виртуальный адрес и перенаправляет ответ на коммутатор получателя. Помимо базового процесса работы контрольного планировщика рассматривается вопрос о том, как и на основе чего выбирается тот или иной контроллер для обслуживания запросов.

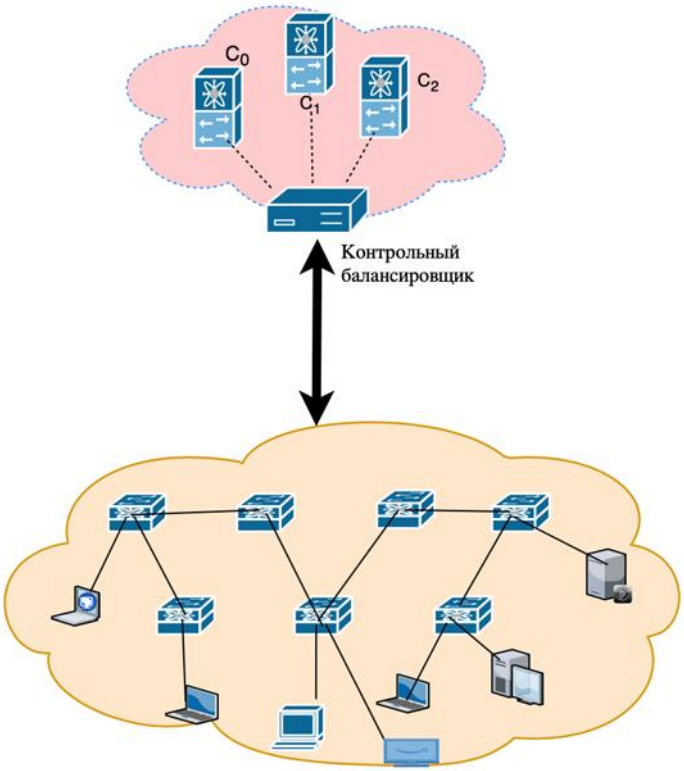


**Рисунок 29** – Модель сети для кластера контроллеров

### *2.7.2. Имитационная модель программно-конфигурируемой сети*

Традиционно, в системах с распределением нагрузки несколько взаимосвязанных узлов объединяются и образуют единый мощный узел. Эти автономные узлы работают независимо друг от друга, формируя команды по обслуживанию и распределению задач. Чтобы воспользоваться возможностью распределения задач (нагрузки) между всеми участвующими узлами, необходимы схемы взаимодействия и оптимизированные алгоритмы распределения нагрузки. Таким образом, выбор той или иной схемы обосновывается множеством параметров, таких как политика отправки запроса, политика выбора обслуживающего устройства, а также класс поступающего запроса [10; 19; 27; 30; 76; 106].

Рассмотрим фрагмент ПКС с несколькими контроллерами и представим его в виде системы массового обслуживания с очередями, как показано на Рисунке 30. Очередь на входе контрольного планировщика формируется согласно времени прибытия. Далее в данном главе будут рассмотрены две модели: аналитическая и имитационная.

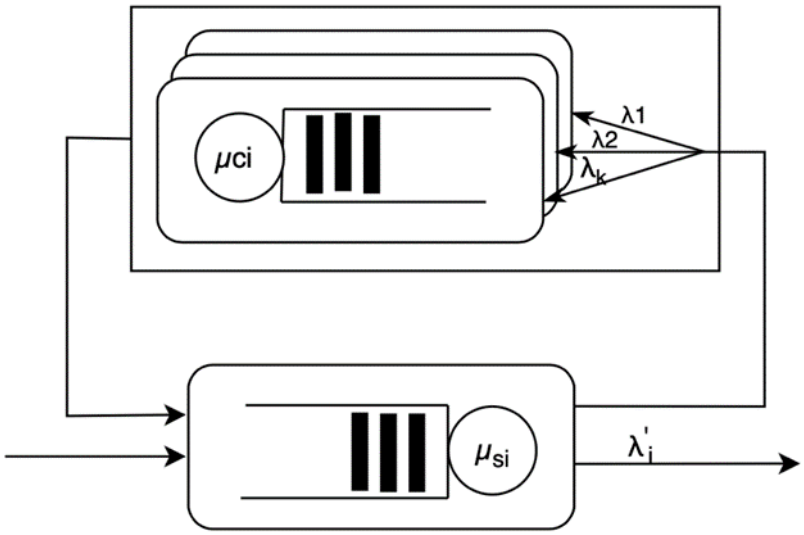


**Рисунок 30** – СМО для представленной системы контроллеров

В ходе проведения исследований были введены ограничения для упрощения реализации фрагмента сети, представленного на Рисунке 29. Предполагается, что каждая заявка обслуживается независимо друг от друга, а также может быть обслужена любым контроллером кластера. В рамках исследования рассматривали только процесс распределения нагрузки, который безвозвратно распределяет заявки по контроллерам, т.е. заявка присваивается к одному контроллеру без повторного перераспределения. Для простоты предполагается, что модель системы устойчива и контроллеры идентичны, т.е. имеют одинаковую скорость обработки заявок. Имитационная модель будет учитывать классы заявки и динамически определять состояние контроллера для обслуживаний.

### *2.7.3. Аналитический модель программно-конфигурируемых сетей*

При поступлении запроса на контрольный балансировщик определяются адрес и уровень загрузки контроллера кластера, а затем запрос направляется на адрес мало загруженного контроллера, если такой существует, или на наименее загруженный из имеющихся. Динамический процесс распределения и присваивания запросов контроллерам являются главной задачей для уменьшения времени отклика системы и эффективного использования его ресурсов.

Модель системы массового обслуживания для исследования процесса обмена данными между коммутаторами и кластером контроллеров представлена на

Рисунке 30. Система состоит из $m$ контроллеров для обслуживания, поступающих на контрольный планировщик $k$ запросов. В работе предполагается, что модель работы одного контроллера можно представить, как модель СМО M/M/1, что соответственно можно расширить до модели M/M/m с целью анализа возможностей функционирования ПКС в случае использования кластера контроллеров для распределения нагрузки. Предполагается, что запросы поступают согласно закону распределения Пуассона, а процесс обслуживания запроса происходит согласно экспоненциальному закону распределения.

Для анализа функционирования фрагмента сети будем использовать значения среднего времени обслуживания запросов системы.

Пусть $\mu$ – интенсивность обслуживания контроллера и $\Delta\omega$ – требуемое качество обслуживания запроса. Для того, чтобы гарантировать данное требование по качеству обслуживания запроса, нужно динамически выделять ему нужные ресурсы сети. В общем, $\mu$ является постоянной величиной и, соответственно, чтобы гарантировать требуемое качество обслуживания, запрос должен быть отправлен к контроллеру, способному удовлетворить требования $\Delta\omega$. В связи с этим, с увеличением интенсивности запросов может ухудшиться возможность обслуживания контроллера, если количество запросов превысить его предел возможностей обслуживания. Таким образом, среднее время обслуживания запроса изменяется в зависимости от изменения интенсивности и количества запросов. Представим $\mu$ как $\omega(t)$ – время обслуживания контроллером одного запроса в заданном интервале времени $t$.

В данном случае, для представления нужного качества обслуживания, нужно чтобы $(\omega(t) \leq \Delta\omega)$ для данной точки времени $t$.

Пусть $p_k$– вероятность того, чтобы в системе присутствовали $k$ запросов.

Здесь

$$p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_i+1}, p_0 = 1 - \frac{\lambda}{\mu}.$$

Система СМО M/M/1 может быть характеризована коэффициентами интенсивности поступления и обслуживания запросов, такими как

$$\lambda_i = \lambda, k = 1,2,3 \dots , \mu_k = \mu, k = 1,2,3 \dots$$

Применяя эти коэффициенты в $p_k$, получим

$$p_k = p_0(\frac{\lambda}{\mu})^k, k \geq 0.$$

Для устойчивости системы $0 \leq \rho < 1$, нужно чтобы $p_0 > 0$ и отсюда можно сказать, что $p_0$ является постоянной величиной. Соответственно, получим

$$p_k = (1-\rho)\rho^k, \rho = \frac{\lambda}{\mu}.$$

Применяя закон Литтла $N = \lambda\omega$, можно рассчитать количество запросов в очередь на обслуживание системе

$$N = \sum_{k=0}^{\infty} kp_k = \frac{\rho}{1-\rho}.$$

Таким образом, получим

$$\omega(t) = \frac{\frac{1}{\mu}}{1-\rho} = \frac{1}{\mu - \lambda}. \quad (1)$$

Для уравнения (1) предполагается, что $\omega(t)$ соответствует ожиданиям по качеству обслуживания запроса. Однако при увеличении интенсивности поступающих запросов требуются дополнительные ресурсы. Уравнение (1) не может обусловливать процесс обслуживания, поэтому применяется модель M/M/m для анализа среднего времени обслуживания контроллера.

Предполагаются также система с бесконечным местом в очереди и постоянной интенсивностью поступления запросов $\lambda$.

Система обеспечивает максимальное $m$ число контроллеров, поэтому

$$\mu_k = \min[k\mu, m\mu] = \begin{cases} k\mu & 0 \leq k \leq m, \\ m\mu & m \leq k. \end{cases}$$

Для устойчивости системы

$$\lambda = \frac{\lambda}{m\mu} \leq 1.$$

Соответственно

$$p_k = \begin{cases} p_0 \dfrac{(m\rho)^k}{k!} & k \le m, \\ p_0 \dfrac{\rho^k m^m}{m!} & k \ge m. \end{cases}$$

А также для $p_0$ получим

$$p_0 = \left[1 + \sum_{k=1}^{m-1} \frac{(m\rho)^k}{k!} + \sum_{k=m}^{\infty} \frac{(m\rho)^k}{k!} \frac{1}{m^{k-m}}\right]^{-1}.$$

Вероятность того, чтобы поступающий запрос попал в очередь

$$p_q = \sum_{k=m}^{\infty} p_k = p_0 \frac{(m\rho)^m}{m!(1-\rho)} = 1 - \sum_{k=0}^{m} p_0 \frac{(m\rho)^k}{k!}.$$

Так как нам нужно найти среднее время обслуживания запроса $\omega(t)$, можно заметить, что число $m$ контроллеров является ещё и параметром $\omega(t)$.

Соответственно,

$$\omega(t,m) = \mathrm{E}[\omega(t,m)] = \frac{1}{\lambda}\left(m\rho + \rho \frac{(m\rho)^m}{m!} \frac{p_0}{(1-\rho)^2}\right).$$

Применяя $\lambda = \frac{\lambda}{m\mu}$ в (1) получим

$$\omega(t,m) = \frac{1}{\mu} + \frac{1}{\lambda} \frac{\left(\frac{\lambda}{\mu}\right)^m}{m!} \frac{p_0}{\left(1 - \frac{\lambda}{m\mu}\right)^2}. \quad (2)$$

Как было рассмотрено выше, для того, чтобы гарантировать требуемое качество обслуживания, неравенство (3) должно быть удовлетворено, следующим образом:

$$\frac{1}{\mu} + \frac{1}{\lambda} \frac{\left(\frac{\lambda}{\mu}\right)^m}{m!} \frac{p_0}{\left(1 - \frac{\lambda}{m\mu}\right)^2} \le \frac{1}{\mu - \lambda}. \quad (3)$$

Пусть $r \ge 1$ определяет дополнительный трафик в системе, тогда, умножая $\lambda$ на $r$ и упрощая уравнение (2) (при этом должны быть удовлетворены $r\lambda/\mu \le m$ и $r \ge 1$), получим

$$f(r,m) = \frac{\lambda}{\mu} - (\mu - \lambda) \frac{(r\lambda)^{m-1}}{m!\mu^m} \frac{p_0}{\left(1 - \frac{r\lambda}{m\mu}\right)^2}.$$

Модель работает согласно следующему алгоритму планирования распределения:

1) определить количество контроллеров m в кластере C;

2) для всех контроллеров определить интенсивность обслуживания: $\mu_k, \forall k$;

3) определить интенсивность поступления запросов на контрольный планировщик в интервал времени t: $\lambda_k(t)$;

4) для всех контроллеров определить $\omega(t, m)$,

5) определить контроллер $c_{target}$ для обслуживания поступающего запроса:

– задать промежуточное максимальное значение времени обслуживания запроса;

– проверить из кластера C время отклика $\omega_j$контроллера $c_j$:

– если $\omega_j < \omega_{jmax}$ , то $\omega(t, m) = \omega_j$ и $c_{target} = c_j$;

– для $j=1,\ldots, m$ пройти весь цикл и найти менее загруженный контроллер $c_{target}$ с наименьшим временем обслуживания;

– если $c_{target}$ не найдется, сохранять запросы в очередь;

6) при поступлении нового запроса, повторять процесс с шага 3.

### *2.7.4. Результаты моделирования*

Для оценки предложенного алгоритма была разработана имитационная модель фрагмента ПКС в среде AnyLogic. Модель описывает процесс обслуживания входящих сообщений на разные блоки обслуживания.

После рассмотрения аналитического метода оценки качества обслуживания в сети ПКС проанализировали ситуацию в случае изменения трафика. Для удобства был рассмотрен случай использования одного контроллера с увеличенной производительностью в 3 раза и проведено сравнение со случаем использования кластера из 3 контроллеров с одинаковыми значениями производительности. Результаты моделирования представлены в рисунках ниже.

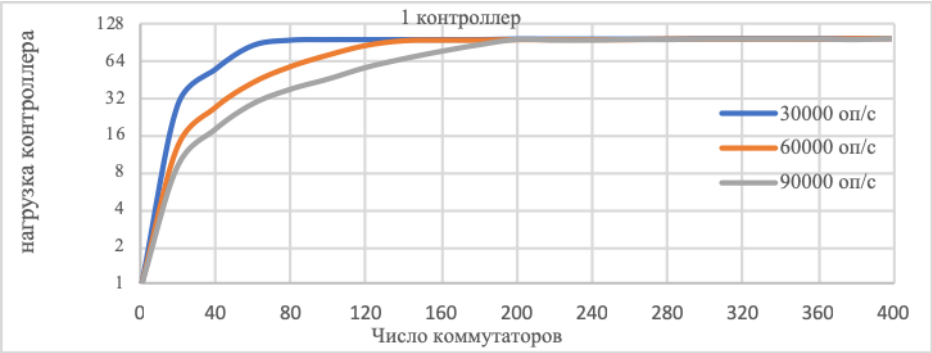


**Рисунок 31** – Зависимости нагрузки контроллера от количества подключаемых коммутаторов в случае одного контроллера

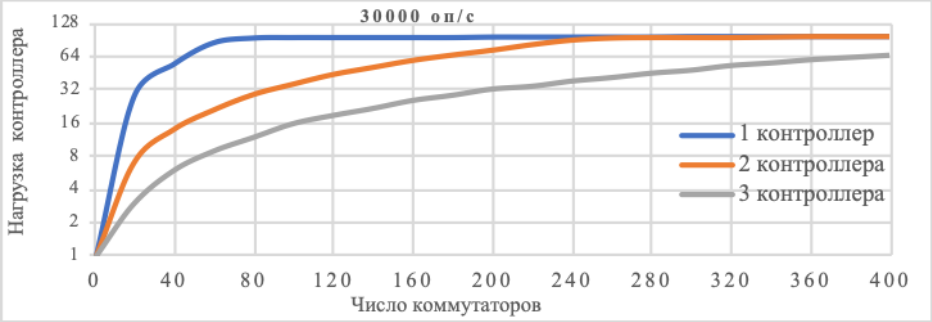


**Рисунок 32** – Зависимости нагрузки контроллера от количества подключаемых коммутаторов в случае кластера из 3 контроллеров

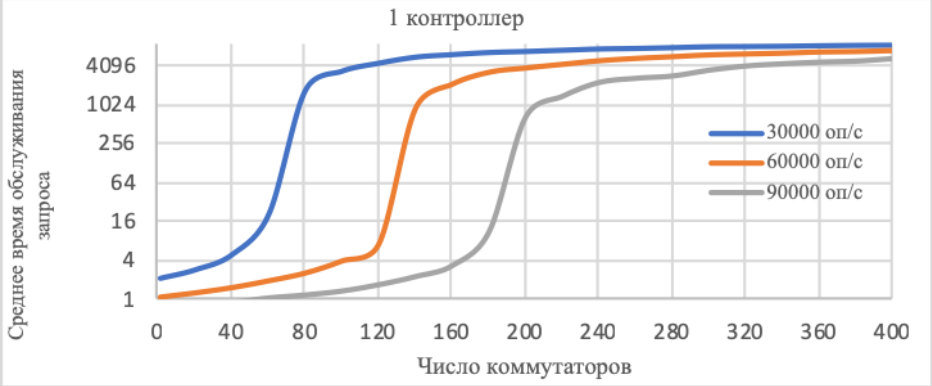


**Рисунок 33** – Зависимости среднего времени обслуживания контроллера от количества подключаемых коммутаторов в случае одного контроллера

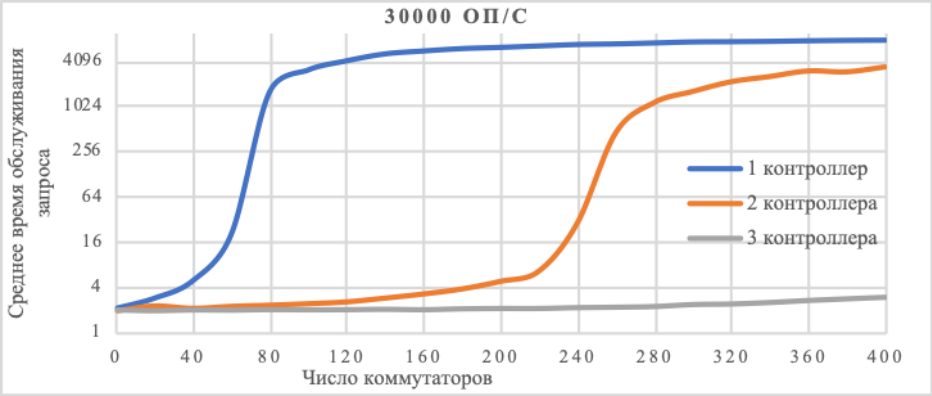


**Рисунок 34 –** Зависимости среднего времени обслуживания контроллера от количества подключаемых коммутаторов в случае кластера из 3 контроллеров

На Рисунках 31 и 32 показаны изменения нагрузки контроллера, в зависимости от числа подключаемых коммутаторов. На графике зависимостей видно, что при увеличении числа коммутаторов соответственно увеличивается нагрузка на контроллере вплоть до критического значения. Пропорционально увеличивается среднее время обслуживания запроса, что показано на Рисунке 32.

На Рисунках 31 и 33 показаны графики результатов моделирования системы с одним контролером, при этом были установлены значения производительности ядра ($\mu$) в 30000 оп/с, 60000 оп/с и 90000 оп/с соответственно. Результаты показывают, что контроллер может обслуживать, в лучшем случае, до 180 коммутаторов до того, как ухудшится показатель обслуживания, обусловленный средним временем обслуживания. На Рисунках 32 и 34 представлены графики результатов моделирования системы, где был использован алгоритм планирования распределения нагрузки; вместо одного контроллера с производительностью $\mu = 90000$ *оп/с* кластер из 3 контроллеров, каждый с производительностью $\mu = 30000$ *оп/с*. Результаты показывают, что система может обслуживать до 500 коммутаторов без значительного уменьшения показателей обслуживания.

Проведенный анализ показывает, что использование кластера контроллеров в 3 раза улучшает возможности сети в представлении требуемого качества обслуживания. Это объясняется тем, что контроллеры могут использоваться рационально в зависимости от потребностей сети те могут, находится в разных режимах (ожидания, спящий), что улучшает процесс балансировки нагрузки.

## Выводы по главе 2

1. Проанализированы принципы функционирования программно-конфигурируемой сети для последующего описания в виде СМО.

2. С целью исследования особенностей функционирования ПКС-контроллера разработана методика проведения тестирования. На базе модельной сети лаборатории проведено тестирование и представлены результаты проведенных экспериментов по тестированию контроллера OpenDaylight Beryllium

SR 4 и аппаратной платформы, программно-конфигурируемых сетей с использованием разработанной методики.

3. Разработаны модели программно-конфигурируемой сети, позволяющие оценить эффективность её работы в условиях высокой нагрузки и рационально планировать размещение элементов сети на этапе развертывания и масштабирования.

4. Проведён сравнительный анализ таких характеристик контроллера как, производительность, время обработки потоков, стабильность и масштабируемость. Показано, что использование кластера контроллеров в 3 раза улучшает возможности сети в представлении требуемого качества обслуживания.

# Глава 3. Метод кластеризации ресурсов программно-конфигурируемых сетей для динамического распределения контроллеров

## 3.1. Проблема распределения контроллеров программно-конфигурируемых сетей

Архитектура физически распределенных контроллеров (РК) стала совершенно новым подходом к организации уровня управления в ПКС [39]. Ввиду того, что ПКС – одна из ключевых технологий сети связи пятого поколения [4; 56; 57]. Архитектура РК появилась, в частности, из-за двух основных недостатков, присущих ПКС: использование единого централизованного контроллера и проблема масштабируемости [99]. В ПКС масштабируемость отражает способность контроллера обрабатывать несколько запросов маршрута пересылки от коммутаторов. Как известно, контроллер ПКС имеет ограниченный ресурс при обработке запросов [10; 98]. Чтобы решить эту проблему, исследователи пытались ограничить количество запросов маршрута пересылки, отправляемых на контроллер [107]. Однако такая стратегия базируется на добавлении интеллектуальных функций коммутатору и тем самым нарушает саму концепцию ПКС.

Анализ аспектов надежности обусловлен тем, что единый контроллер имеет проблему единой точки отказа [74]. В случае выхода контроллера из строя коммутаторы теряют способность пересылать новые пакеты, и, в конечном итоге, вся сеть выходит из строя. Для решения этой проблемы была предложена частичная модификация функций коммутатора OpenFlow, а также придание ему гибридных свойств. Отличительной особенностью нового OpenFlow-hybrid коммутатора является то, что он может переключаться в режим традиционного коммутатора при обрыве соединения с контроллером ПКС [97]. Другим вариантом решения может стать технология физического РК в ПКС, где на одном уровне управления используются несколько контроллеров, равномерно распределяющих нагрузку между собой [76].

Появление данной концепции подтолкнуло к разработке новых проектов. Один из них является HyperFlow – приложение, разработанное для работы поверх контроллера NOX с целью реализации логически централизованной архитектуры.

Кроме этого, существуют различные структуры уровней управления, такие как ONIX [67], DISCO [92], ONOS [24] и другие с логически централизованной архитектурой распределения контроллеров. Из логически распределенных архитектур можно выделить KANDOO [109] – уровень управления с иерархической структурой, использующий статический алгоритм, и ORION [40] – гибридный уровень управления, совмещающий плоскую и иерархическую структуры.

Поскольку на данный момент отсутствуют стандарты и рекомендации, описывающие взаимодействие компонентов ПКС, ученые работают над самыми разными вариантами реализации этой технологии. Время отклика контроллера ПКС – основной фактор при принятии решения о необходимости развертывания дополнительного контроллера ПКС [8].

Вопросы количества и распределения контроллеров были представлены в некоторых публикациях [26; 43; 47]. В первом исследовании по этой тематике [57] были рассмотрены возможности развертывания ПКС в архитектурах WAN, где основная проблема – задержка. Наилучший выбор расположения контроллера зависит от заданной сетевой задержки передачи данных, обеспечивающей разумную стабильность и скорость между ответами контроллера и удаленным контроллером. Исследователи представляют задачу размещения контроллеров и описывают трудности, возникающие между устройствами передачи и контроллерами при оптимизации сетевой задержки. Было обнаружено, что дополнительные контроллеры улучшают время отклика в большинстве топологий, при этом количество контроллеров будет зависеть от топологии сети и выбора метрик. В некоторых случаях одного контроллера достаточно для удовлетворения общих требований для ПКС.

В [48] предложено наиболее надежное расположение ПКС-контроллеров. С одной стороны, представлена новая метрика, называемая ожидаемым процентом допустимого пути следования при сбое ПКС, с другой стороны, алгоритм, дающий наиболее идеальные результаты. Пути следования – это пути, определенные сетью для связи коммутаторов с соответствующим контроллером и контроллерами между

собой, поэтому этот процент отражает надежность управления сетью. Результаты исследования основаны на том факте, что алгоритм Greedy обеспечивает наиболее оптимальное решение, при этом местоположение контроллера должно быть тщательно выбрано и зависеть от используемого алгоритма.

В [100] ученые попытались оптимизировать надежность ПКС-контроллера путем минимизации ожидаемого процента потерь на трактах управления. Для того чтобы оценить надежность, исследователи используют алгоритм «грубой силы (brute force)», чтобы сгенерировать все возможные комбинации контроллеров в каждом потенциальном местоположении и измерить стоимость каждого размещения контроллера ПКС. Они считают, что применение нескольких контроллеров снижает надежность сети. Однако для всех протестированных архитектур превышение определенного числа контроллеров может иметь нежелательный эффект – снижение производительности. Ученые приходят к выводу, что наилучшее количество контроллеров находится в пределах [0,035n – 0,117n], где n – количество узлов.

В [62] представлены две категории решений две категории решений вопроса размещения контроллеров: первая категория предлагает физически распределенные контроллеры, но логически централизованные посредством синхронизации всей информации, касающейся сети и балансировки нагрузки между контроллерами. Вторая категория предлагает логически распределенную архитектуру контроллеров, где каждый контроллер управляет своим доменом и распространяет полезную информацию с другими экземплярами.

## 3.2. Программно-конфигурируемые сети с распределенными контроллерами

В раннее предлагаемой архитектуре программно-конфигурируемых сетей уровень управления использовал единый централизованный контроллер, управляющий всеми процессами организации и контроля на нескольких сетевых устройствах. В ходе исследований и эксплуатации было выявлено, что данная физически централизованная архитектура может быть эффективна для небольших

сетей, однако крупномасштабные сети с высокими требованиями не могут работать через один контроллер, так как он представляет «узкое место» в сети [98]. Использование единого централизованного контроллера уже не отвечает нынешним требованиям сетей, что подталкивает проектировщиков к внедрению ПКС с распределенными контроллерами в свои проекты. Идея данных сетей заключается в использовании нескольких контроллеров вместо одного для управления платформой передачи данных [108].

Программно-конфигурируемые сети с физически распределенными контроллерами разделяются в две основные категории: логически централизованные и логически распределенные архитектуры. Далее, логически распределенную категорию можно разделить на две подкатегории: плоско распределенные и иерархически распределенные (Рисунок 35).

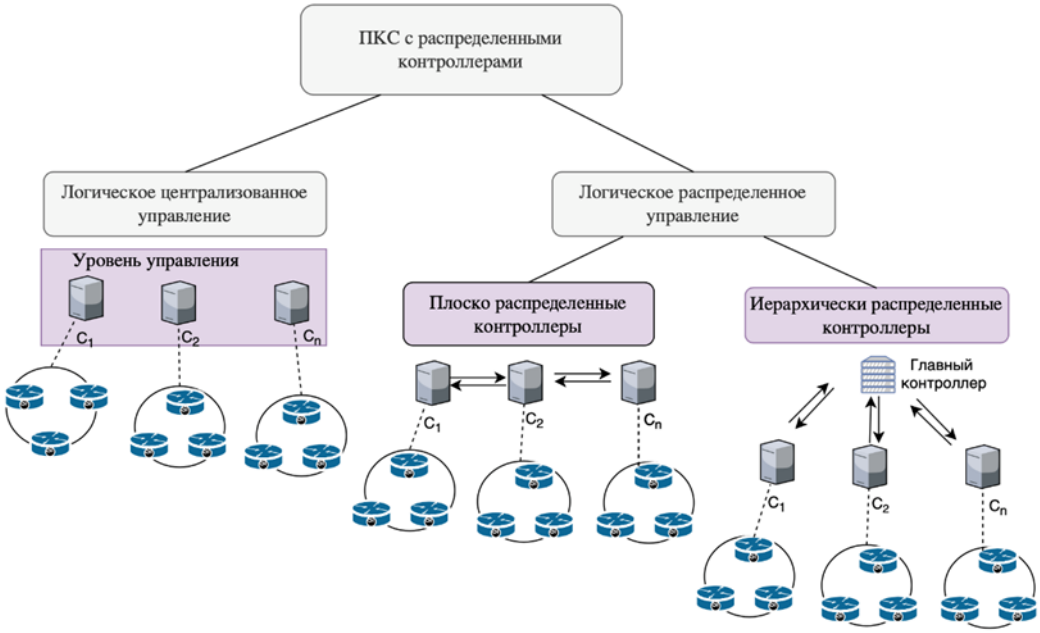


**Рисунок 35** – Архитектура ПКС с физически распределенными контроллерами

В логически централизованной архитектуре обязанности одинаково распределяются между несколькими контроллерами. Однако для уровня передачи данных вся структура выглядит как единый контроллер, который управляет всей сетью. Все контроллеры всегда оповещены о любых изменениях в сети, используют единую базу данных и мгновенно делятся информацией, благодаря сетевой синхронизации. Одним словом, логически централизованная архитектура

остается рядом с начальной концепцией ПКС, которая использует единый контроллер.

В логически распределенной архитектуре контроллеры распределены физически и логически. Кроме того, каждый контроллер имеет представление только о тех сетевых устройствах, за которые он отвечает, и принимать решения относительно конфигурации может лишь для них.

Данная категория имеет две подкатегории:

1. *Плоско распределенная архитектура*. В плоской или горизонтальной архитектуре все контроллеры расположены на одном уровне управления, каждый контроллер имеет одинаковые обязанности, отвечает за свою часть сети и взаимодействует с другими на основе заранее объявленного механизма.

2. *Иерархически распределенная архитектура*. В иерархической архитектуре используется более одного уровня контроллеров. Это значит, что некоторые контроллеры имеют больше прав и обязанностей. Наиболее эффективное применение таких сетей – сеть, в которой используется главный контролер над уровнем распределенных контроллеров. Логически распределенная архитектура уходит от первой концепции ПКС, так как предлагает разделение определенных обязанностей между контроллерами внутри сети.

Кроме данных категорий и подкатегорий программно-конфигурируемых сетей с распределенными контроллерами, они различаются по использованию статического, либо динамического алгоритма.

В статическом алгоритме позиции контроллеров и связи с коммутаторами определяются при конфигурации сети и остаются неизменными с течением времени.

В динамическом алгоритме позиции контроллеров и связи с коммутаторами могут изменяться с течением времени, что делает сеть гибкой, позволяя подстраиваться под изменения нагрузки в сети.

## 3.3. Разработка алгоритма динамического распределения ПКС-контроллеров

Иерархическая структура распределенных контроллеров в программно-конфигурируемых сетях использует несколько контроллеров в уровне управления с механизмами динамической кластеризации для балансировки нагрузки между этими контроллерами.

Реализация алгоритма была апробирована на базе модельной сети лаборатории программно-конфигурируемых сетей СПбГУТ. Модельная сеть рассматривается как общая трехуровневая система ПКС с несколькими контроллерами в уровне управления и механизмами динамической кластеризации для балансировки нагрузки между контроллерами. В соответствии с данной структурой, уровень управления состоит из двух подуровней: нижний, содержащий РК, и верхний, представляющий собой главный контроллер (ГК). ГК берет на себя ответственность за организацию и контроль РК, а также выступает в качестве шлюза для связи с контроллерами других сетей (Рисунок 36).

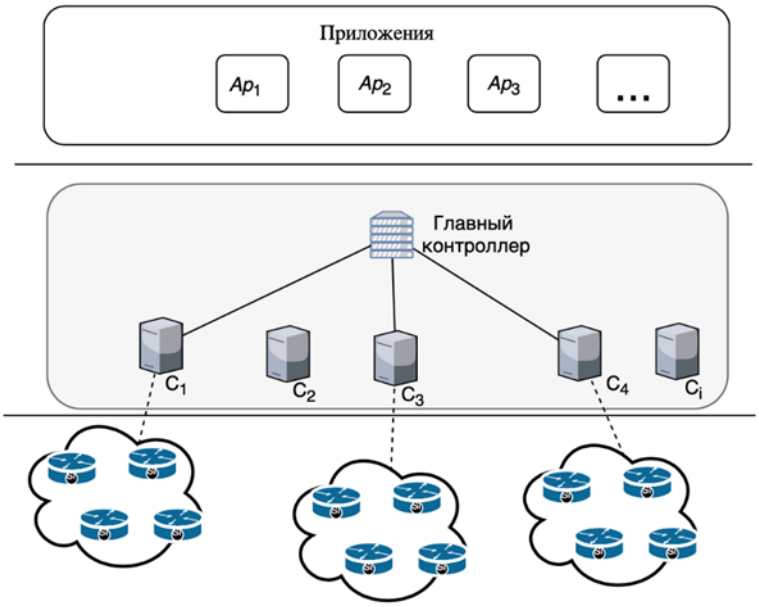


**Рисунок 36** – ПКС как трехуровневая система

РК организованы в кластеры, каждый из которых содержит только один ПКС-контроллер и группу устройств передачи данных (коммутаторы OpenFlow). РК работают вместе под управлением ГК для достижения требуемой

масштабируемости и надежности сети. ГК отвечает за настройку распределенных кластеров и информирование каждого контроллера о членах кластера – коммутаторах OpenFlow.

Формирование кластера является динамическим процессом. Это означает, что кластеры время от времени могут меняться из-за состояний контроллера. Алгоритм DDC (Dynamic and Distributed Clustering, алгоритма динамического распределения) изменяет кластеры с помощью рабочей нагрузки каждого контроллера. ГК получает периодические отчеты о рабочей нагрузке и решает сохранять кластеры или изменять их.

Вся работа предложенной структуры ПКС делится на раунды, которые с точки зрения продолжительности являются гетерогенными. Каждый раунд состоит из двух основных этапов: настройки и устойчивого состояния (Рисунок 37).

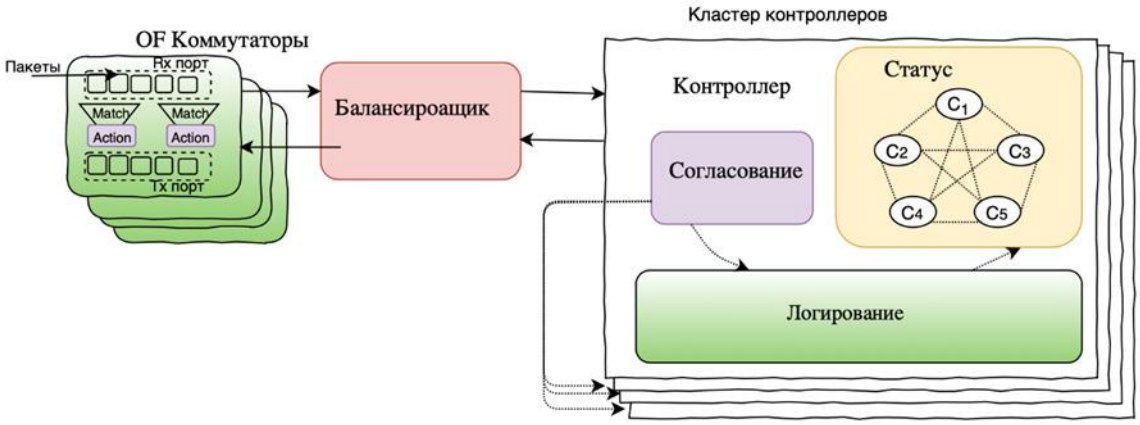


**Рисунок 37** – Модель периодического регистрации рабочей нагрузки контроллеров

На этапе настройки ГК формирует кластеры и объявляет контроллеру коммутаторы OpenFlow, являющиеся членами его кластера. ГК выбирает структуру кластера, основываясь на периодических сообщениях рабочей нагрузки каждого РК.

На этапе устойчивого состояния РК связывается с коммутаторами и управляет ими в своем кластере. Набор коммутаторов OpenFlow создает свои таблицы пересылки на основе инструкций, полученных от РК. Периодически каждый контроллер проверяет и вычисляет свою рабочую нагрузку $W_i$ , а затем

отправляет отчет ГК, который сравнивает рабочую нагрузку каждого контроллера с максимальной рабочей нагрузкой $W_{\max}$, назначенной для всех контроллеров. Предполагается, что все РК являются однородными с точки зрения аппаратных и программных возможностей и, таким образом, имеют один и тот же $W_{\max.}$ Затем ГК вычисляет процентную нагрузку $L_i$ каждого контроллера по формуле

$$L_i = \frac{W_i}{W_{\max}} 100\%. \quad (4)$$

ГК имеет два максимальных пороговых значения для рабочей нагрузки контроллера: желтый и красный. Если процентная нагрузка превышает 80%, контроллер преодолевает первый желтый порог, и ГК помещает его в таблицу загруженных контроллеров, которые могут быть перегружены, если нагрузка продолжает расти. Когда процентная нагрузка достигает 90% (красный порог), это означает, что контроллер скоро будет перегружен. ГК принимает решение о перестроении кластера для преодоления проблемы перегрузки, которая может привести к отказу или блокировке контроллера.

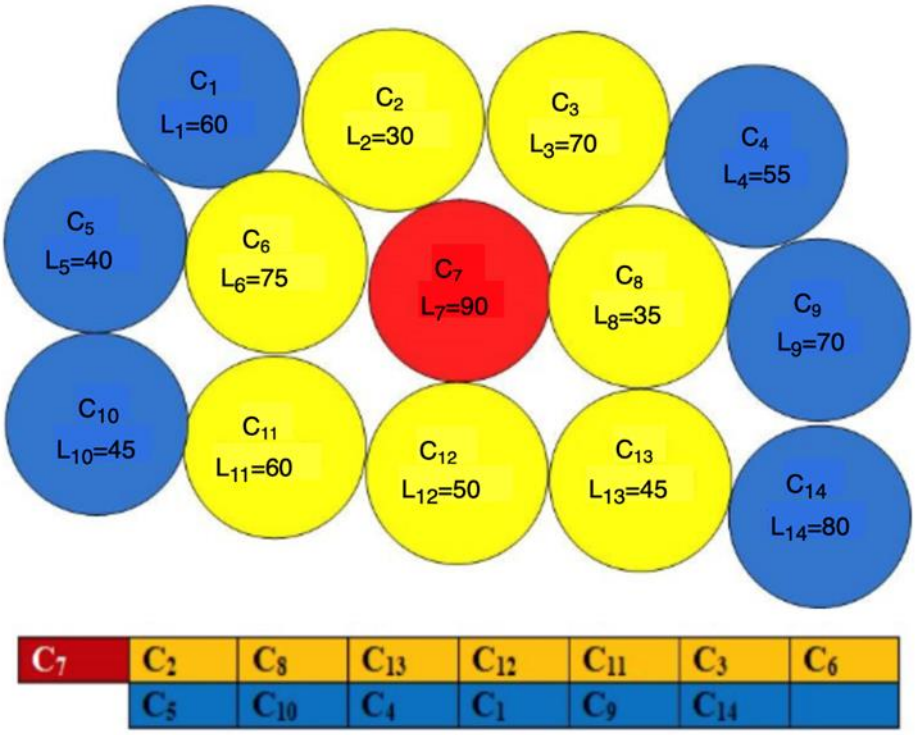


**Рисунок 38** – Часть таблицы загруженных контроллеров для одного контроллера

Для всех РК в загруженной таблице ГК организует соседние контроллеры, а также смежные соседним контроллерам на основе их процентной загрузки, включая их в реорганизацию кластера. Пример части таблицы загруженного контроллера представлен на Рисунке 4. Если все соседние контроллеры имеют процентную нагрузку ниже желтого порога, ГК перестраивает кластер, содержащий загруженный контроллер, и соседние, чтобы преодолеть проблему возможной перегрузки сети. Если один из соседних контроллеров также загружен, ГК перестраивает кластеры этих соседних контроллеров и смежные соседним контроллерам. При такой смене рабочая нагрузка равномерно распределяется среди кластеров, и перегрузка контроллера исчезает.

Во время функционирования системы производится периодическая регистрация (мониторинг) трафика, в результате которой получаем набор значений $\lambda = \{\lambda_1, \lambda_1, \dots, \lambda_n\}$. Вывод о превышении порогового значения делается на основе анализа этой последовательности, т.е. за интервал времени, в течение которого получен выбор из $n$ значений.

Для обеспечения устойчивости системы и определения количества контроллеров в кластере устанавливается допустимый доверительный интервал $\Delta_\lambda$, позволяющий уточнить с относительной погрешностью $\delta = \frac{\Delta_\lambda}{\bar{\lambda}}$ допустимое истинное пороговое значение, где $\bar{\lambda}$ – среднее значение интенсивности трафика.

Представим трафик, поступающий на ГК от $i$ подключенных коммутаторов, имеющих разные интенсивности. Для $n$ наблюдений среднее значение интенсивности трафика

$$\bar{\lambda} = \frac{1}{n}\sum_{i=1}^{n} \lambda_i.$$

Доверительный интервал для среднего значения

$$\lambda = \bar{\lambda} \pm t_{\alpha,n}\frac{\sigma_\lambda}{\sqrt{n}},$$

где $\sigma_\lambda = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\lambda_i - \bar{\lambda}\right)^2}$ – выборочное среднеквадратическое отклонение (стандартное отклонение);

$t_{\alpha,n}\frac{\sigma_\lambda}{\sqrt{n}} = \Delta_\lambda$ – доверительный интервал в $n$ наблюдений,

где $t_{\alpha,n}$ – коэффициент Стьюдента для доверительной вероятности $\alpha$ и количества наблюдений $n$ (степеней свободы $n$-1).

Тогда с доверительной вероятностью $\alpha = 0{,}99$ пороговое значение для устойчивости системы будет установлено при соблюдении неравенства

$$\bar{\lambda} - \Delta_{\lambda} \leq \lambda_{\text{ист. зн}}.$$

Последнее выражение говорит о том, что среднее значение трафика не превышает порогового значения $\lambda_{\text{ист. зн}}$ с вероятностью $\alpha$.

### 3.4. Результаты моделирования

Система из ГК и подчиненных ему десяти РК спроектирована с использованием параметров моделирования, указанных в Таблице 3. Предполагается, что все РК начинают работать с фиксированной рабочей нагрузкой $L_{str}$ и случайным образом получают рабочую нагрузку от членов кластера в процессе моделирования на основе событий. Рассмотрены три случая симуляции, в каждом из которых изменялась максимальная рабочая нагрузка РК.

**Таблица 3** – Параметры моделирования

| *Параметры* | *Значение* | |
|---|---|---|
| Количество РК(i) | 10 | |
| Начальная рабочая нагрузка ($L_{str}$) | 200 (событий/с) | |
| Максимальная рабочая нагрузка ($W_{max}$) | Случай 1 | 1000 (событий/сек) |
| | Случай 2 | 1100 (событий/сек) |
| | Случай 3 | 1200 (событий/сек) |

Для лучшего отражения производительности новой технологии динамической кластеризации ее сравнивали с популярной технологией статической кластеризации, представленной моделью системы, в которой кластеры остаются неизменными после их формирования. Предполагалось, что РК в статической системе кластеризации отказывают после перегрузки на 20% от максимальной рабочей нагрузки.

При моделировании среднюю рабочую нагрузку использовали как показатель производительности для оценки алгоритма DDC при балансировке

нагрузки между РК. Средняя рабочая нагрузка каждого контроллера всегда вычисляется для динамической и статической кластеризации. В результате были получены рабочие нагрузки контроллеров и зависимости (Рисунок 39), иллюстрирующие статус рабочей нагрузки контроллера C5 для статической и динамической кластеризации в трех случаях моделирования.

В первом случае можно видеть, что кривые динамической и статической кластеризации будут совмещены до тех пор, пока рабочая нагрузка не подойдет слишком близко к красному порогу. В этот момент алгоритм DDC реорганизовывает кластеры, вследствие чего уменьшается нагрузка на C5, в то время как в статической системе кластер остается неизменным. Все это приводит к перегрузке и отказу контроллера C5 при достижении значения 1200 событий/с, что на 20% больше максимальной рабочей нагрузки. Во втором случае кривая динамической кластеризации в какой-то момент начинает убывать, однако контроллер C5 не загружается по красному порогу. Объяснить это можно реорганизацией кластера за счет перегрузки соседнего контроллера. Третий случай похож на первый, где при достижении красного порога проводится реорганизация кластеров.

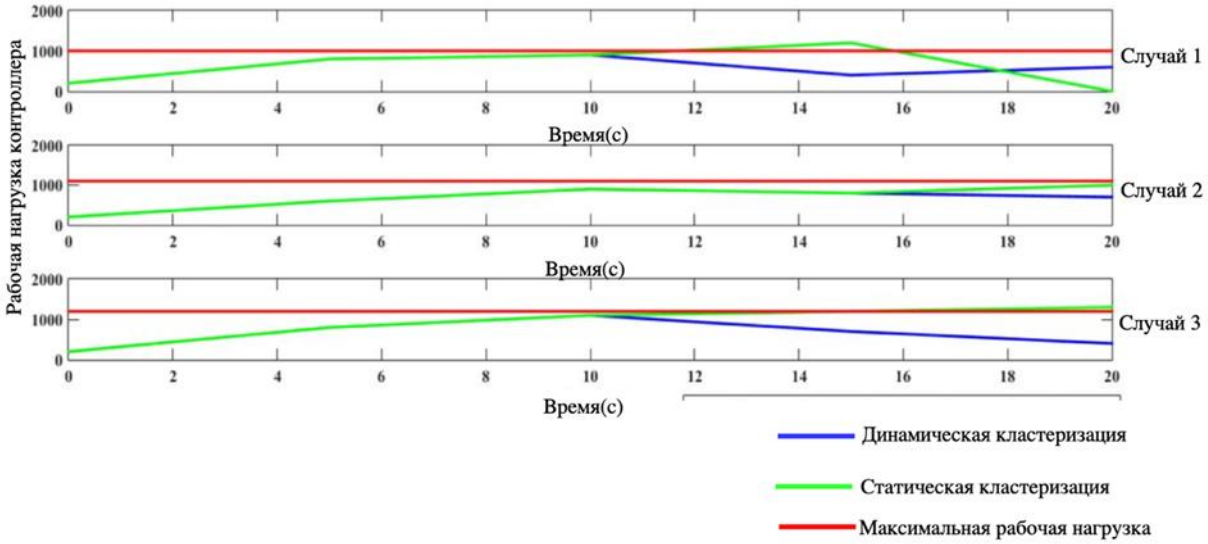


**Рисунок 39** – Статус рабочей нагрузки для C5

На Рисунке 40 проиллюстрирован статус рабочей нагрузки контроллера $C_7$ для двух ситуаций кластеризации в трех случаях симуляции.

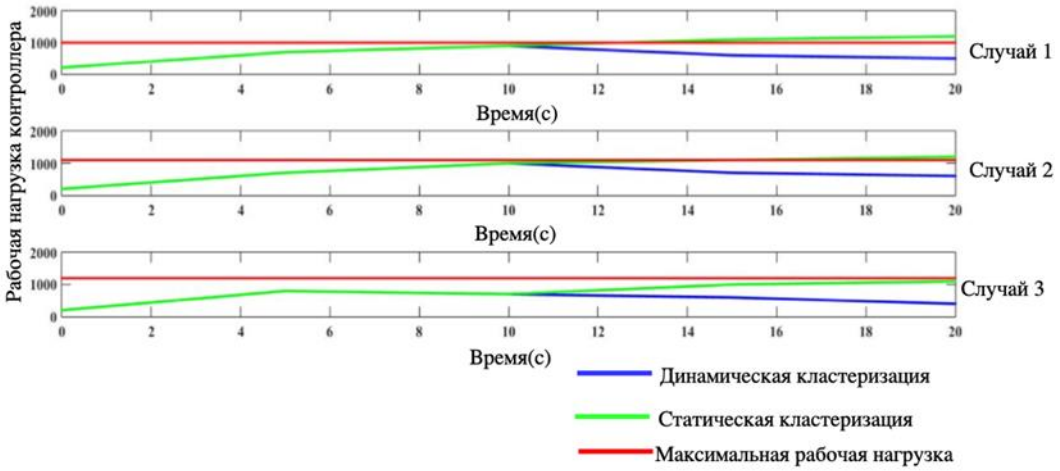


**Рисунок 40** – Статус рабочей нагрузки для C7

Оба графика демонстрируют большую разницу между статической и динамической кластеризацией. Статическая кластеризация может привести к сбою контроллера при его перегрузке, из-за чего часть сети перестанет работать. Алгоритм DDC уравновешивает рабочую нагрузку между контроллерами и предотвращает перегрузочные ситуации.

На Рисунке 41 представлена средняя рабочая нагрузка каждого контроллера как при динамической, так и статической кластеризации для трех рассмотренных случаев. Средняя рабочая нагрузка РК в DDC меньше, чем в статической кластеризации.

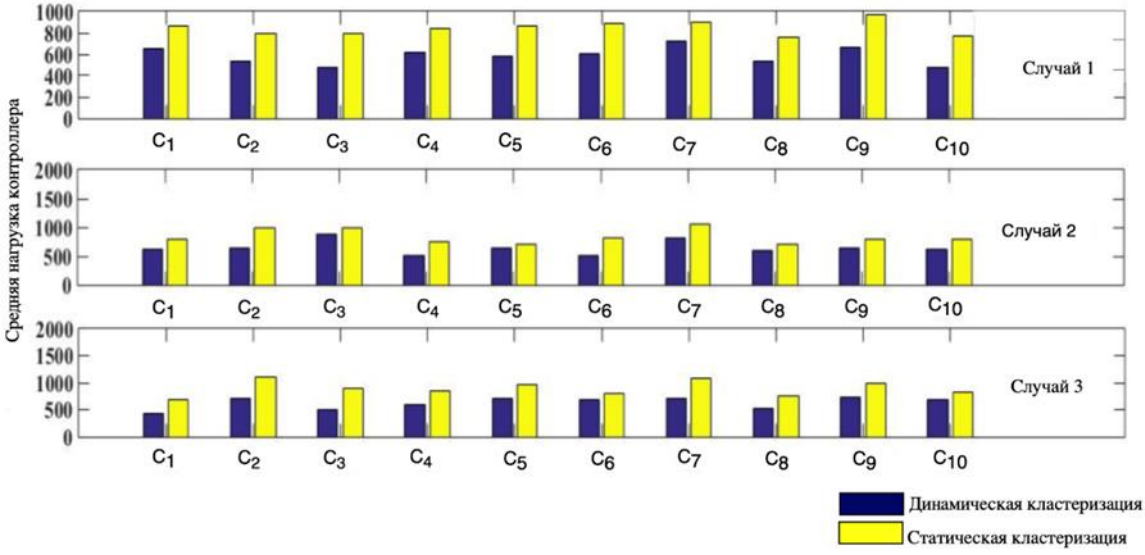


**Рисунок 41** – Средняя нагрузка для каждого контроллера в трех случаях

## 3.5. Разработка модели классификации и приоритизации трафика в программно-конфигурируемых сетях

### *3.5.1. Характеристики сетевого трафика*

Подключенные к сети приложения генерируют трафик (входящий и исходящий) в зависимости от определенных функций конфигурации приложений. Передаваемые пакеты включают трафик конфигурации сети, протокол сетевого времени (NTP, *от англ.* Network Time Protocol), систему доменных имен (DNS, *от англ.* Domain Name System), связь между устройствами и сервером, а также трафик, генерируемый в результате взаимодействия с пользователем.

Несмотря на то, что разные приложения в сети могут использовать различные протоколы и передавать данные для разных целей, большинство трафика использует протоколы TCP/IP. Для однозначного определения принадлежности пакета тому или иному потоку в OpenFlow-таблице в полях, принадлежащих MatchField указываются соответствующие значения. Таким образом, на основе группы параметров, например, IP-адрес источника/назначения возможно выделить соответствующий потоку и мониторить показатели счетчика потока (Packet count, Byte count). Параметры пакетов исследуемых потоков отображены в Таблице 4.

**Таблица 4** – Параметры пакетов в сети

| *Тип* | *Параметр* |
|---|---|
| Протокол канального уровня | ARP / LLC |
| Протокол сетевого уровня | IP / ICMP / ICMPv6 / EAPoL |
| Протокол транспортного уровня | TCP / UDP |
| Протокол уровня приложения | HTTP / HTTPS / DHCP / BOOTP / SSDP / DNS / MDNS / NTP |
| Дополнительные параметры IP | Оповещение о прокладке / маршрутизации |
| Содержание пакета | Размер / Необработанные данные |
| Адрес IP | Счетчик потока при заданном IP-адресе источника /назначения |
| Класс IP | Источник / Назначение |

Каждый поток в трафике содержит основную информацию о пакете, от MAC-уровня до уровня приложения. Сетевой трафик может рассматриваться как данные

временных рядов и содержит полезную информацию о пользователях, устройствах и состоянии сети. Для сбора трафика в данном случае применялись анализаторы пакетов сетевого трафика Wireshark. Из-за ограничений средств сетевой безопасности, таких как протокол уровня защищенных сокетов (SSL, *от англ.* Secure Sockets Layer) и протокол защиты транспортного уровня (TLS, *от англ.* Transport Layer Security), для классификации возможно использование только заголовков пакетов.

Соответственно, для классификации сетевого трафика необходим оптимальный набор признаков. Признак трафика – это атрибут, значение которого отличается для разных типов классов трафика. Например, средний размер пакета, как правило, различен для потоков мультимедийного контента и потоков загрузки, поскольку в последних почти все пакеты имеют полный размер, что не относится к мультимедийным потокам.

### *3.5.2. Модифицированный алгоритм кластеризации k-means*

Цель построения алгоритма состоит в том, чтобы решать задачу кластеризации для потоков трафика в сети связи [3]. Предполагается, что один поток трафика реализует (или участвует в реализации) одной из возможных услуг связи. При этом возможен конечный набор $k$ видов трафика, например, передача видео, передача музыки, речи, интерактивного видео, загрузка файлов и др. Каждый из видов трафика имеет определенные характеристики, которые отражаются в его параметрах в, возможно, некоторых признаках, которые могут быть получены путем его мониторинга. Пусть $d$ – количество таких характеристик. При выполнении практических экспериментов $d = 13$. Перечень параметров приведен в Таблице 5.

Возьмем за основу алгоритм кластеризации k-means [63], который позволяет выделять заданное количество кластеров. Модифицируем этот алгоритм с целью его применения в данном случае. Особенность кластеризации (классификации) потоков состоит в следующем: общее количество характеристик трафика, доступных для мониторинга достаточно велико; трафик характеризуется

различными параметрами, имеющими различные единицы измерения и диапазоны возможных численных значений; количество наблюдений (результатов мониторинга, потоков) изменяется со временем.

**Таблица 5** – Набор признаков для модели классификации трафика

| *Название* | *Описание* |
|---|---|
| Source IP (src_IP) | IP адрес источника |
| Destination IP (dst_IP) | IP адрес назначения |
| Source Port (src_port) | Порт источника |
| Destination Port (dst_port) | Порт назначения |
| Average window size | Средний размер рассмотренного набора потоков, *байт* |
| Number of packets | Количество пакетов |
| Packet size | Размер пакетов, *байт* |
| Average packet size | Средний размер пакета, *байт* |
| Standard deviation of packet sizes | Стандартное отклонение размера пакетов, *байт* |
| Average inter-arrival time | Среднее время поступления пакетов, *с* |
| Standard deviation of inter-arrival times | Стандартное отклонение времени поступления пакетов, *мс* |
| Flow duration | Продолжительность потока, *с* |
| Flow size | Размер патока, *байт* |

Будем рассматривать $d$-мерное пространство, в котором координаты точки (элемента) определяются $d$ числами. Будем полагать, что пространство рассматриваемых характеристик трафика является метрическим. Расстояние между двумя точками $x_i$ и $x_j$ (за точку может быть принят поток по результату его мониторинга) определено как

$$S(i,j) = \sqrt{\sum_{r=1}^{d} \left( x_i^{(r)} - x_j^{(r)} \right)^r}. \quad (5)$$

Будем полагать, что значения характеристик потоков (параметров) могут изменяться от некоторого минимального до некоторого максимального значения

$$c_{min}^{(r)} \le x^{(r)} \le c_{max}^{(r)}, r = 1 \dots d. \quad (6)$$

Поскольку характеристики потока могут иметь различные единицы измерения и различные диапазоны возможных значений, будем нормировать их значения

$$\tilde{x}^{(r)} = \frac{1}{c_{max}^{(r)}}\left(\tilde{x}^{(r)} - c_{min}^{(r)}\right), \qquad r = 1 \dots d. \quad (7)$$

Тогда

$$0 \le \tilde{x}^{(r)} \le 1, \quad r = 1 \dots d. \quad (8)$$

Работа алгоритма состоит из двух основных процессов: «обучение» (или адаптация) и классификация потоков. Обучение заключается в выделении заданного количества $k$ кластеров и вычислении их центров масс, т.е. координат центров кластеров, как

$$x_{cm}^{(r)} = \frac{1}{m_r}\sum_{i=1}^{m_r} \tilde{x}_i, \quad r = 1 \dots d. \quad (9)$$

где $\tilde{x}$ нормированное, согласно (8), значение $r$-й характеристики потока.

Выделение кластеров производится согласно алгоритму k-means, т.е. представляет собой итерационную процедуру, в ходе которой производится перераспределение элементов по кластерам и пересчет центров масс, пока центры кластеров не стабилизируются.

Найденные центры масс могут быть использованы в задаче классификации потоков трафика. Данная задача может решаться оценкой степени близости данного потока к центрам масс согласно

$$S(i, w) = \sqrt{\sum_{r=1}^{d}\left(x_i^{(r)} - x_{cm,w}^{(r)}\right)^2}. \quad (10)$$

Принадлежность данного потока некоторому типу потоков (кластеру) может быть определено как

$$\hat{r} = argmin_w S(i, w). \tag{11}$$

В отличие от «классического» алгоритма, в данном случае количество подлежащих кластеризации объектов (потоков) изменяется во времени, т.е. увеличивается в процессе выполнения мониторинга. В начале наблюдений количество объектов мало, и результат кластеризации может быть недостоверным.

Для оценки полученного результата вычисляются среднеквадратические отклонения элементов кластеров от их центров масс

$$\sigma_w = \sqrt{\frac{1}{m_w - 1} \sum_{i=1}^{m_w} S(i, w)^2}. \tag{12}$$

где $S(i, w)$ – расстояние между $i$-м элементом $w$-го кластера и центром масс $w$-го кластера, согласно (10),
$m_w$ – количество элементов в $w$-м кластере.

Также вычисляется общее среднеквадратическое отклонение для всех элементов

$$\sigma = \sqrt{\frac{1}{n - 1} \sum_{i=1}^{n} S(i, x_0)^2}, \tag{13}$$

где $n$ – общее количество элементов;
$S(i, x_0)$ – расстояние между $i$-м элементом и общим центром масс $x_0$.

$$x_0^{(r)} = \frac{1}{n} \sum\nolimits_{i=1}^{n} \tilde{x}_i^{(r)}, \;\; r = 1 \dots d, \tag{14}$$

Сравнение $\sigma_w$ и $\sigma$ позволяет судить о качестве решения задачи кластеризации потоков. Чем меньше величина

$$\delta_w = \frac{\sigma_w}{\sigma}, \qquad w = 1 \dots k, \tag{15}$$

тем меньше разброс элементов внутри кластера $w$, по сравнению с разбросом между всеми элементами без разделения на кластеры.

Таким образом, с помощью (12) и (15) можно характеризовать решение об отнесении потока к некоторому кластеру (типу).

Целесообразно ввести некоторое пороговое значение $\delta_0$, которое свидетельствует о возможности принятия решения. Иначе говоря, решение об отнесении потока к некоторому типу $w$ может быть принято только тогда, когда $\delta_w \leq \delta_{w0}, w = 1 \dots k$. Величина порогового значения может выбираться эмпирически, на основе собранных данных мониторинга.

Сравнение $\sigma_w$ с $S(i, w)$ позволяет оценить степень близости $i$-го потока к потокам выбранной группы. Чем меньше величина

$$\eta_{w,i} = \frac{S(i, w)}{\sigma_w}. \tag{16}$$

тем больше уверенность, что $i$-й поток относится к типу $w$. Согласно правилу $3\sigma$ можно сказать, что если эта величина менее 1/3, то вероятность того, что поток относится к типу $w$ – не менее 0,99. Однако на практике такие оценки не всегда применимы, поэтому для этой величины также целесообразно выбрать некоторое эмпирическое значение $\eta_0$ и принимать решение, если $\eta_{w,i} \leq \eta_0$.

Таким образом, модификация алгоритма k-means состоит в определении размерности пространства, правил оценки численных характеристик и способа оценки качества принимаемого решения.

Эффективность данного метода по сравнению с «классическим» алгоритмом состоит в возможности учета различных характеристик трафика и оценки качества решения, что «классический» алгоритм не позволяет сделать. Это выражается в снижении ошибки принятия ошибочных решений классификации потоков.

### *3.5.3. Модель классификации и приоритизации трафика ПКС*

Модельная сеть состоит из ПКС-приложения, которое классифицирует сетевой трафик и принимает решения о приоритизации трафика, клиентских

агентов (хост-устройств) с приложениями, генерирующими сетевой трафик и маршрутизаторы, применяющих правила приоритетов трафика к активным потокам (Рисунок 42).

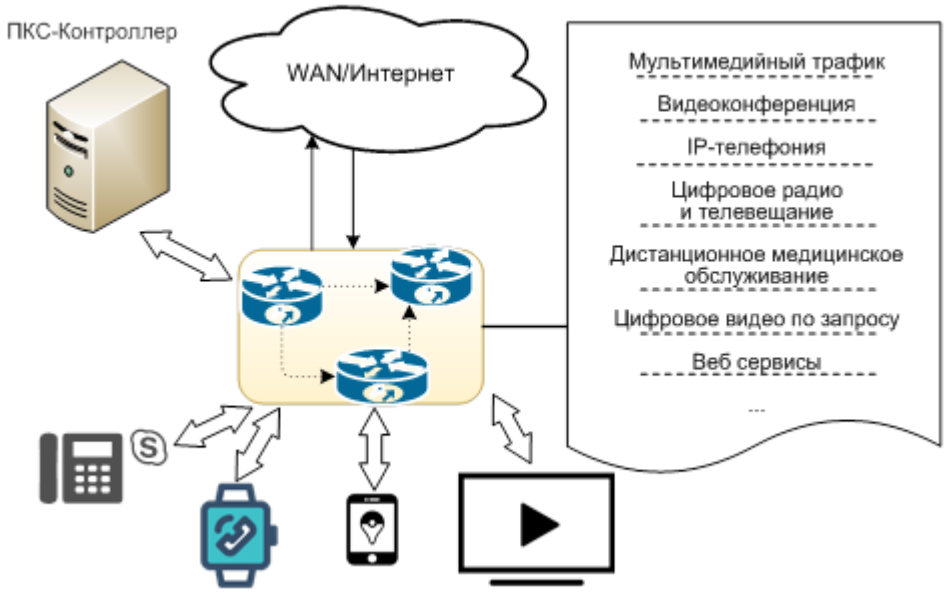


**Рисунок 42** – Модель ПКС для задачи классификации трафика

Предлагаемый метод классификации и приоритизации трафика в ПКС работает следующим образом. Для классификации сначала составляется набор данных, из которого извлекаются признаки потоков конкретного типа трафика. Затем выбирается минимальный набор признаков, которые с высокой точностью характерны для потока; после чего применяется алгоритм классификации для обучения классификатора, который в дальнейшем используется в сценарии в реальном времени.

Поскольку одни и те же «обучающие» данные могут быть неэффективными после продолжительного периода времени, т. к. признаки потоков меняются в течение определенного периода времени, необходимо регулярно обновлять базу данных, т. е. переобучать классификатор. Тем самым можно с точностью определить изменения в характеристиках трафика. Механизм динамической классификации схематично приведен на Рисунке 43.


**Рисунок 43** – Механизм классификации и приоритизации трафика

В процессе классификации в режиме реального времени каждый поток классифицируется как поток мультимедиа или поток загрузки, продолжается добавление значений признаков и вектора класса во временной файл, и только после того, как заданное количество потоков было классифицировано, полученные значения добавляются в набор признаков в существующих данных для переобучения модели. На Рисунке 44 представлена «тепловая карта» корреляций признаков потока, где поле class представляет класс в которую классифицируется поток, в данном контексте класс мультимедиа или класс загрузки.

Соотношение между ожидаемыми значениями признаков трафика и реальными для классификации подтверждают эффективность работы предложенного метода, где для рассмотренного мультимедийного трафика и трафика загрузки точность классификации патока составляет около 98%. Данная карта корреляций признаков потока является критичной для задачи приоритизации классифицированного типа трафика. В случае неправильной классификации потока, соответственно, ошибочно к ним будут применены правила приоритизации.

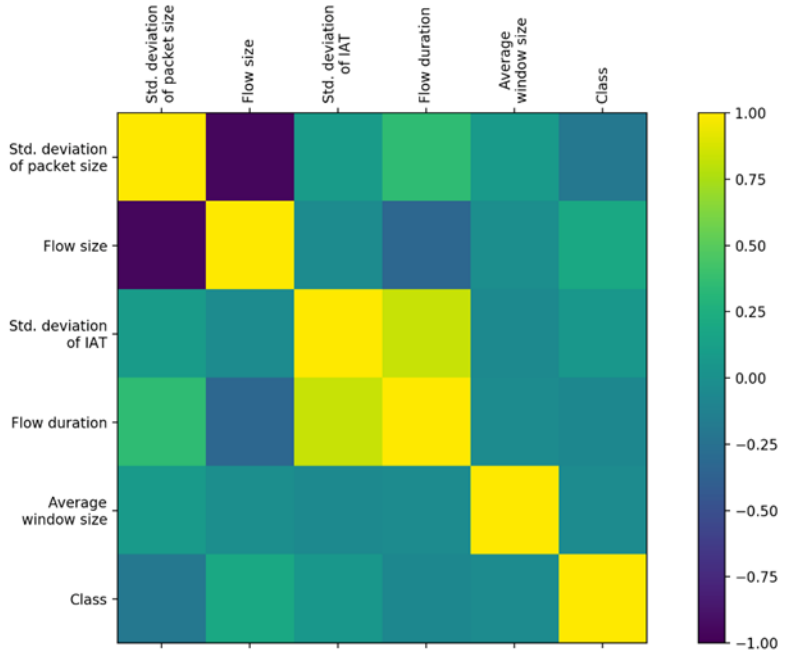


**Рисунок 44** – «Тепловая» карта корреляций признаков потока

На данном этап принимается решение о приоритизации полученного типа трафика для улучшения качества обслуживания.

## Выводы по главе 3

1. Проведен сравнительный анализ методов динамического распределения контроллеров в программно-конфигурируемых сетях.

2. Предложен метод распределения ПКС-контроллеров по алгоритму DDC для кластеризации ресурсов сети.

3. Проведено моделирование в среде моделирования Matlab и выявлены результаты эффективной работы предлагаемых алгоритмов.

4. Предложен метод классификации трафика в программно-конфигурируемых сетях на основе модифицированного алгоритма k-means. Модификация алгоритма k-means состоит в определении размерности пространства, правил оценки численных характеристик и способа оценки качества принимаемого решения.

5. Разработана модель классификации и приоритизации трафика ПКС, которая была апробирована на базе модельной сети.

# Глава 4. Разработка модели идентификации и приоритизации трафика Интернета вещей на основе сегментации ресурсов в программно-конфигурируемых сетях

## 4.1. Концепция сетевой сегментации

Будущие мобильные сети связи должны упрощать процесс настройки сети по требованиям к качеству конкретных предоставляемых услуг, в частности установление требуемой скорости передачи данных, задержки, надежности, безопасности и других сервисов для различных категорий пользователей [21]. Для этого в сетях связи 5G/IMT-2020 введена технология сетевой сегментации. Концепция сетевой сегментации в сетях связи была раньше предложена в контекстах архитектур распределенных сервисов и приложений; но его использование в мобильных сетях связи является новым [9]. С точки зрения оператора мобильной связи, сетевой сегментации – это процесс создания по потребности пользователям, набор логически независимых сетей (сетевые сегменты), размещенных по общей физической инфраструктуре; каждая из которых настроена для представления специфически определенных сетевых услуг (Рисунок 45).

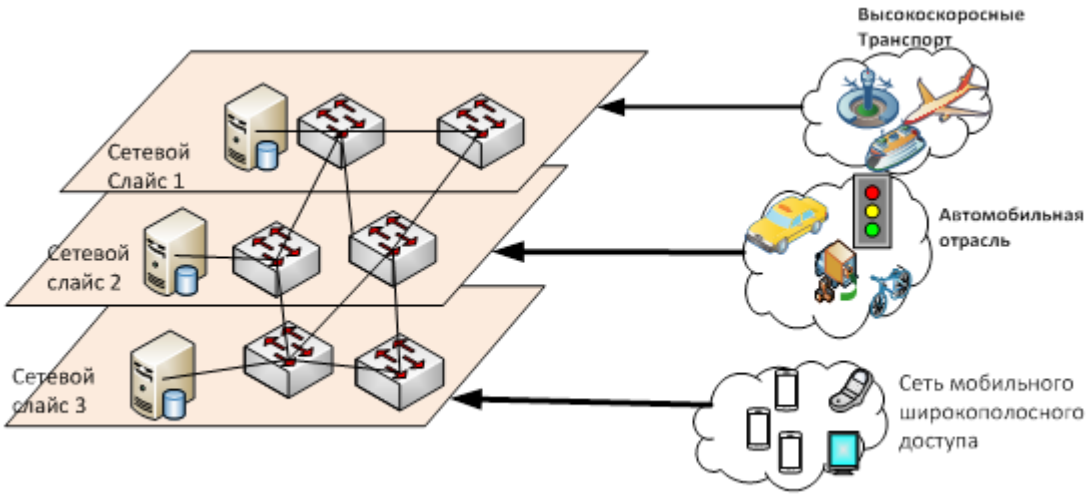


**Рисунок 45** – Архитектура сетевой сегментации в сетях связи 5G/IMT-2020

В контексте ПКС, сетевой сегмент может быт представлен как виртуальная сеть, работающая независимо от ее физической инфраструктурной сети. К примеру, виртуальный сетевой мост между виртуальным узлом сети и его

физическим узлом. Виртуальный узел сети может выполнить специальные ряды сетевых услуг как обычный физический сетевой узел (маршрутизатор, брандмауэр, или сервер сетевых услуг). Установление виртуальных сетевых мостов может быть осуществлено ПКС-коммутатором. Технология программно-конфигурируемых сетей позволяет сетевому администратору управлять физической сетью для того, чтобы выделить необходимые ресурсы созданному сетевому сегменту. А виртуальный сетевой узел может быть установлен при использовании технологии виртуализации сетевых функций. Технологии программно-конфигурируемых сетей и виртуализации сетевых функций позволяют сегментации в сетях связи удовлетворять требованиям по высокой степени гибкости сети [9; 75].

## 4.2. Сравнительный анализ моделей и методов сетевой сегментации

Прежде всего, представим базовую физическую сетевую инфраструктуру как ориентированный мульти-граф $G = (N, L)$, где $N$ – набор узлов сети, а $L$ – набор каналов связи между узлами. Узел $n \in N$ характеризуется вычислительной мощностью $p$, измеряется в флопс/с (flops/s); канал связи $l \in L$ характеризуется пропускной способностью $b > 0$ и измеряется в битах/с (bits/s). Узел с положительной вычислительной мощностью $p > 0$ представляет собой облачное местоположение с возможностью выполнять виртуальные сетевые функции, тогда как узел с нулевой вычислительной мощностью $p = 0$ служит только как маршрутизатор в ПКС [71]

С точки зрения моделирования сетевые сегмента могут быть представлены как коллекции взаимосвязанных виртуальных сетевых функций. Наиболее распространенным подходом в литературе является рассмотрение виртуальных сетевых функций как единицы вычисления [71].

После изучения и анализа доступных на сегодня документов и различного рода материалов автором выделены следующие основные модели сетевой сегментации.

### *4.2.1. Единая эталонная модель*

В единой эталонной модели сетевой сегмент рассматривается как объединение подмножеств сетевых ресурсов и виртуальных сетевых функций на данное время [31; 49].

Сетевая сегментация в этом случае представляет собой создание и управление такими сегментами поверх физической сетевой инфраструктуры. Схематически данная модель представлена на Рисунке 46.

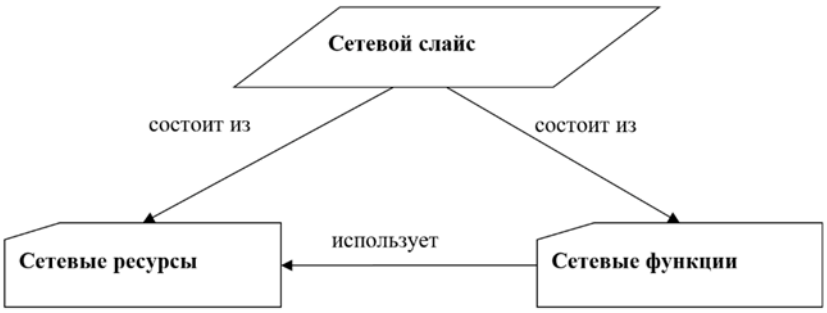


**Рисунок 46** – Эталонная модель сетевого сегмента

Математически сегмент единой эталонной модели описывается следующим кортежем:

$$s = \langle NR, NF, R \rangle$$

где $s$ – сетевой сегмент;
$NR$ – множество сетевых ресурсов;
$NF$ – множество сетевых функций;
$R$ – множество отношений, связывающих сетевые ресурсы и сетевые функции между собой.

В этой модели существует отношение «использует», устанавливающее факт использования сетевой функцией данного сетевого ресурса. Если примем, что ноль соответствует «функция не использует ресурса», а единица соответствует «функция использует ресурс», то в математической записи для отношения «использует» получим следующее отображение:

$$\text{«использует»}: NF \times NR \to \{0,1\}.$$

Для статистического анализа степени вовлеченности сетевого ресурса в работу сегмента или учет сетевых ресурсов, связанных с данной сетевой функцией или для чего-то ещё, можно отношение «использует» представить следующей двухмерной матрицей. Благо количество сетевых ресурсов и сетевых функций

ограничено. В матрице строки соответствуют сетевым функциям, а столбцы сетевым ресурсам. В ячейку $(i, j)$ матрицы записываем 1 или 0 в зависимости от того используется или нет соответственно $j$-й сетевой ресурс $i$-й сетевой функцией. Получим матрицу следующего вида:

$$U = \begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,n} \end{bmatrix},$$

где $m$ – количество сетевых функций;
$n$ – количество сетевых ресурсов,
$b_{i,j} \in \{0,1\}, i = \overline{1,m}, j = \overline{1,n}$.

Каждая $i$-я строка в матрице $U$ – это вектор, соответствующий сетевой функции и отражающий как она использует сетевые ресурсы:

$$R_i^T = \begin{bmatrix} b_{i,1} & \cdots & b_{i,n} \end{bmatrix}.$$

Аналогично, каждый $j$-й столбец представляет собой вектор, соответствующий сетевому ресурсу и отражающий как он используется сетевыми функциями:

$$C_j = \begin{bmatrix} b_{1,j} \\ \vdots \\ b_{m,j} \end{bmatrix}$$

Теперь произведение $R_i^T R_k$ двух векторов двух сетевых функций дает корреляцию этих сетевых функций при работе сегмента. Матричное произведение $UU^T$ содержит все такие произведения векторов сетевых функций. Элемент $(i, k)$ (тоже, что и элемент $(k, i)$) содержит произведение $R_i^T R_k (= R_k^T R_i)$. Матрица $U^T U$ содержит все произведения векторов сетевых ресурсов, обеспечивая их корреляцию по сетевым функциям: $C_j^T C_l (= C_l^T C_j)$.

Теперь предположим, что существует сингулярное разложение (SVD) [26] $U$ такое, что $W$ и $V$ являются ортогональными матрицами, и $\Sigma$ – диагональная матрица. То есть:

$$U = W \Sigma V^T$$

Тогда матричные произведения будут

$$UU^T = (W\Sigma V^T)(W\Sigma V^T)^T = (W\Sigma V^T)\left(V^{T^T}W^T\Sigma^{\mathrm{T}}\right) = W\Sigma V^T V W^T\Sigma^{\mathrm{T}} = W\Sigma W^T\Sigma^{\mathrm{T}},$$

$$U^T U = (W\Sigma V^T)^T(W\Sigma V^T) = \left(V^{T^T}W^T\Sigma^{\mathrm{T}}\right)(W\Sigma V^T) = VW^T\Sigma^{\mathrm{T}}W\Sigma V^T = V\Sigma^{\mathrm{T}}\Sigma V^T.$$

Так как $\Sigma\Sigma^T$ и $\Sigma^T\Sigma$ являются диагональными, то мы видим, что $W$ должен содержать собственные векторы $UU^T$, в то время как $V$ содержит собственные векторы $U^TU$. Оба произведения имеют те же самые собственные значения, отличные от нуля, заданными записями, отличными от нуля $\Sigma\Sigma^T$, или равно, записями, отличными от нуля $\Sigma^T\Sigma$. Теперь сингулярное разложение станет:

$$\begin{matrix} & & U & & & & W & & & & V^T \\ & & (C_j) & & & & & & & & (D_j) \\ & & \downarrow & & & & \downarrow & & & & \downarrow \\ (R_i^T) & \rightarrow & \begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,n} \end{bmatrix} & = & (S_i^T) & \rightarrow & \left[\begin{bmatrix} w_1 \end{bmatrix}\cdots\begin{bmatrix} w_l \end{bmatrix}\right] & \cdot & \begin{bmatrix} \delta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \delta_l \end{bmatrix} & \cdot & \begin{bmatrix} [ & v_1 & ] \\ & \vdots & \\ [ & v_l & ] \end{bmatrix} \end{matrix}$$

Значения $\delta_1, \cdots, \delta_l$ называются сингулярными значениями, а $w_1, \cdots, w_l$ и $v_1, \cdots, v_l$ – левые и правые сингулярные векторы, соответственно. Отметим, что единственной частью $W$, которая вносит вклад в $R_i$, является $i$-я строка. Пусть эта строка будет $S_i$. Аналогично, единственной частью $V^T$, которая вносит вклад в $C_j$, является $j$-й столбец $D_j$. Они – не собственные векторы, но *зависят от всех* собственных векторов. Оказывается, что, когда выбираем $k$ наибольших сингулярных значений, и их соответствующие сингулярные векторы от $W$ и $V$, то получаем ранг $k$ аппроксимацию к U с наименьшей ошибкой (норма Frobenius). Эта аппроксимация имеет минимальную ошибку. Можно теперь рассмотреть векторы сетевых функций и векторы сетевых ресурсов как «пространство элементов». Вектор $S_i$ тогда имеет $k$ записей, каждая из которых описывает появление $i$-й сетевой функции в одном из $k$ элементов. Аналогично, вектор $D_j$ описывает отношение между j-м сетевым ресурсом и каждым элементом. Данная аппроксимация определяется как

$$U_k = W_k\Sigma_{\mathrm{k}}V_k^T.$$

Мы можем теперь сделать следующее:

1) рассмотреть связи между сетевыми ресурсами $j$ и $q$, сравнивая векторы $D_j$ и $D_q$. Это обеспечивает автоматическую кластеризацию сетевых ресурсов;

2) сравнить сетевые функции i и $p$, сравнивая векторы $S_i$ и $S_p$, обеспечивая автоматическую кластеризацию сетевых функций;

3) оценить рейтинги и эффективность использования сетевых ресурсов;

4) сделать поиск сетевых функций и ресурсов в пространстве элементов;

Для осуществления поиска нужно сначала перевести запрос в пространство элементов, затем выполнить те же действия, что и ранее:

$$C_j = W_k \Sigma_{\mathrm{k}} D_j \, , \, D_j = \Sigma_{\mathrm{k}}^{-1} W_k^T C_j .$$

Это означает, что, если имеем вектор запроса $g$, то мы должны сделать перевод $\breve{\mathrm{g}} = \Sigma_{\mathrm{k}}^{-1} W_k^T g$ прежде, чем сравниваем этот запрос с векторами сетевых ресурсов в пространстве элементов. Можно сделать то же самое для псевдовекторов сетевых функций:

$$R_i^T = S_i^T \Sigma_{\mathrm{k}} V_k^T , S_i^T = R_i^T V_k^{-T} \Sigma_{\mathrm{k}}^{-1} = R_i^T V_k \Sigma_{\mathrm{k}}^{-1} , S_i = R_i V_k^T \Sigma_{\mathrm{k}}^{-1} .$$

Переходим к рассмотрению модифицированной эталонной модели.

### *4.2.2. Модифицированная эталонная модель*

Данная модель расширяет эталонную модель, вынеся сетевые сервисы отдельно. То есть сегмент рассматривается как объединение подмножеств сетевых ресурсов, виртуальных сетевых функций и сетевых сервисов на данное время. Каждый сегмент имеет свое собственное управление [37; 80; 94].

Сетевая сегментация в этом случае представляет собой процесс создания и управления этими расширенными сегментами поверх физической сетевой инфраструктуры. Схематически данная модель представлена на Рисунке 47.

Математически сетевой сегмент модифицированной эталонной модели описывается следующим кортежем:

$$s = \langle MF, NR, NF, NS, R \rangle,$$

где $s$ – сетевой сегмент;
$MF$ – множество функций управления;

$NR$ – множество сетевых ресурсов;
$NF$ – множество сетевых функций;
$NS$ – множество сетевых сервисов;
$R$ – множество отношений, связывающих сетевые ресурсы и сетевые функции, а также сетевые ресурсы и сетевые сервисы.

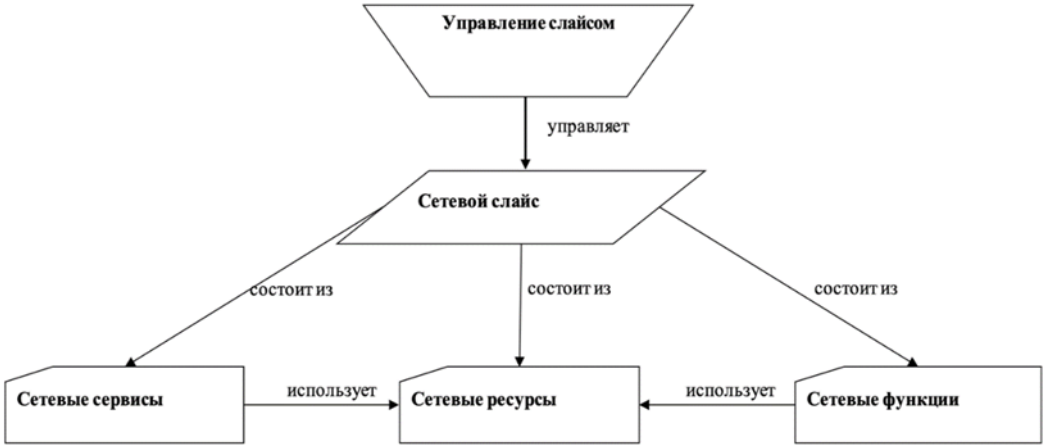


**Рисунок 47** – Модифицированная эталонная модель семента

### *4.2.3. Модель сегмента с одной S/D парой*

Предполагается сосуществование на основе одной физической сетевой инфраструктуры нескольких сетевых сегментов. В этой модели каждый сегмент обеспечивает сетевой трафик между только одним источником (Source) и только одним приемником (Destination), т.е. сегмент представляется source-destination (S/D) парой [71].

Определение каждого сегмента включает в себя однонаправленный связанный список виртуальных сетевых функций (VNFs), который определяет последовательность выполнения различных VNFs.

Сетевая сегментация в таком случае представляет собой создание и управление такими сегментами поверх физической сетевой инфраструктуры. Схематически данная модель представлена на Рисунке 48.

Математически сегмент этой модели описывается следующим кортежем:

$$s = \langle S, D, NR, NFC, R \rangle,$$

где $s$ – сетевой сегмент;
$S$ – источник трафика;
$D$ – приемник трафика;
$NR$ – множество сетевых ресурсов;
$NFC$ – последовательность (связанный список) сетевых функций;
$R$ – множество отношений, связывающих элементы модели между собой.

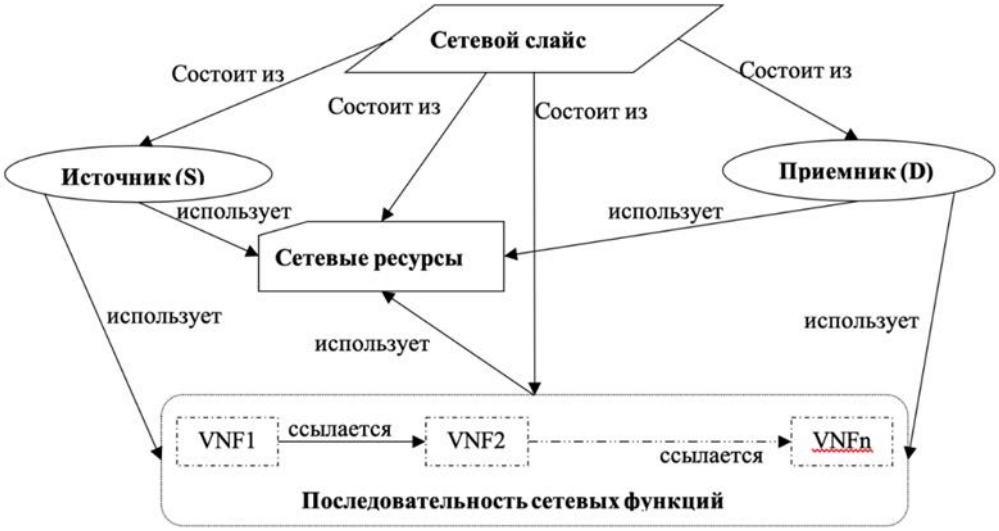


**Рисунок 48** – Модель одиночной S/D пары

В этой модели существует отношение «использует», устанавливающее факт использования сетевой функцией данного сетевого ресурса. А также модель содержит отношение «ссылается», устанавливающее факт ссылки одной сетевой функции на другую. Примем, что ноль соответствует «функция не использует ресурса», а единица соответствует «функция использует ресурс»; а также примем, что ноль соответствует «данная функция не ссылается на эту», а единица соответствует «данная функция ссылается на эту», то в математической форме для отношения «использует» получим следующее отображение:

$$\text{«использует»}: NF \times NR \to \{0,1\},$$

а для отношения «ссылается» получим следующее отображение:

$$\text{«}ссылается\text{»}: NF \times NF \to \{0,1\}.$$

Далее рассуждая как в п. 1.1, составим для этих двух отношений две двухмерные матрицы, в одной из которых строки соответствуют сетевым функциям, а столбцы сетевым ресурсам; а в другой матрице строки и столбцы соответствуют сетевым функциям. В ячейку $(i, j)$ первой матрицы записываем 1 или 0 в зависимости от того используется или нет соответственно сетевой ресурс $i$-й сетевой функцией. А в ячейку $(i, j)$ второй матрицы записываем 1 или 0 в зависимости от того ссылается или нет соответственно $i$-й сетевая функция на $j$-ю сетевую функцию. Получим матрицы следующего вида:

$$M = \begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,n} \end{bmatrix},$$

где $m$ – количество сетевых функций;
$n$ – количество сетевых ресурсов или сетевых функций в зависимости от отношения модели;
$b_{i,j} \in \{0,1\}, i = \overline{1,m}, j = \overline{1,n}$.

Для матрицы отношения «ссылается» $m = n$, т.е. у нас квадратная матрица.

Для сетевого сегмента $s$ этой модели также можно определить объём генерируемого им трафика и требуемую для обработки этого трафика вычислительную мощность.

Обозначим через $V_s$ объём генерируемого сегментом трафика, а через $P_s$, требуемую для обработки этого трафика вычислительную мощность. Обозначим также через $W_s$ множество путей, по которым отдельные части трафика сегента $s$ передаются от источника (S) к приемнику (D). Для части объёма трафика, проходящей через путь $w \in W_s$, используем обозначение $V_{sw}$. Тогда по определению имеем

$$V_s = \sum_{w \in W_s} V_{sw} .$$

Вычислительная мощность сегмента может быть определена через вычислительные мощности составляющих его узлов маршрутизации трафика. Пусть $P_{sn}$ – вычислительная мощность узла $n$, через который проходит трафик сегмента $s$, тогда по определению имеем

$$P_s = \sum_{n \in N_s} P_{sn} .$$

### *4.2.4. Модель сегмента со многими S/D парами*

В этой модели каждый сегмент обеспечивает сетевой трафик между множеством источников (Sources) и множеством приемников (Destinations), т.е. сегмент представляется набором source-destination (S/D) пар.

Данная модель является развитием предыдущей. Она берет модель одной S/D пары за основу и просто размножает её. Каждый сегмент модели со многими S/D парами состоит из сегментов предыдущего типа, т.е. типа одной S/D пары.

Сетевая сегментация в данном случае представляет собой создание и управление такими сегментами поверх физической сетевой инфраструктуры. Схематически данная модель представлена на Рисунке 49.

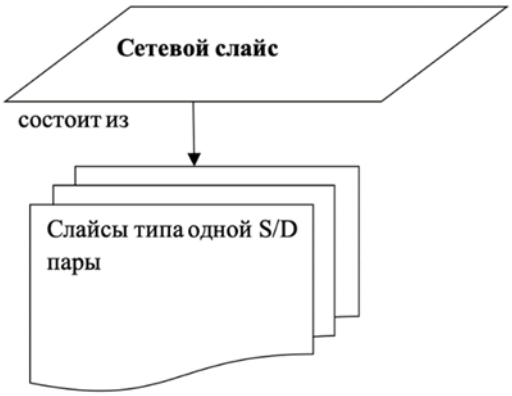


**Рисунок 49** – Модель со многими S/D парами

Математически сегмент этой модели описывается множеством сегментов одной S/D пары:

$$s = \{x | \; x \; это \; сегмент \text{ типа одной S/D пары}\},$$

где $s$ – сетевой сегмент этой модели.

В данном случае, для внутренних сегментов справедливо все, что описано выше по отношению к сегментам типа одной S/D пары. Еще можно рассмотреть появление нового отношения «состоит из». Это отношение указывает на состав внутренних сегментов из сетевых ресурсов и сетевых функций, что позволяет вводить следующие два отображения для данного отношения:

1) «*состоит из*»: $X \times NR \rightarrow \{0,1\}$,

2) «*состоит из*»: $X \times NF \rightarrow \{0,1\}$.

Можно также рассмотреть отношения как между источниками (sources), приемниками (destinations), так и межу источниками и приемниками разных внутренних сегментов.

Для сетевого сегмента $s$ этой модели также можно определить объём генерируемого им трафика и требуемую для обработки этого трафика вычислительную мощность.

Обозначим через $V_s$ объём генерируемого сегментом трафика, через $V_{si}$ объём генерируемого внутренним сегментом трафика, через $P_s$ требуемую для обработки трафика основного сегмента вычислительную мощность, а через $P_{si}$ требуемую для обработки трафика внутреннего сегмента вычислительную мощность. Тогда по определению имеем

$$V_s = \sum V_{si} \, , P_s = \sum P_{si}.$$

Таким образом трафик и требуемая вычислительная мощность внутреннего сегмента (сегмента одной S/D пары) определяются формулами:

$$V_{si} = \sum_{w \in W_{si}} V_{siw} \, , P_{si} = \sum_{n \in N_{si}} P_{sin} \, ,$$

поэтому объём трафика и требуемая вычислительная мощность основного сегмента (сегмент многих S/D пар) определяются формулами:

$$V_s = \sum_{si} \sum_{w \in W_{si}} V_{siw} \, , P_s = \sum_{si} \sum_{n \in N_{si}} P_{sin}.$$

### *4.2.5. Контент-ориентированная модель*

Благодаря разработке интеллектуальных мобильных устройств и различных мобильных приложений контент-ориентированный сервис стал самым популярным сервисом, который занимает сетевые ресурсы и приводит к высокой нагрузке на трафик [60]. Поскольку пользователи обычно интересуются самим контентом, а не местом хранения контента, то в качестве новой парадигмы предлагается информационно-ориентированная сеть (ICN – information-centric networking) [61]. В ICN пользователь может загружать свой контент в кэш маршрутизаторов сети и в то же время скачать необходимый контент из этого кеша в режиме реального времени.

В контент-ориентированном (или информационно-ориентированном) сетевой сегментации сетевые ресурсы выделяются именно контенту. Так появляются контент-ориентированные ресурсы, которые включают ресурсы

кеширования и коммуникационные ресурсы, которые используются для передачи контента.

В этой модели каждый сетевой сегмент связан только с одним конкретным контентом и состоит из двух внутренних подсегментов: сегмент кеширования и сегмент коммуникации.

Сетевой сегмент в этом случае представляет собой процесс создания и управления информационно-ориентированными сегментами для каждого контента поверх физической сетевой инфраструктуры. Схематически данная модель представлена на Рисунке 50.

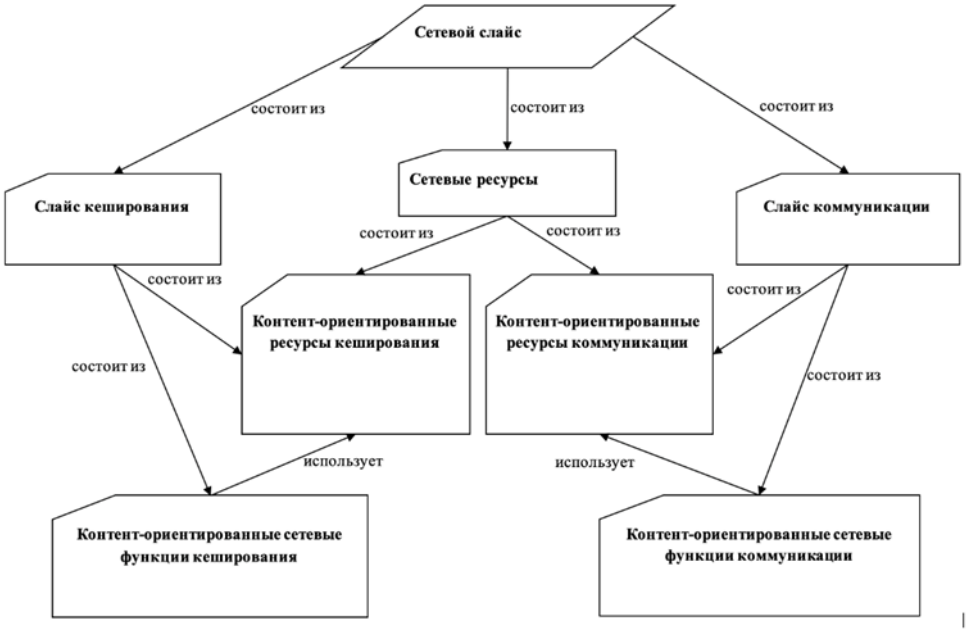


**Рисунок 50** – Контент-ориентированная модель сегмента

В данной модели существуют только один контент-ориентированный сервис, и поэтому сетевые функции тоже только контент-ориентированные.

Математически контент-ориентированный (или информационно-ориентированный) сегмент описывается следующим кортежем:

$$s = \langle s1, s2, NR, COCNR, COComNR, COCNF, COComNF, R \rangle,$$

где $s$ – контент-ориентированный сетевой сегмент;
$s1$(внутренний сегмент сетевого сегмента) – контент-ориентированный сегмент кеширования;
$s2$ (внутренний сегмент сетевого сегмента) – контент-ориентированный сегмент коммуникации;
$NR$ – множество всех сетевых ресурсов;
$COCNR \subset NR$ – множество контент-ориентированных ресурсов кеширования;
$COComNR \subset NR$ – множество контент-ориентированных ресурсов коммуникации;
$COCNF$ – множество контент-ориентированных функций кеширования;

$COComNF$ – множество контент-ориентированных функций коммуникации;
$R$ – множество отношений, связывающих контент-ориентированные сетевые ресурсы и контент-ориентированные сетевые функции, а также внутренние сегменты и контент-ориентированные сетевые ресурсы и функции.

Далее, как везде выше, можно вводить двумерные матрицы и предполагая, что существуют для них сингулярные разложения (SVD), рассуждать, как это сделано ранее.

### *4.2.6. Номинальная модель*

Рассматривается субстратная (базовая) физическая сетевая инфраструктура как неориентированный граф $G = (N_s, L_s)$, где $N_s$ – набор узлов сети, а $L_s$ – набор связей между узлами [21]. Узел $n_s \in N_s$ характеризуется вычислительной мощностью $c_{n_s} > 0$, измеряется в флопс/с (flops/s); связь $l_s \in L_s$ характеризуется пропускной способностью $c_{l_s} > 0$ и измеряется в битах/с (bits/s). Пусть $p_{n_s}$ – стоимость полного использования ресурсов субстратного узла $n_s$, а $p_{l_s}$ – стоимость полного использования ресурсов субстратной связи $l_s$. Для каждой субстратной линии связи $l_s$ сети предполагается задержка распространения $\Delta_{l_s}$, пропорциональная ее длине.

Для сетевого сегмента, запрос сквозного трафика t обозначаем $d_t \in \mathbb{R}_+$, $t \in T$ должен быть маршрутизирован через набор $N_v$ виртуальных сетевых функций. Эти виртуальные сетевые функции предоставляются субстратными сетевыми узлами. Пусть теперь:

- ✓ $\delta_{n_s}^{n_v} \in \{0, 1\}$ – способность субстратного узла $n_s$ размещать соответствующую виртуальную функцию $n_v$. Аналогичным образом, виртуальные связи $l_v \in L_v$ устанавливаются между виртуальными функциями;
- ✓ $x_{n_s}^{t,n_v} \in \{0, 1\}$ – указывает, обрабатывается / маршрутизируется ли запрос трафика $d_t$ через виртуальную функцию $n_v$, находящуюся на субстратном узле $n_s$;

- ✓ $f_{l_s}^{t,l_v} \in \{0,1\}$ – указывает, маршрутизируется ли по физической линии связи $l_s$ запрос трафика $d_t$ на виртуальной линии связи $l_v$ между виртуальными функциями $(n_{v1}, n_{v2}) \in L_v$.

Этим определением подразумевается, что трафик маршрутизируется по одному пути между источником и получателем.

Также пусть:

- ✓ $y_{n_s}^{n_v} \in \mathbb{Z}_{\geq 0}$ – указывают количество модулей мощности размера k, выделенное под виртуальной функции $n_v \in N_v$, находящейся на субстратном узле $n_s \in N_s$;
- ✓ $u_{n_s}, u_{l_s} \in \mathbb{R}_{[0,1]}$ – обозначают использование ресурсов физического субстратного узла и линии связи соответственно.

Тогда математическая формулировка общей задачи проектирования сетевого сегмента, следующая [21]:

$$\min \sum_{n_s \in N_s} p_{n_s} u_{n_s} + \sum_{l_s \in L_s} p_{l_s} u_{l_s} \qquad (1a)$$

так чтобы:

$$\sum_{n_s \in N_s} \delta_{n_s}^{n_v} x_{n_s}^{t,n_v} = 1 \quad \forall t \in T, n_v \in N_v \qquad (1b)$$

$$\sum_{t \in T} d_t x_{n_s}^{t,n_v} \leq k y_{n_s}^{n_v} \quad \forall n_s \in N_s, n_v \in N_v \qquad (1c)$$

$$\sum_{n_v \in N_v} k y_{n_s}^{n_v} \leq c_{n_s} u_{n_s} \qquad (1d)$$

$$\sum_{(n_s,w) \in L_s} f_{(n_s,w)}^{t,l_v} - f_{(w,n_s)}^{t,l_v} = x_{n_s}^{t,n_{v1}} - x_{n_s}^{t,n_{v2}} \qquad (1e)$$

$$\sum_{t \in T} \sum_{l_v \in L_v} d_t f_{l_s}^{t,l_v} \leq c_{l_s} u_{l_s} \quad \forall l_s \in L_s \qquad (1f)$$

$$\sum_{l_s \in L_v} \sum_{l_s \in L_s} \Delta_{l_s} f_{l_s}^{t,l_v} \leq b_t \quad \forall t \in T \qquad (1g)$$

$$x_{n_s}^{t,n_v}, f_{l_s}^{t,l_v} \in \{0,1\}; y_{n_s}^{n_v} \in \mathbb{Z}_+; u_{n_s}, u_{l_s} \in \mathbb{R}_{[0,1]} \qquad (1h)$$

Целевая функция (1a) минимизирует затраты на настройку сетевого сегмента над субстратной сетью.

Ограничение (1e) обеспечивает сохранение потока на каждом субстратном узле $n_s \in N_s$, в то время как ограничение (1g) гарантирует, что потоки трафика не будут превышать время распространения (bt).

### *4.2.7. Г-надежная модель*

В Г-надежном подходе неопределенность в отношении требований к трафику представлена подмножеством размера Г отклоненных требований, в отношении которых решение оптимизации должно быть надежным [21]. Г представляет уровень защиты, где Г = 0 означает отсутствие защиты вообще (то есть номинальное требование к трафику), в то время как наибольшее значение Г означает, что все требования к трафику находятся на своих пиковых значениях. Здесь авторы предполагают, что требования к трафику могут быть смоделированы случайными значениями $d^t \in \left[\bar{d}^t - \hat{d}^t, \bar{d}^t + \hat{d}^t\right]$, где $\bar{d}^t$ – номинальные значения, а $\hat{d}^t$ – максимальное отклонение требования к трафику. В Г-надежной модели выражение ограничений пропускной способности (1c) следующее [28]:

$$\sum_{t \in T} \bar{d}^t x_{n_s}^{t,n_v} + \max_{\substack{D \subseteq T, \\ |D| \leq \Gamma}} \sum_{t \in D} \hat{d}^t x_{n_s}^{t,n_v} \leq k y_{n_s}^{n_v}$$

Математически Г-надежная модель записывается следующим образом [19]:

$$\min \sum_{n_s \in N_s} p_{n_s} u_{n_s} + \sum_{l_s \in L_s} p_{l_s} u_{l_s} \qquad (2a)$$

так чтобы:

$$\sum_{n_s \in N_s} \delta_{n_s}^{n_v} x_{n_s}^{t,n_v} = 1 \quad \forall t \in T, n_v \in N_v \qquad (2b)$$

$$\sum_{t \in T} \bar{d}^t x_{n_s}^{t,n_v} + \sum_{t \in T} \rho_{n_s}^{t,n_v} + \Gamma \pi_{n_s}^{n_v} \leq k y_{n_s}^{n_v} \quad \forall t \in T, n_s \in N_s, n_v \in N_v \qquad (2c')$$

$$\rho_{n_s}^{t,n_v} + \pi_{n_s}^{n_v} \geq \hat{d}^t x_{n_s}^{t,n_v} \quad \forall t \in T, n_s \in N_s, n_v \in N_v \qquad (2c'')$$

$$\sum_{n_v \in N_v} k y_{n_s}^{n_v} \leq c_{n_s} u_{n_s} \qquad (2d)$$

$$\sum_{(n_s,w)\in L_s} f_{(n_s,w)}^{t,l_v} - f_{(w,n_s)}^{t,l_v} = x_{n_s}^{t,n_{v1}} - x_{n_s}^{t,n_{v2}} \quad (2e)$$

$$\sum_{t\in T}\sum_{l_v\in L_v} \bar{d}^t f_{l_s}^{t,l_v} + \sum_{t\in T} \rho_{l_s}^t + \Gamma\pi_{l_s} \le c_{l_s} u_{l_s} \quad \forall l_s \in L_s \qquad (2f')$$

$$\rho_{l_s}^t + \pi_{l_s} \ge \sum_{l_v\in L_v} \hat{d}^t f_{l_s}^{t,l_v} \quad \forall t \in T, l_s \in L_s \qquad (2f'')$$

$$\sum_{l_v\in L_v}\sum_{l_s\in L_s} \Delta_{l_s} f_{l_s}^{t,l_v} \le b_t \quad \forall t \in T \qquad (2g)$$

$$x_{n_s}^{t,n_v}, f_{l_s}^{t,l_v} \in \{0,1\}; y_{n_s}^{n_v} \in \mathbb{Z}_+; u_{n_s}, u_{l_s} \in \mathbb{R}_{[0,1]} \quad (2h)$$

Ограничениями (2c'), (2c'') и (2f'), (2f'') номинальная модель преобразуется в Г-надежную модель. Последняя модель обеспечивает высокую надежность, но обычно приводит к довольно консервативному и, следовательно, дорогостоящему решению.

### *4.2.8. Легкая надежная модель*

Авторы в этом случае поставили цель добиться высокой надежности как в предыдущем случае, но только за меньшей ценой. Когда количество требований, которые можно отклонять одновременно, увеличивается, нарушаются ограничения ресурсов [21]. Целевая функция модели заключается в минимизации этих нарушений ограничений пропускной способности.

В отличие от Г-надежной модели, где цена на надежность была подвергнута оптимизации, легкий надежный подход учитывает верхнюю границу цены на надежность в составе модели, что позволяет проектировать сетевые сегменты с учетом затрат. Математическая формулировка данной модели, следующая [21]:

$$\min \sum_{n_v\in N_v}\sum_{n_s\in N_s} \gamma_{n_s}^{n_v} + \sum_{l_s\in L_s} \gamma_{l_s} \qquad (3a)$$

где $\gamma_{n_s}^{n_v}$, $\gamma_{l_s}$ используются в качестве параметров требования к трафику для балансировки задействованного ограничения.

Оптимизация выполняется так чтобы:

$$\sum_{n_s \in N_s} \delta_{n_s}^{n_v} x_{n_s}^{t,n_v} = 1 \quad \forall t \in T, n_v \in N_v \qquad (3b)$$

$$\sum_{t \in T} \bar{d}^t x_{n_s}^{t,n_v} + \sum_{t \in T} \rho_{n_s}^{t,n_v} + \Gamma \pi_{n_s}^{n_v} - \gamma_{n_s}^{n_v} \leq k y_{n_s}^{n_v} \quad \forall n_s \in N_s, n_v \in N_v \qquad (3c')$$

$$\rho_{n_s}^{t,n_v} + \pi_{n_s}^{n_v} \geq \hat{d}^t x_{n_s}^{t,n_v} \quad \forall t \in T, n_s \in N_s, n_v \in N_v \qquad (3c'')$$

$$\sum_{n_v \in N_v} k y_{n_s}^{n_v} \leq c_{n_s} u_{n_s} \qquad (3d)$$

$$\sum_{(n_s,w) \in L_s} f_{(n_s,w)}^{t,l_v} - f_{(w,n_s)}^{t,l_v} = x_{n_s}^{t,n_{v1}} - x_{n_s}^{t,n_{v2}} \qquad (3e)$$

$$\sum_{t \in T} \sum_{l_v \in L_v} \bar{d}^t f_{l_s}^{t,l_v} + \sum_{t \in T} \rho_{l_s}^t + \Gamma \pi_{l_s} - \gamma_{l_s} \leq c_{l_s} u_{l_s} \quad \forall l_s \in L_s \qquad (3f')$$

$$\rho_{l_s}^t + \pi_{l_s} \geq \sum_{l_v \in L_v} \hat{d}^t f_{l_s}^{t,l_v} \quad \forall t \in T, l_s \in L_s \qquad (3f'')$$

$$\sum_{l_v \in L_v} \sum_{l_s \in L_s} \Delta_{l_s} f_{l_s}^{t,l_v} \leq b_t \quad \forall t \in T \qquad (3g)$$

$$\sum_{n_s \in N_s} p_{n_s} u_{n_s} + \sum_{l_s \in L_s} p_{l_s} u_{l_s} \leq (1 + \sigma) z^* \qquad (3h)$$

В этом подходе вводятся переменные $\gamma_{n_s}^{n_v}$ и $\gamma_{l_s}$ (в ограничениях (3c') и (3f')), чтобы допускать нарушения ограничений пропускной способности. Кроме того, предоставляется новое ограничение (3h), которое дает верхнюю границу для ухудшения целевой функции.

В (3h) параметр z* относится к оптимальному решению номинальной задачи (ограничения (1a) до (1h) см. 1.6.), а σ относится к дополнительной цене, которую мы готовы заплатить за устойчивость к неопределенностям трафика.

Целевая функция (3a) минимизирует нарушение пропускной способности узла и ограничений пропускной способности канала ((3c'), (3c'') и (3f''), (3g))

### *4.2.9. Сравнительный анализ моделей сетевой сегментации*

В таблице ниже приведены основные недостатки и достоинства рассмотренных выше моделей и подходов. Данное сравнение основывается на сопоставлении недостатков и преимуществ моделей и подходов между собой.

**Таблица 6** – Сравнительный анализ моделей и методов сетевой сегментации

| *Основные модели сетевых сегментов* | *Основные недостатки модели* | *Основные достоинства модели* |
|---|---|---|
| **Эталонная модель** | – не представлен явно слой сетевых сервисов,<br>– нет четкого разделения плоскостей управления и трафика,<br>– не указаны четкие методы оптимизации большинства параметров | – модель проста в изучении,<br>– простота реализации и развертывания.<br>– универсальность,<br>– возможно большое количество различных предоставляемых сервисов,<br>– возможно большое количество участников трафика,<br>– множество путей связи между участниками трафика,<br>– поддерживает любой вид трафика,<br>– высокая гибкость,<br>– высокая масштабируемость,<br>– модель устойчива к неопределенностям требований к трафику.<br>– модель имеет и семантическое и математическое представление,<br>– возможно введение в модель разных отношений между компонентами для разностороннего контроля и мониторинга сегментов |
| **Модифицированная Эталонная модель** | – не указаны четкие методы оптимизации большинства параметров | – все достоинства эталонной модели,<br>– представлен явно слой сетевых сервисов,<br>– явное и четкое разделение плоскостей управления и трафика,<br>– легко преобразуется в онтологическую модель сегмента |
| **Модель одной S/D пары** | – в процессе обмена сообщениями участвуют только две стороны,<br>– ограниченное количество сервисов,<br>– жестко задана последовательность выполнения виртуальных сетевых функций | – простота реализации и развертывания,<br>– высокая скорость,<br>– допускает естественное разделение по отношению к различным заинтересованным сторонам,<br>– модель устойчива к неопределенностям требований трафика,<br>– модель имеет и семантическое и математическое представление,<br>– возможно введение в модель разных отношений между компонентами для разностороннего контроля и мониторинга сегментов |
| **Модель многих S/D пар** | – жестко задана последовательность выполнения виртуальных сетевых функций во внутренних сегментах,<br>– не очень хорошая гибкость. | – все достоинства модели одной S/D пары,<br>– много S/D пар могут быть ассоциированы к одному сегменту,<br>– хорошая изоляция трафика,<br>– есть алгоритм автомасштавирования сегментов в реальном масштабе времени. |

Продолжение таблицы 6

| *Основные модели сетевых сегментов* | *Основные недостатки модели* | *Основные достоинства модели* |
|---|---|---|
| **Контент-ориентированная модель** | – модель ориентирована только на контент, т.е. обеспечивает только один сервис,<br>– требует дополнительных ресурсов для кеширования | – модель очень надежна в информационно-ориентированных сетях,<br>– высокая скорость,<br>– модель имеет и семантическое и математическое представление,<br>– возможно введение в модель разных отношений между компонентами для разностороннего контроля и мониторинга сегментов |
| **Номинальная модель** | – не учитывает отклонения требований к трафику, которые могут привести к перегрузке ресурса в субстратной сети.<br>– модель не пригодна для практической реализации,<br>– недостаточная надежность,<br>– требует много вычислений,<br>– нет семантического представления | – наглядный пример постановки задачи проектирования сетевых сегментов как оптимизационная задача,<br>– хорошая модель для обучения |
| **Г-надежная модель** | – консервативное и дорогостоящее решение,<br>– не учитывает затрат на надежность,<br>– требует много вычислений,<br>– модель не пригодна для практической реализации,<br>нет семантического представления | – все достоинства номинальной модели,<br>– высокая надежность,<br>– модель устойчива к неопределенностям требований к трафику |
| **Легкая надежная модель** | – модель не пригодна для практической реализации,<br>– требует много вычислений,<br>– нет семантического представления | – все достоинства Г-надежной модели,<br>– учитываются затраты на надежность в составе модели |

Анализируя таблицу, можно заметить, что в качестве универсальной модели сетевой сегментации нужно присмотреться к модифицированной эталонной модели или модели многих S/D пар с учетом, конечно, их достоинств и недостатков. Эталонная модель действительно заслуживает свое название, так как ее эталонность доказывается разработанностью и универсальностью.

В информационно-ориентированных сетях, где главный сервис – это контент-ориентированный сервис предпочтительнее контент-ориентированная модель.

Для любых односервисных сегментов можно применить модель одной S/D пары.

## 4.3. Разработка модели идентификации и приоритизации трафика Интернета вещей на основе сегментации ресурсов в программно-конфигурируемых сетях

### *4.3.1. Модель контроля параметров качества обслуживания для приоритизации приложений Интернета Вещей в программно-конфигурируемых сетях*

Организация динамической приоритизации и управление трафиком приложений Интернета Вещей в условиях гетерогенности сетей позволит внедрить новые услуги, такие как тактильный интернет, дополненная реальность, медицинские приложения и другие [16; 41; 52; 91; 97].

Для обеспечения данных условий требуется инициализация и идентификация устройств Интернета Вещей с передачей требований по QoS от данных устройств управляющей системе сети. Стоит также учитывать то, что при построении полноценной инфраструктуры сети 5G/IMT-2020 сегментов программно-конфигурируемых сетей будет больше одного, также будут активно применяться виртуальные сегменты (с одной или более виртуальными сетевыми функциями), для их объединения используются так называемые оркестраторы, которые позволяют уже работать с подконтрольной инфраструктурой, как с единым ресурсом на более высоком уровне абстракции. Следовательно, требуется обеспечить процесс передачи требований по QoS к приложениям оркестратора сети. Для этого разработана модель протокола, цель которого организовать процесс регистрации Интернет Вещей и последующего их сопровождения при условии их мобильности. Также, на модель возлагается функции регулирования параметров QoS через организацию работы с оркестратором сети, который в свою очередь работает уже с модулями ПКС-контроллера(-ов) (версия протокола openflow не менее 1.3), отвечающими за эти задачи в сети и гипервизорами (возможно также и с виртуальными или логическими сущностями) сегментов виртуальных сетей. Данный фреймворк реализует собой определенный алгоритм взаимодействия систем (и/или их элементов) в сетях связи пятого поколения, при этом взаимодействие между некоторыми из них может происходит с помощью различных протоколов. В условиях появления и постепенного внедрения

различных сервисов IoT на работающую сеть, предлагаемая модель позволит разграничить трафик различных приложений IoT и сможет предоставить требуемые показатели QoS для определенных приложений.

### *4.3.2. Архитектура высокого уровня модели и общие описания взаимодействия элементов*

На Рисунке 51 сеть связи логически разделена на несколько сегментов: уровень доступа, уровень агрегации, уровень ядра сети, при этом такое распределение характерно для каждого из операторов.

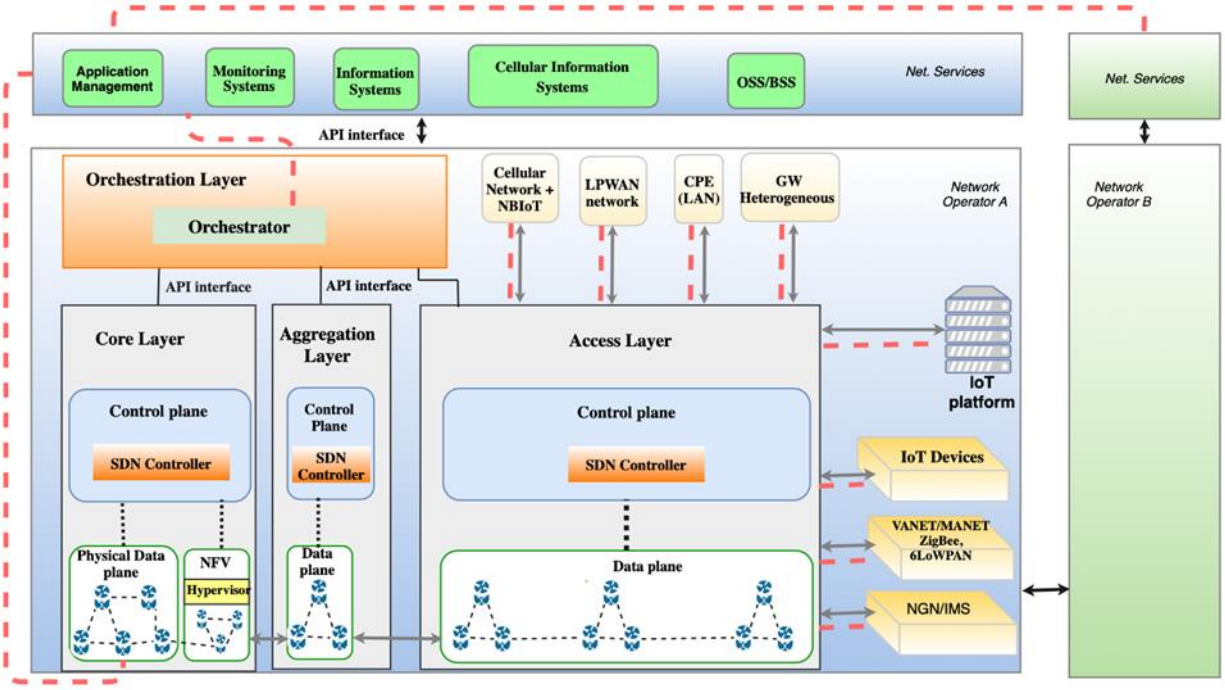


**Рисунок 51** – Инфраструктура сети связи с системой контроля QoS

При этом каждым из сегментов сети (доступ, агрегация, ядро) управляет ПКС-контроллер, также возможно наличие и виртуальных сегментов сетей, каждый из которых может реализовывать одну или более сетевых функций. Для реализации взаимодействия между сегментами необходимо наличие оркестратора сети, который в свою очередь по южному интерфейсу управляет контроллерами и взаимодействует с гипервизорами виртуальных сегментов сети, а также непосредственно с виртуальными машинами, реализующие ту или иную сетевую функцию. Однако с учетом того, что большинство коммерческих компаний, имеющие большие, распределенные корпоративные сети, уже на данный момент

переходят на технологии программно-конфигурируемых сетей и виртуализации сетевых функции, с уверенностью можно сказать, что данная модель позволит реализовать различные приложения Интернета Вещей, которые на данный момент труднореализуемы из-за тех требований, которые они предъявляют к сетям связи и способам управления их потоками. Например, такие приложения как: дополненная виртуальная реальность, тактильный интернет и медицинские сети очень требовательны к такому параметру, как круговая задержка (RTT), а некоторые из них также и к скорости и ее постоянности в процессе предоставления услуги (AR, VR). Поэтому, применение данной модели возможно сначала на сетях корпоративного масштаба с последующей его трансформацией на операторские коммерческие сети.

На Рисунке 52 красной пунктирной линией отображена взаимосвязь между основными элементами. Следующие элементы являются основными:

1) поддержка предлагаемой модели в виде приложения оркестратора сети (виртуальный или физический сервер);

2) поддержка данной модели в виде библиотеки языка разработки для устройств IoT, либо иного другого блока программного обеспечения, реализующего необходимый набор функций и логики реализации взаимодействия с другими элементами модели;

3) поддержка данной модели в виде библиотеки для сторонней платформы IoT, либо иного другого блока программного обеспечения, реализующего необходимый набор функций и логики реализации взаимодействия с другими элементами модели, предназначенного для платформы IoT, с которой требуется организовать канал с соответствующими показателями QoS с целью выполнения жесточайших критериев качества при предоставлении услуг в перспективных сетях связи пятого поколения;

Взаимодействия происходят между следующими устройствами:

– Интернет вещь и платформа IoT;

– платформа IoT и сервис провайдера, при этом взаимодействие происходит по защищенному, секретному каналу связи;

– платформа IoT и другая платформа IoT;

– приложение модели на оркестраторе – приложение одного оператора связи и другого оператора связи.

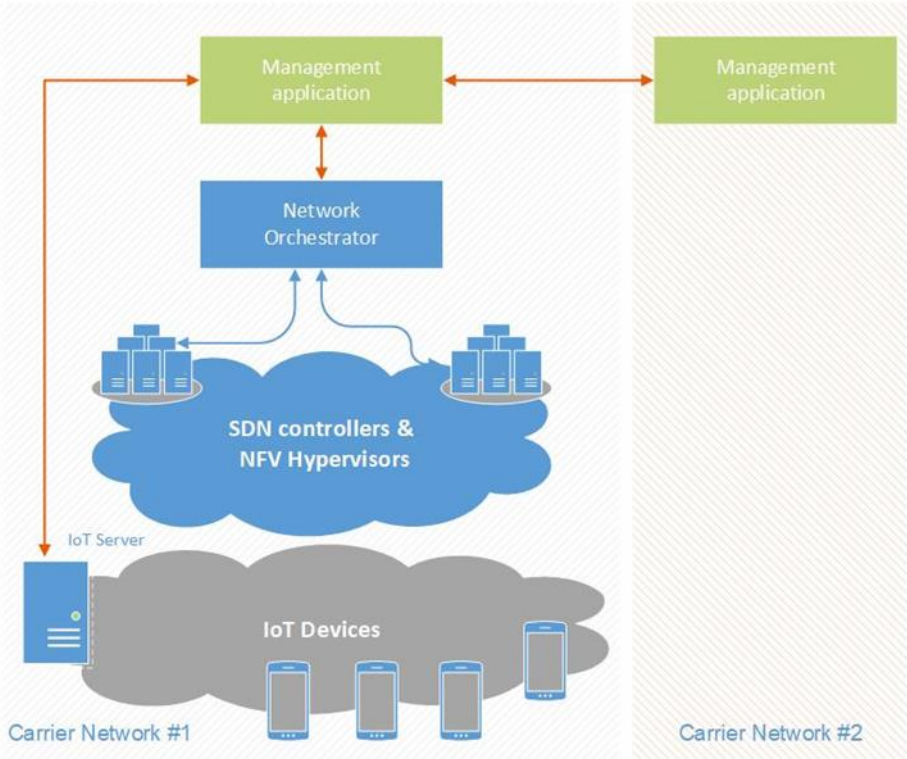


**Рисунок 52** – Взаимодействия между элементами модели контроля QoS

Организация элемента модели (Framework's App), в виде приложения оркестратора позволит реализовать услуги «прозрачно», по всей подконтрольной оператору сети, при этом на данном уровне реализованы другие информационные системы оператора, в том числе реализующие функционал по билингу, OSS/BSS, мониторингу, анализу данных и так далее, что позволит легко интегрировать данный функционал при реализации фреймворка.

### *4.3.3. Функциональные элементы модели*

**IoT-сервер** – это сервер, который может быть представлен в программном или программно-аппаратном виде. Сервис-провайдер предоставляет ряд услуг, развернутых на подконтрольном (-ых) ему сервере (-ах), например: хранение данных с IoT-устройств и их обработка, предоставление брокера MQTT, обеспечение установления связи между IoT-устройствами.

Таким образом, IoT-сервер реализует следующие главные функции:

– взаимодействие с IoT-device. Взаимодействие может быть организовано на основе одного из протоколов Интернета Вещей (HTTP 2.0, CoAP, MQTT, e.t.c);

– реализация функциональности AAA для зарегистрированных IoT-devices.

– взаимодействие с Management Application (MA). Взаимодействие происходит через API MA, использующийся при этом протокол определяется реализацией MA;

– взаимодействие с другим IoT-сервером. Данное взаимодействие предусматривается при предоставлении услуги, для которой необходимо соединение «IoT-device – IoT-device», с участием IoT-сервера (-ов). При этом, использующийся протокол, определяется реализацией IoT-серверов.

**Management Application (MA)** – это сервер, который может быть представлен как в программном, так и аппаратно-программном виде. Основные функции MA следующие:

– проверка возможности взаимодействия с IoT-сервером. Сервис-оператор для обеспечения качества предоставляемых услуг, заключает договор с оператором сети, в результате оператор сети заносит IP-адреса сервис-оператора в реестр. Данный реестр используется оператором сети при проверке на этапе запроса на взаимодействие от IoT-сервера (-ов). Также на данном этапе определяется ряд предоставляемых услуг оператором сети сервис-оператору (например, ограничение по параметрам QoS);

– взаимодействие с IoT-сервером. Данное взаимодействие происходит через API MA с помощью определенного протокола уровня приложений. Программный интерфейс реализует набор функций, достижимость некоторых определяется изначально установленной возможностью взаимодействия с IoT-сервером в процессе первоначальной проверки. Основными функциями является принятие запросов на обслуживание определенного устройства/группы устройств от IoT-сервера, а также сигнальный обмен с IoT-сервером в процессе сопровождения IoT-устройства(-в) и предоставления соответствующего качества;

– взаимодействие с оркестратором сети. Данное взаимодействие происходит через API оркестратора;

– взаимодействие «MA-MA». Данный вид взаимодействия возможен при необходимости установления соединения либо с IoT-сервером, находящемся в сети, подконтрольной другому оркестратору (этого же или другого оператора сети, например, международный роуминг), либо при установлении соединения «IoT-device – IoT-device», с учетом того, что одно из устройств находится в сети, подконтрольной другому оркестратору.

### *4.3.4. Организация контроля QoS для приоритизации приложений*

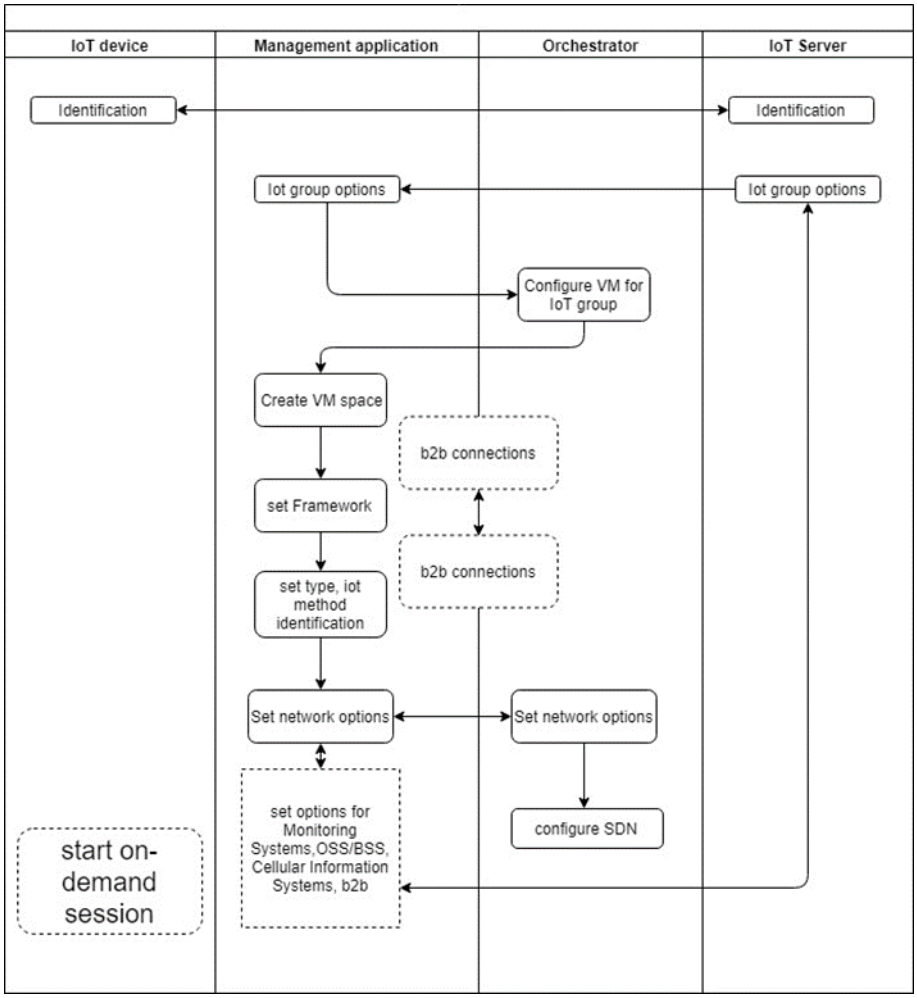


**Рисунок 53** – Процесс взаимодействия между основными элементами модели

1. «Процесс идентификации» – данный процесс реализует набор действий, которые направлены на выполнение следующих трех действий: AAA (Authentication, Authorization, Accounting). При этом данный процесс происходит только между Интернет Вещью (виртуальной или физической) и сторонней платформой IoT. При этом в процессе аутентификации происходит сопоставление запроса «Вещи» при установлении соединения и выделения ей вычислительных

ресурсов платформы согласно заранее заведенной в базу данных информации, позволяющей сопоставить данную вещь и конкретного пользователя, при этом Интернет Вещь проходит процесс аутентификации согласно уникальному ей идентификатору. Процесс авторизации позволяет определить ряд услуг, доступных данной вещи, при этом стоит также учесть, что в некоторых случаях «Вещь» может реализовывать определенный набор функций, характерных по своим задачам различному типу Вещей, то есть, например, Вещь может реализовывать функции ИВ Тактильного Интернета, а также одну и/или несколько медицинских функций, в соответствии с неоднородностью характеристик Интернет Вещей возможно их разделение (классификация) на группы в модели. В процессе accounting на сервере IoT (платформе) ведется учет использования вычислительных ресурсов, выделенных для конкретной Интернет Вещи, также ведется логирование процессов при их возможном сбое, в некоторых случаях возможно также реализации биллинга на стороне сервис-оператора (владельца IoT-платформы). Стоит отметить, что в данном процессе могут быть реализованы и другие функции, однако в каждом частном случае, набор функций определяется сервис-провайдером (владельцем IoT-платформы).

2. «Lot group options» – формирование сервером IoT и последующая передача параметров группы Интернета Вещей и/или добавление в данную группу Интернет Вещи с данными по требуемому QoS (Quality of Service), используемый протокол передачи данных (MQTT, CoAP, e.t.c.), уровень безопасности (требуется ли дополнительно шифрованные «каналы» связи) того или иного сектора (под сектором следует понимать участок сети).

3. «Configure VM for IoT-group» – после выполнения вышестоящей операции, модель производит передачу запроса на оркестраторе сети в соответствии с переданными данными в процессе «lot group options», после чего оркестратор производит конфигурацию виртуальной площадки (VM) для развертывания параметров данной модели под конкретную группу устройств IoT и/или производит дополнение/очищение группы конкретными устройствами.

4. Create VM Space – создание виртуальной площадки для подсистемы группы IoT. То есть, к примеру, создается обособленная «виртуальная площадка» для группы ИВ по определенным критериям обслуживания (различные сервис-операторы IoT, QoS, финансовое регулирование или биллинг, DPI-требования и др.). Либо добавляются в данную группу новые зарегистрированные устройства IoT. Если рассмотреть по первому критерию, приведенному в примере: сервис-оператор Интернета Вещей «А» (владелец IoT-платформы, предоставляющий определенный набор услуг и имеющий соглашение-контракт с оператором сети связи на поддержку работы по данной модели) и, например, равноценный по правам доступа сервис-оператор «В», работающие в одном сегменте сети, должны быть логически обособлены в модели.

5. Следующим шагом является процесс «Set Framework», который включает в себя ввод и настройку параметров сети, QoS, протокола IoT-группы. Следует отметить, что на данном шаге возможны также и дополнительные процессы («Set type, IoT method Identification»), которые необходимы в условиях межоператорских соединений. Как уже отмечалось выше, одним из взаимодействий в данной модели является взаимодействие «Framework’s App #1 – Framework’s App #N», при этом данное взаимодействие охватывает и другие процессы, такие как: «Create VM Space», «Set Framework».

6. Следующим шагом является «Set options for Monitoring Systems, OSS/BSS, Cellular Information Systems, b2b». Данные процессы направлены, при необходимости, на интеграцию с системами мониторинга, OSS/BSS и другими, необходимыми при предоставлении услуги оператором клиенту. При этом на Рисунке 53 можно также заметить обратную связь между процессами, что подразумевает реализацию полноценного мониторинга предоставления услуги с помощью модели информирование информационных систем не только оператора, но и сервера IoT. Например, в нештатной ситуации или иной другой, когда сеть, как система, ограниченная в ресурсах, не сможет далее обеспечить требуемое качество предоставления услуги, то на основе приоритетов, зависящих как от типа приложения Интернета Вещи, так и от договора между оператором сети и сервис-

оператором (владельца IoT-сервера) модель будет вынуждена отказать в продолжении предоставления услуги, при том, сообщить код ошибки серверу. Обратная связь с сервером IoT позволяет производить не только анализ и информирование систем друг друга, но и прогноз увеличения количества Интернета Вещей в каждом из секторов, их типе и так далее, что в итоге позволит производить более оптимальное бизнес-планирование операторов сетей связи, сервис-операторов IoT и т.д.

### *4.3.5. Моделирования сегмента модельной сети*

В этой части оценка производительности предлагаемой модели проводится в надежной среде моделирования. Многие среды моделирования являются пробными для внедрения современных компьютерных систем и проверки их производительности. Эти среды различаются по предоставляемым средствам и связанным с ними возможностям. Среда моделирования основана на Java и построена на основе CloudSim. Он позволяет создавать пограничные облака с разным количеством виртуальных машин (VM). Кроме того, удаленное выполнение веб-сервисов включено. Моделирование выполняется на рабочем компьютере с процессором Intel Core i5, частотой 3,07 ГГц и объемом памяти 8 ГБ. Результаты моделирования иллюстрируются в Таблице 7.

**Таблица 7** – Параметры моделирования

| *Параметр* | *Описание* | *Значение* |
|---|---|---|
| S | Число источников | 20 |
| $N_S$ | Количество сегментов | 6 (Приложения) |
| $N_{VM}$ | Число виртуальных машин | 8 |
| n | Размер заголовки идентификатора | 3 бита |
| $W_{mmax}$ | Максимальная нагрузка сервера | 100 флопс/с |
| λ | Интенсивность поступления | 15 Мбс |
| μ | Интенсивность обслуживания сервера | 8 Мбс |
| RAM, HDD | RAM, Память | 2048 Мб, 10 Гб |

Создан пограничный сервер с восемью виртуальными машинами. Пограничный сервер MEC работает на основе модели для пограничных серверов Micro-cloud. Двадцать разнородных источников создают и генерируют данные для шести различных приложений, каждое приложение рассматривается как отдельный сегмент. Каждое приложение резервирует часть ресурсов на пограничном вычислительном сервере (MEC server). Ресурсы для приложений предполагаются неоднородными, поэтому некоторым приложениям выделяется больше ресурсов, чем другим, на основе предварительно распределенной схемы. Эта схема устанавливается на основе вероятности запросов и потока данных, выделенных для каждого приложения. Для нашей реализации количество виртуальных машин, выделенных для каждого приложения, включено в Таблицу 8. Предполагается, что приложения 2 и 3 имеют высокий поток и, таким образом, выделяются обе виртуальные машины.

**Таблица 8** – Распределение ресурсов для разных приложений

| APP#1 | APP#2 | APP#3 | APP#4 | APP#5 | APP#6 |
|---|---|---|---|---|---|
| VM#7 | VM#0 & VM#1 | VM#6 | VM#2&VM#3 | VM#4 | VM#5 |

Для оценки производительности рассматриваются два основных параметра производительности; задержка и вероятность блокировки. Мы измеряем каждый параметр для каждого приложения в двух случаях. Первый случай представляет предлагаемую структуру, в которой ресурсы выделяются для каждого приложения. Второй случай представляет альтернативный случай, в котором сервера MEC служит без каких-либо классификаций. Все ресурсы сервера MEC служат для всех задач приложениями.

На Рисунке 54 представлено среднее время задержки для каждого приложения в обоих рассмотренных случаях. Для всех приложений ясно, что сетевая сегментация обеспечивает более высокую производительность.

На Рисунке 55 показана средняя вероятность блокировки для каждого приложения в обоих указанных случаях. Вероятность блокирования намного лучше в случае сегментации для всех рассматриваемых приложений.

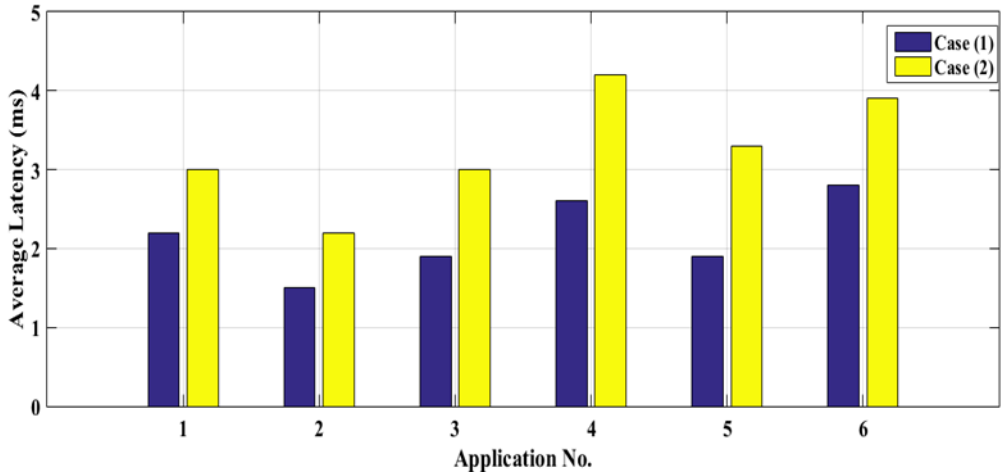


**Рисунок 54** – Среднее время задержки для каждого приложения в рассмотренных случаях

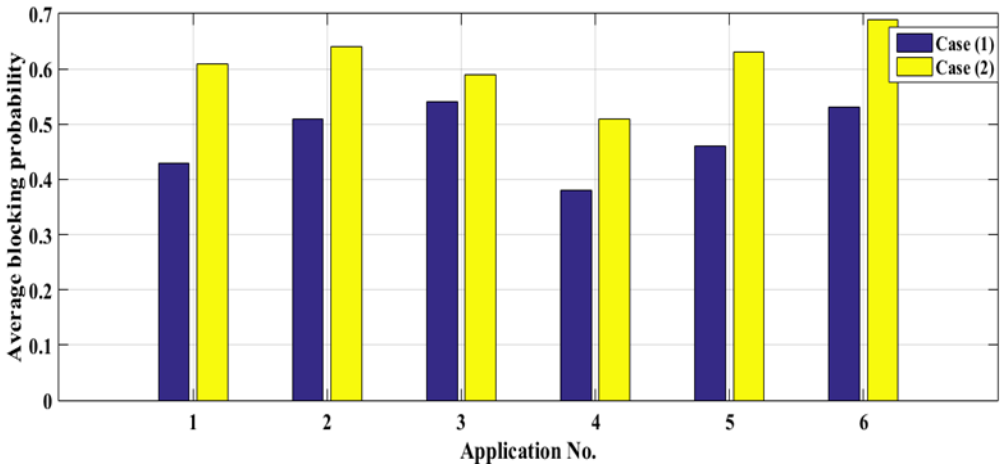


**Рисунок 55** – Средняя вероятность блокировки для каждого приложения в рассмотренных случаях

Технология программно-конфигурируемых сетей вместе с виртуализацией сетевых функции обеспечивает гибкость сети, позволяя динамически предоставить сетевые сегменты при необходимости. Новые приложения могут быть легко развернуты в сети в соответствии с требованиями заказчика. Сетевая сегментация радикально значительно упрощает процесс представления сетевых услуг в сетях 5G/IMT-2020 по сравнению с традиционными моделями реализации, позволяя динамическое распределение ресурсов сети по требованиям: пересылка трафика с учетом требований к обслуживанию, управление трафиком с учетом физического состояния каналов, объединение ресурсов облака и пропускная способность сети. Благодаря сетевой сегментации в сетях связи 5G/IMT-2020 способна

адаптироваться к растущим требованиям QoS, что позволяет создавать новые бизнес-модели, основанные на требованиях пользователя/приложения.

## Выводы по главе 4

1. Проведен сравнительный анализ методов сетевой сегментации.

2. Разработана модель идентификации и приоритизации трафика Интернета вещей на основе сегментации ресурсов в программно-конфигурируемых сетях.

3. Проведено имитационное моделирование в среде моделирования CloudSim и выявлены результаты эффективной работы разработанной модели.

# ЗАКЛЮЧЕНИЕ

В ходе данной диссертационной работы решены следующие задачи:

1. Проведен анализ основных требований к будущим сетям связи, в частности, к сетям пятого поколения 5G/IMT-2020.

2. Показано, что одна из задач будущих сетей связи состоит в представлении сетевых услуг по запросу приложений или пользователей.

3. Проведен анализ архитектур построения программно-конфигурируемых сетей, обозначены концепции и разработки, способствовавшие её появлению и последующему развитию.

4. Проведен анализ сетевой сегментации как архитектурного решения организации сетевой инфраструктуры для оптимизации сетевых ресурсов, сетевых функции и сетевых сервисов.

5. Проанализированы принципы функционирования программно-конфигурируемой сети для последующего описания в виде СМО.

6. С целью исследования особенностей функционирования ПКС-контроллера разработана методика проведения тестирования. На базе модельной сети лаборатории проведено тестирование и представлены результаты проведенных экспериментов по тестированию контроллера OpenDaylight Beryllium SR 4 и аппаратной платформы, программно-конфигурируемых сетей с использованием разработанной методики.

7. Разработаны модели программно-конфигурируемой сети, позволяющие оценить эффективность её работы в условиях высокой нагрузки и рационально планировать размещение элементов сети на этапе развертывания и масштабирования.

8. Проведён сравнительный анализ таких характеристик контроллера как, производительность, время обработки потоков, стабильность и масштабируемость. Показано, что использование кластера контроллеров в 3 раза улучшает возможности сети в представлении требуемого качества обслуживания.

9. Проведен сравнительный анализ методов динамического распределения контроллеров в программно-конфигурируемых сетях.

10. Предложен метод распределения ПКС-контроллеров по алгоритму DDC для кластеризации ресурсов сети.

11. Проведено моделирование в среде моделирования Matlab и выявлены результаты эффективной работы предлагаемых алгоритмов.

12. Предложен метод классификации трафика в программно-конфигурируемых сетях на основе модифицированного алгоритма k-means. Модификация алгоритма k-means состоит в определении размерности пространства, правил оценки численных характеристик и способа оценки качества принимаемого решения.

13. Разработана модель классификации и приоритизации трафика ПКС, которая была апробирована на базе модельной сети.

14. Проведен сравнительный анализ методов сетевой сегментации.

15. Разработана модель идентификации и приоритизации трафика Интернета вещей на основе сегментации ресурсов в программно-конфигурируемых сетях.

16. Проведено имитационное моделирование в среде моделирования CloudSim и выявлены результаты эффективной работы разработанной модели.

В диссертационной работе получены следующие научные результаты, которые могут быть использованы для внедрения на сетях операторов связи:

1. Разработана методику проведения тестирования ПКС-контроллера и представлены результаты проведенных экспериментов по тестированию ПКС-контроллера OpenDaylight Beryllium SR 4 и аппаратной платформы, программно-конфигурируемых сетей с использованием разработанной методики на основе утилит Cbench, ПО ProLan QuTester Plus, OCCT и Mininet. Проведён сравнительный анализ таких характеристик контроллера как, производительность, время обработки потоков, стабильность и масштабируемость.

2. Разработана модель ПКС, на базе которой выполнена серия компьютерных экспериментов. Результаты показывают, что с увеличением подключаемого числа коммутаторов, соответственно, увеличивается нагрузка на контроллере, вплоть до критического значения. Результаты моделирования также показывают, что

использование кластера из 2 контроллеров пропорционально увеличивается среднее время обслуживания запроса. Кластеры в 3 раза улучшают возможности сети в представлении требуемого качества обслуживания.

3. Предложен метод распределения ПКС-контроллер по алгоритму DDC (Dynamic and Distributed Clustering – динамически и распределённый алгоритм кластеризации) для кластеризации ресурсов сети по критерии уровня загруженности контроллера. Результаты показывают, что предложен алгоритм, который уравновешивает рабочую нагрузку между контроллерами и предотвращает перегрузочные ситуации, позволяя избежать сбой сети.

4. Разработан метод классификации и приоритизации сетевого трафика на основе модифицированного алгоритма k-means с учетом параметров трафика и предварительного обучения. Результаты показывают, что разработанный метод позволяет распределять ресурсы сети по определённым приоритизированным типам трафика, что позволяет оптимизировать работы приложений поверх программно-конфигурируемых сетей.

5. Разработана структура идентификации и приоритизации трафика Интернета вещей на основе сегментации ресурсов в программно-конфигурируемых сетях. Предлагаемый подход позволяет осуществлять динамическое распределение ресурсов сети по требованиям: пересылка трафика с учетом требований к обслуживанию, управление трафиком с учетом физического состояния каналов, объединение ресурсов облака и пропускная способность сети.

Перспективное направление дальнейших исследований может быть связано с разработкой протокола обеспечения изоляции и безопасности сетевых сегментов в процессе динамического распределения сетевых ресурсов пользователями/приложениями.

# СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

| | |
|---|---|
| 5G/IMT-2020 | Fifth Generation network |
| API | Application Programming Interface |
| CPU | Central Processor Unit |
| ETSI | European Telecommunications Standards Institute |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IoT | Internet of Things |
| ITU-T | International Telecommunication Union – Telecommunication sector |
| NFV | Network Function Virtualization |
| OF-CONFIG | OpenFlow Management and Configuration Protocol |
| ONF | Open Network Foundation |
| QoS | Quality of Service |
| RAN | Radio Access Network |
| SDN | Software Defined Networks |
| VNF | Virtualized Network Function |
| ПКС | Программно-Конфигурируемые Сети |
| СМО | Система Массового Обслуживания |
| ЦОД | Центр Обработки Данных |

# СПИСОК ЛИТЕРАТУРЫ

[1] Бородин А.С., Сети связи пятого поколения как основа цифровой экономики / А.С. Бородин, А.Е. Кучерявый // Электросвязь. – 2017. – No 5. – С. 45–49.

[2] Владыко, А.Г. Тестирование SDN контроллеров на базе модельной сети / А.Г. Владыко, Н.А. Матвиенко, М.И. Новиков, Р.В. Киричек // Информационные технологии и телекоммуникации. – 2016. – Т. 4. – No 1. – С. 17–28.

[3] Гимадинов Р. Ф., Кластеризация в мобильных сетях 5G. случай частичной мобильности / Гимадинов Р. Ф., Мутханна А. С., Кучерявый А. Е // Информационные технологии и телекоммуникации. – 2015. – № 2 (10). – С. 44–52. – ISSN: 2307–1303.

[4] Гольдштейн Б.С. Сети связи пост-NGN / Б.С.Гольдштейн, А.Е.Кучерявый // БХВ, С.Петербург, 2013.

[5] Кучерявый А.Е. Сети связи общего пользования. Тенденции развития и методы расчёта / А.Е. Кучерявый, А.И. Парамонов, Е.А.Кучерявый // СПб: ФГУП ЦНИИС, 2008.

[6] Кучерявый, А.Е. Интернет вещей / А.Е. Кучерявый // Электросвязь. – 2013. –No 1. – С. 21–24.

[7] Мухизи С., Анализ возможностей применения SDN/NFV в мобильных сетях 5G / Мухизи С., Киричёк Р.В. // Молодежная научная школа по прикладной теории вероятностей и телекоммуникационным технологиям (АРТСТ). 2017. С. 166–174.

[8] Мухизи с., Исследование моделей балансировки нагрузки в программно-конфигурируемых сетях / Мухизи с., Мутханна А.С.; Киричёк Р.В, Кучерявый А.Е. // Электросвязь. 2019, No 01. С. 23–29

[9] Мухизи С., Анализ технологии сетевого слайсинга в сетях связи пятого поколения / Мухизи С., Киричёк Р.В. // Информационные Технологии И Телекоммуникации; 2017 Том 5 No 4. С. 57–63.

[10] Мухизи, С. Исследование и разработка имитационной модели функционирования фрагмента программно-конфигурируемой сети как системы

массового обслуживания / С. Мухизи, Г.И. Шамшин, А.С. Мутханна, Р.В. Киричек // Информационные технологии и телекоммуникации. – 2016. – Т. 4, No 4. – С. 49–57.

[11] 3GPP TR 22.830 V0.3.0 (2018); Technical Specification Group Services and System Aspects; Feasibility Study on Business Role Models for Network Slicing. [Электронный ресурс], Режим доступа: https://www.3gpp.org/ftp/Specs/archive/22_series/22.830/

[12] 3GPP TS 28.530 v0.5.0; Management of network slicing in mobile networks; Concepts, use cases and requirements (Release 15), February 2018. [Электронный ресурс], Режим доступа: https://www.3gpp.org/ftp/Specs/archive/28_series/28.530/

[13] 3GPP. [Электронный ресурс], Режим доступа: http://www.3gpp.org

[14] 5G PPP, View on 5G Architecture, July 2016. [Электронный ресурс], Режим доступа: https://5g-ppp.eu/wp-content/uploads/2017/07/5G-PPP-5G-Architecture-White-Paper-2-Summer-2017_For-Public-Consultation.pdf

[15] 5G PPP. [Электронный ресурс], Режим доступа: https://5g-ppp.eu/

[16] Abuarqoub A. , Behaviour Profiling in Healthcare Applications Using the Internet of Things Technology / A. Abuarqoub, MH. Hammoudeh // Proceedings of Fourth International Conference on Advances in Information Processing and Communication Technology, 2016.

[17] Afolabi I., Network Slicing & Softwarization: A Survey on Principles, Enabling Technologies & Solutions / I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, H. Flinck. // DOI 10.1109/COMST.2018.2815638, IEEE Communications Surveys & Tutorials.

[18] Alsmadi I.M., A Systematic Literature Review on Software-Defined Networking/ Alsmadi I.M., AlAzzam I., Akour M.// Information Fusion for Cyber-Security Analytics. SCI, 2017, vol. 691, pp. 333–369.

[19] Ansell J., Making Queueing Theory More Palatable to SDN/OpenFlow-based Network Practitioners/ Ansell J., Seah W., Ng B., Marshall S. // IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 1119–1124 (2016).

[20] Ateya A., Intelligent core network for Tactile Internet system / A. Ateya, A. Muthanna, I. Gudkova , A. Vybornova and A. Koucheryavy // International Conference on Future Networks and Distributed Systems, ACM, P.15, Cambridge, Jul.2017.

[21] Baumgartner A., Network slice embedding under traffic uncertainties – A light robust approach/ A. Baumgartner, T. Bauschert, A.A. Blzarour, V.S. Reddy // Proceedings of the International Conference on Network and Service Management (CNSM), Tokyo (2017), pp. 1–5.

[22] BBF SD-406 End-to-End Network Slicing. [Электронный ресурс], Режим доступа: https://www.broadband-forum.org/5g

[23] Benzekki K., Software-defined networking (SDN): a survey / K. Benzekki, A. El Fergougui, A. Elbelrhiti Elalaoui // Security and Communication Networks. – 2016. – Vol. 9(18). – pp.5803–5833

[24] Berde P., ONOS: Towards an open, distributed SDN OS / P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, // SIGCOMM HotSDN. New York, NY, USA: ACM, 2014, pp. 1–6.

[25] Blanco B., Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN / Blanco B., FajardoJ.O., Giannoulakis I., Kafetzakis E., Peng S., Pérez-Romero J., Trajkovska I., Khodashenas P.S., Goratti L., Paolino M., Sfakianakis E. //. 2017, Computer Standards & Interfaces.pp. 216–228.

[26] Blial O., An Overview on SDN Architectures with Multiple Controllers / O. Blial, M. Ben Mamoun, R. Benaini // Journal of Computer Networks and Communications. – 2016. – Vol.2016

[27] Borshchev A., The Big Book of Simulation Modeling: Multimethod Modeling with Anylogic 6 / Borshchev A. // AnyLogic North America (2013).

[28] Caraguay A. L. V., SDN: Evolution and Opportunities in the Development IoT Applications / A. L. V. Caraguay, A. B. Peral, L. I. B. Lopez, and L. J. G. Villalba // International Journal of Distributed Sensor Networks, p. 10, May 2014.

[29] Dixit A., ElastiCon: An Elastic Distributed SDN Controller / A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, R. R. Kompella // CM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). – 2014. – С. 17–27

[30] Dombacher C., Queueing Models for Call Centres. 2010. [Электронный ресурс], Режим доступа: http://www.telecomm.at/documents/Queueing_Models_CC.pdf

[31] ETSI Draft V7 – 12th July 2017, Network Slicing Terms and Systems. [Электронный ресурс], Режим доступа: https://datatracker.ietf.org/meeting/99/materials/slides-99-netslicing-alex-galis-netslicing-terms-and-systems-02.

[32] ETSI GS NFV-EVE 012 (V3.1.1), Network Functions Virtualization (NFV) Release3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework, December 2017. [Электронный ресурс], Режим доступа: https://docbox.etsi.org/isg/nfv/open/Publications_pdf/Specs-Reports/NFV-EVE%20007v3.1.1%20-%20GS%20-%20NFVI%20Hw%20rqmts%20spec.pdf

[33] ETSI, Mobile Edge Computing A key technology towards 5G, ETSI White Paper, No. 11, September 2015. [Электронный ресурс], Режим доступа: https://www.etsi.org/images/files/etsiwhitepapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[34] ETSI, Network Functions Virtualization (NFV)-ArchitecturalFramework.ETSIGSNFV002V1.2.1(Dez.2014). [Электронный ресурс], Режим доступа: http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.02.01_60/gs_nfv002v010201p.pdf

[35] ETSI. [Электронный ресурс], Режим доступа: https://www.etsi.org/

[36] Feamster N., The road to SDN: an intellectual history of programmable networks. / N. Feamster, J. Rexford, E. Zegura // ACM SIGCOMM Computer Communication Review. – 2014. – Vol. 44(2). – С. 87–98

[37] Ferrús R., On 5G Radio Access Network Slicing: Radio Interface Protocol Features and Configuration / R. Ferrús, O. Sallent, J. Pérez-Romero and R. Agustí // IEEE Communications Magazine, vol. PP, no. 99, pp. 2-10. 2018.

[38] Foukas X., Network slicing in 5G: survey and challenges / X. Foukas, G. Patounas, A. Elmokashfi, et al. // IEEE Commun. Mag. 55(5), 94–100 (2017).

[39] Fu Y., A hybrid hierarchical control plane for flow-based large-scale software-defined networks / Fu Y, J. Bi, Z. Chen, K. Gao, B. Zhang, G. Chen, and J. Wu // IEEE Trans. Netw. Service Manag., vol. 12, no. 2, pp. 117–131, June 2015

[40] Fu Y., Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks / Y. Fu, J. Bi, K.Gao, Z. Chen, J. Wu, B. Hao // Proc. IEEE ICNP, pp. 569–576, Oct. 2014.

[41] Fujdiak R., Advanced optimization method for improving the urban traffic management / Fujdiak R., Masek P., Mlynek P., Misurec J., Muthanna A // In: 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT), pp. 48–53. IEEE (2016)

[42] Galis A., Towards 5G Network Slicing – Motivations and Challenges/ A. Galis, Chih-Lin I // IEEE 5G Tech Focus, vol. 1, 2017

[43] Ganjali Y., HyperFlow: A distributed control plane for OpenFlow / Y. Ganjali, A. Tootoonchian//Proceedings of the internet network management conference on Research on enterprise networking. Berkeley, CA, USA, pp. 3–3, 2010.

[44] GSMA, Introduction to network slicing. [Электронный ресурс], Режим доступа: https://www.gsma.com/futurenetworks/wp-content/uploads/2017/11/GSMA-An-Introduction-to-Network-Slicing.pdf

[45] GSMA. [Электронный ресурс], Режим доступа: https://www.gsma.com/futurenetworks/

[46] Heller B., Leveraging SDN layering to systematically troubleshoot networks / B. Heller, C.Scott, N. McKeown, S. Shenker, A.Wundsam, Ho. Zeng, S. Whitlock, V. Jeyakumar, N.Handigol, J. M.McCauley, K. Zarifis, P. Kazemian // Proc. ACM SIGCOMM Workshop Hot Topics SDN, pp. 37–42, 2013.

[47] Heller B., The controller placement problem / B. Heller, R. Sherwood, N. McKeown // Proceedings of the first workshop on Hot topics in software defined networks, ACM, pp. 7–12, 2012.

[48] Hu Y., Reliability-aware controller placement for Software-Defined Networks / Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan // IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 672–675, 2013.

[49] IEEE. [Электронный ресурс], Режим доступа: https://www.ieee.org/

[50] IETF, Interconnecting (or Stitching) Network Slice Subnets. [Электронный ресурс], Режим доступа: https://tools.ietf.org/html/draft-defoy-coms-subnet-interconnection-03

[51] IETF, Software-Defined Networking (SDN): Layers and Architecture Terminology. [Электронный ресурс], Режим доступа: https://tools.ietf.org/html/rfc7426

[52] Ignatova L., Analysis of the Internet of Things devices integration in 5G networks / Khakimov A., Mahmood A., Muthanna A. // Systems of Signal Synchronization, Generating and Processing in Telecommunications (SINKHROINFO). 2017. 4 p.

[53] ITU-T IMT2020: Application of network softwarization to IMT-2020. [Электронный ресурс], Режим доступа: http://www.itu.int/en/ITU-T/focusgroups/imt-2020/Pages/default.aspx.

[54] ITU-T Y.3011 Framework of network virtualization for future networks. [Электронный ресурс], Режим доступа: https://www.itu.int/rec/T-REC-Y.3011-201201-I/en

[55] ITU-T Y.3150 (2018): High-level technical characteristics of network softwarization for IMT-2020. [Электронный ресурс], Режим доступа: https://www.itu.int/rec/T-REC-Y.3150-201801-I/en

[56] Jagadeesan N. A., Software-Defined Networking Paradigms in Wireless Networks: A Survey / N. A. Jagadeesan, B. Krishnamachari // ACM Computing Surveys, vol. 47, no. 2, pp. 27:1–11, 2015.

[57] Jain S.: Experience with a globally deployed software defined WAN. / S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J.

Zhou, M. Zhu, J. Zolla, U. Hlzle, S. Stuart, and A. Vahdat. // ACM SIGCOMM Computer Communication Review. – 2013. – Vol. 43(4). – C. 3–14

[58] Jararweh Y., SDIoT: A Software Defined Based Internet of Things Framework / Jararweh Y., Al-Ayyoub M., Darabseh A., Benkhelifa E., Vouk M., Rindos A // Journal of Ambient Intelligence and Humanized Computing, pp. 453–461. 2015

[59] Jiang D., An Overview of 5G Requirements / D. Jiang, G. Liu // 5G Mobile Communications, Springer, pp. 3–26, 2017.

[60] Jin H, Information-centric mobile caching network frameworks and caching optimization: a survey / H Jin, D Xu, C Zhao, D. Liang // EURASIP J. Wirel. Commun. Netw. 2017, pp. 1–33

[61] Jin H., Content-oriented network slicing optimization based on cache-enabled hybrid radio access network / Jin H., Lu H, Zhao C. J. // Wireless Communications and Network. 2018. [Электронный ресурс], Режим доступа: https://doi.org/10.1186/s13638-017-0995-z.

[62] Karakus M. A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN) / M. Karakus, A. Durresi // Computer Networks. – 2017. – Vol. 112. – C. 279–293

[63] Mohammad S. M., Machine Learning for Internet of Things Data Analysis: A Survey / Mohammad S. M., Mohammadreza R., Mohammadamin B., Peyman A., Payam B., Amit P. S. // Digital Communications and Networks, Volume 4, Issue 3, August 2018, Pages 161–175.

[64] Muthanna A., September. Analytical Evaluation of D2D Connectivity Potential in 5G Wireless Systems / Muthanna A., Masek P., Hosek J., Fujdiak, R., Hussein O., Paramonov A. and Koucheryavy, A // In International Conference on Next Generation Wired/Wireless Networking Springer International Publishing. –pp. 395–403. 2016.

[65] Kirichek R., Model Networks for Internet of Things and SDN / Kirichek R., Vladyko A. Zakharov M., Koucheryavy A. // ICACT, pp. 76–79. IEEE 2016.

[66] Kirichek R., Software-Defined Architecture for Flying Ubiquitous Sensor Networking / Kirichek R., Vladyko A., Paramonov A., Koucheryavy A.// ICACT, pp. 158–162. 2017.

[67] Koponen T., Onix: A Distributed Control Platform for Large-Scale Production Networks / Koponen T., Casado M., Gude N //Proc. of the 9th USENIX Conf. on OSDI, USENIX Association, Berkeley, CA, USA (2010), pp. 351–364

[68] Kotulski Z., Towards constructive approach to end-to-end slice isolation in 5G networks / Kotulski Z., Nowak T., Sepczuk M. et al. // *EURASIP Journal on Information Security* 2018. [Электронный ресурс], Режим доступа: https://doi.org/10.1186/s13635-018-0072-0.

[69] Kreutz D., Software-Defined Networking: A Comprehensive Survey / D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig. // Proc. IEEE, vol. 103, no. 1, pp. 14–76, Jan. 2015

[70] Ksentini A., Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction/ A. Ksentini, N. Nikaein // in IEEE Communications Magazine, vol. 55, no. 6, pp. 102-108, 2017.

[71] Leconte M., A Resource Allocation Framework for Network Slicing / M. Leconte, G. Paschos, P. Mertikopoulos, U. Kozat // IEEE International Conference on Computer Communications (INFOCOM 2018), Apr. 2018.

[72] Liu L., Field Trial of an OpenFlow-Based Unified Control Plane for Multilayer Multigranularity Optical Switching Networks / Liu L., Zhang D., Tsuritani T., et al.// Journal of Lightwave Technology, 31 (4), pp. 506–514 (2013).

[73] McKeown N., OpenFlow: Enabling innovation in campus networks. / N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner. // ACM SIGCOMM Computer Communication Review. – 2008. – Vol. 38(2). – С. 69–74.

[74] Mohamed Azab, Towards proactive SDN-controller attack and failure resilience / Mohamed Azab, José A.B. Fortes // 2017 International Conference on Computing, Networking and Communications (ICNC), 10.1109/ICCNC.2017.7876169, pp. 442–448.

[75] Muhizi S., A Novel Slice-Oriented Network Model / Muhizi S., Ateya A.A., Muthanna A.S, Kirichek R.V, Koucheryavy A.E. // DCCN 2018. Communications in Computer and Information Science, vol 919. pp 421–431.Springer, Cham

[76] Muhizi S., Analysis and performance evaluation of SDN queue model / Muhizi S., Shamshin G., Muthanna A. S., Kirichek R. V., Vladyko A. G, Koucheryavy A. E. // WWIC 2017. Lecture Notes in Computer Science, vol 10372. pp 37–48. Springer, Cham.

[77] Muhizi S., Framework of QoS Management for Time Constraint Services with Requested Network Parameters based on SDN/NFV Infrastructure /., Muthanna A.S. Volkov A. N., Khakimov A. Kirichek R.V., Koucheryavy A.E.// 2018 ICUMT.pp 1–6.

[78] NGMN 5G White Paper. [Электронный ресурс], Режим доступа: https://www.ngmn.org/fileadmin/ngmn/content/images/news/ngmn_news/NGMN_5G_White_Paper_V1_0.pdf

[79] NGMN, description of network for service provider networks [Электронный ресурс], Режим доступа: https://www.ngmn.org/fileadmin/user_upload/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf

[80] NGMN, Network Slicing Framework. [Электронный ресурс], Режим доступа: https://www.ngmn.org/uploads/media/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf.

[81] NGMN. [Электронный ресурс], Режим доступа: https://ngmn.org/

[82] Nicira Networks: Disruptive Network Virtualization. [Электронный ресурс], Режим доступа: https://web.stanford.edu/class/ee204/2012/Nicira%20Networks.pdf

[83] ONF OpenFlow Management and Configuration Protocol (OF-Config 1.1.1). [Электронный ресурс], Режим доступа: https://www.opennetworking.org/wp-content/uploads/2013/02/of-config-1-1-1.pdf

[84] ONF OpenFlow-enabled SDN and Network Functions Virtualization. [Электронный ресурс], Режим доступа: https://www.opennetworking.org/wp-content/uploads/2013/05/sb-sdn-nvf-solution.pdf

[85] ONF OpenFlow Switch Specification. [Электронный ресурс], Режим доступа: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[86] ONF Software-Defined Networking: The New Norm for Networks. [Электронный ресурс], Режим доступа: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[87] ONF TR-502 SDN architecture. [Электронный ресурс], Режим доступа: https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf

[88] ONF. [Электронный ресурс], Режим доступа: https://www.opennetworking.org/

[89] OpenDayLight. [Электронный ресурс], Режим доступа: https://www.opendaylight.org

[90] Ordonez-Lucena J., Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges/ J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, J. Folgueira // IEEE Communications Magazine, vol. 55, no. 5, pp. 80–87, May 2017.

[91] Pan J., An Internet of Things Framework for Smart Energy in Buildings: Designs, Prototype, and Experiments / J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, and M. Sha // IEEE Internet of Things Journal, vol. 2, no. 6, pp. 527–537, Dec. 2015.

[92] Phemius K., DISCO: distributed multi-domain SDN controllers / K. Phemius, M. Bouet, and J. Leguay. // IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–4,

[93] Pirmagomedov R., Dynamic Data Packaging Protocol for Real-Time Medical Applications of Nanonetworks / R. Pirmagomedov, M. Blinnikov, R. Glushakov, A. Muthanna, R. Kirichek, A.Koucheryavy // Internet of Things, Smart Spaces, and Next Generation Networks and Systems, pp.196–205 (2017)

[94] Rost P., Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks / Rost P., Mannweiler C., Michalopoulos D., Sartori C., et al. // IEEE Commun. Mag., vol. 55, no. 5, pp. 72–79, May 2017.

[95] Sallent O., On radio access network slicing from a radio resource management perspective / O. Sallent, J. Perez-Romero, R. Ferrus, and R. Agusti // IEEE Wireless Communications, vol. PP, no. 99, pp. 2–10, 2017.

[96] Salman O., SDN controllers: A comparative study / O. Salman, I. H. Elhajj, A. Kayssi, A. Chehab // MELECON 2016. pp. 1 – 6. 2016

[97] Sandhya**,** A survey: Hybrid SDN / Sandhya, Yash Sinha, K. Haribabu // Journal of Network and Computer Applications Volume 100, 15 December 2017, pp. 35–55.

[98] Scott-Hayward S., A survey of security in software defined networks / S. Scott-Hayward, S. Natarajan, and S. Sezer // IEEE Communications Surveys Tutorials, vol. 18, no. 1, pp. 623– 654, Firstquarter 2016.

[99] Szabo D., Towards the tactile internet: decreasing communication latency with network coding and software defined networking / Szabo D., Gulyas A., Fitzek F.H., Lucani D.E. // 21th European Wireless Conference; pp. 1–6. 2015, May.

[100] Tootoonchian A., HyperFlow: A distributed control plane for OpenFlow / A. Tootoonchian, Y. Ganjali // Proceedings of the internet network management conference on Research on enterprise networking. Berkeley, CA, USA, pp. 3–3, 2010.

[101] Vladyko A., Comprehensive SDN Testing Based on Model Network / Vladyko A., Muthanna A., Kirichek R. // Internet of Things, Smart Spaces, and Next Generation. LNCS, vol. 9870, pp. 539–549. 2016.

[102] Vladyko A., Fuzzy Model of Dynamic Traffic Management in Software-Defined Mobile Networks / Vladyko A., Letenko I., Lezhepekov A., Buinevich M. // Internet of Things, Smart Spaces, and Next Generation. LNCS, vol. 9870, pp. 561–570. 2016.

[103] Volkov A., SDN approach to control Internet of Thing medical applications traffic / Volkov A., Muhathanna A., Pirmagomedov R., Kirichek R // DCCN 2017. CCIS, vol. 700, pp. 467–476. Springer, Cham (2017).

[104] Xia W., A Survey on Software-Defined Networking / Xia W., Wen Y., Foh C., Niyato D., Xie H. // IEEE Communications Surveys & Tutorials, 17 (1), 27–51 (2015).

[105] Xiong B., A Concise Queuing Model for Controller Performance in Software-Defined Networks / Xiong B., Peng X., Zhao J. // Journal of Computers, 11 (3), 232–237 (2016).

[106] Xiong B., Performance evaluation of OpenFlow-based software-defined networks based on queueing model / Xiong B., Yang K., Zhao J., Li W., Li K. // Computer Networks, 102, 174–183 (2016).

[107] Xu H., Achieving High Scalability Through Hybrid Switching in Software-Defined Networking / H. Xu, H. Huang, S. Chen, G. Zhao, L. Huang // Networking IEEE/ACM Transactions on, vol. 26, no. 1, pp. 618–632, 2018.

[108] Yazici V., Controlling a Software-Defined Network via Distributed Controllers / V. Yazici, M. Oguz Sunay, Ali O. Ercan // NEM Summit. – 2012. – C.16–20

[109] Yeganeh S. H., Kandoo: A framework for efficient and scalable offloading of control applications / S. H. Yeganeh, Y. Ganjali, // SIGCOMM HotSDN. New York, NY, USA: ACM, 2012, pp. 19–24.

# Приложение А. МЕТОДИКА ИСПЫТАНИЙ КОНТРОЛЛЕРА ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ

Программная методика испытаний (ПМИ) контроллера разработана на основе ГОСТ 19.301-79 «Программа и методика испытаний, требования к содержанию и оформлению» и предназначена для проведения испытаний, измерения характеристик и метрик контроллера программно-определяемых сетей.

При разработке методики, также использовалась спецификация IETF – Terminology for Benchmarking SDN Controller Performance, которая задает основные параметры контроллера ПКС для тестирования и указывает рекомендации по составлению сценариев тестирования.

Целью программы испытаний является разработка способов оценки программно-аппаратных характеристик контроллера ПКС, с целью использования его на реальных сетях связи.

Представленная программа может быть использована специалистами по тестированию и разработчиками платформ управления программно-конфигурируемых сетей.

1. **Объект испытаний**

*Название объекта:*

Объектом предварительных испытаний является контроллер программно-конфигурируемой сети.

*Обозначение:*

Контроллер ПКС, SDN Controller.

*Область применения:*

Контроллер ПКС предназначен для упрощения развертывания инфраструктуры программно-определяемой сети (SDN), и централизованного автоматического управления сетью, сервисами и сетевым оборудованием.

2. **Цель испытаний**

– Проведение комплексного программно-аппаратного тестирования SDN контроллера, с целью измерения его характеристик.

– Проверка разработанной методологии испытаний, при проведении натурных экспериментов на базе лаборатории «Программно-конфигурируемых сетей » кафедры сетей связи и передачи данных, с целью выявления недостатков и доработки методики.

– Анализ измеренных характеристик, и определение граничных значений.

– Детерминировать взаимодействие программного и аппаратного обеспечения контроллера критериям и признакам его надежного функционирования.

3. **Требования к методике**

– Методика должна обеспечивать соответствие взаимодействия программных и аппаратных характеристик контроллера ПКС и обеспечивать оценку его граничных значений.

– Методика должна быть универсальной вне зависимости, от реализаций архитектуры объектов управления ПКС.

4. **Общие положения**

*4.1.* ***Перечень руководящих документов***

Настоящая методика разработана в соответствии со следующими документами:

– IETF – Benchmarking Methodology for SDN Controller Performance;

– RFC 2889 – Методология бенчмаркинга для коммутационных устройств LAN;

– ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению.

*4.2.* ***Место и продолжительность испытаний***

– Испытательный стенд находится на территории университета телекоммуникаций имени проф. Бонч-Бруевича, на базе лаборатории «Программно-конфигурируемых сетей» кафедры сетей связи и передачи данных.

– Тестирования проводились в течении 25 дней.

*4.3.* ***Перечень ранее проведенных испытаний***

Испытания по тестированию ПКС проводились в рамках НИОКР по разработке отечественного SDN контроллера в ЦПИКС, а также во многих научно-исследовательских центрах и коммерческих компаниях за рубежом.

*4.4.* ***Перечень предъявляемых на испытания документов***

Документация, используемая в тестировании:

– Общее описание контроллера ПКС;

– Руководство пользователя;

– Описание программного и аппаратного обеспечения для тестирования;

– Сценарии тестирования;

– Методика тестирования.

5. **Объем испытаний**

***5.1. Перечень этапов испытаний***

Этапы испытаний представлены в Таблице 9.

**Таблица 9** – Этапы испытаний

| *№* | *Объект испытаний* | *Требование* | *Наименование испытания* | *Вид испытания* | *Оцениваемые характеристики (мера измерения)* |
|---|---|---|---|---|---|
| 1 | Контроллер ПКС | Не менее одного миллиона запросов в секунду | Измерение пропускной способности контроллера в зависимости от числа подключенных управляемых коммутаторов, при постоянном количестве хостов за каждым коммутатором | Нагрузочное | Пропускная способность контроллера [потоки/секунды] |
| 2 | Контроллер ПКС | Не менее одного миллиона запросов в секунду | Измерение пропускной способности контроллера в зависимости от числа хостов подключенных к постоянному количеству управляемых контроллером коммутаторов | Нагрузочное тестирование | Пропускная способность контроллера [потоки/секунды] |
| 3 | Контроллер ПКС | Не менее одного миллиона запросов в секунду<br>Не более 100–160 миллисекунд | Измерение времени обработки одного запроса (задержки) в зависимости от числа подключенных коммутаторов, при постоянном количестве хостов | Нагрузочное тестирование | Время обработки одного запроса [миллисекунды] |
| 4 | Контроллер ПКС | Не менее одного миллиона запросов в секунду<br>Не более 100–160 миллисекунд | Измерение времени обработки одного запроса (задержки) в зависимости от числа оконечных узлов, при постоянном количестве коммутаторов | Нагрузочное тестирование | Время обработки одного запроса [миллисекунды] |

Продолжение таблицы 9

| *№* | *Объект испытаний* | *Требование* | *Наименование испытания* | *Вид испытания* | *Оцениваемые характеристики (мера измерения)* |
|---|---|---|---|---|---|
| 5 | Контроллер ПКС | Не менее одного миллиона запросов в секунду<br>Работа с не менее чем с 256 коммутаторами | Исследование масштабируемости контроллера (производительность контроллера, в режиме максимальной нагрузки на контроллер) | Стрессовое тестирование | Производительность процессора контроллера [Kbyte/se],<br>Доступность файлового сервиса (контроллера ПКС) [%*] |
| 6 | Контроллер ПКС | Безотказная работа в течении 24 часов тестирования<br>Работа с не менее чем с 64 коммутаторами | Тестирование надежности работы контроллера (производительность контроллера, в режиме максимальной нагрузки на контроллер) | Нагрузочное тестирование | Производительность процессора контроллера [%*],<br>Занимаемая память RAM [Гб],<br>Температура [°C] |
| 7 | Контроллер ПКС | Не менее 10000 количество измерений | Тестирование производительности контроллера, в зависимости от времени и частоты изменения топологии сети, при постоянном параметре количества измерений. | Нагрузочное тестирование | Производительность процессора контроллера [%*],<br>Занимаемая память RAM [Гб, %*] |
| *Примечание* – %* – мера измерения, показатель количества от общего числа. | | | | | |

### *5.2. Последовательность проведения испытаний*

Последовательность проведения испытаний представляет собой алгоритм состоящий, из четырех этапов. Испытания, присутствующие на следующем этапе не начинаются, пока не завершатся все процессы предыдущего этапа.

Этап 1. Разработка методик испытаний контроллера.

Для проведения испытаний контроллера были разработаны следующие методики тестирования:

1. Измерение пропускной способности контроллера в режиме нагрузочного тестирования, в зависимости от количества управляемых контроллером коммутаторов (1, 2, 4, 8, 16, 32, 64, 128), при фиксированном количестве оконечных узлов (1000000).

2. Измерение пропускной способности в режиме нагрузочного тестирования, в зависимости от количества подключенных оконечных узлов (10, 100, 1000, 10000, 100000, 1000000), при фиксированном количестве коммутаторов (32).

3. Измерение времени обработки одного запроса (задержки) контроллера в режиме нагрузочного тестирования, в зависимости от количества управляемых контроллером коммутаторов (1, 2, 4, 8, 16, 32, 64, 128).

4. Измерение времени обработки одного запроса (задержки) контроллера в режиме нагрузочного тестирования в зависимости от количества подключенных оконечных узлов (10, 100, 1000, 10000, 100000, 1000000), при фиксированном количестве управляемых контроллером коммутаторов (32). В каждом эксперименте проводилось 10 тестов продолжительностью 90 секунд каждый, после чего высчитывалось среднее значение исследуемого параметра. Тестирование проводилось при 4 активных ядрах центрального процессора (ЦП).

5. Исследование масштабируемости контроллера. Измерение производительности контроллера в режиме стрессового тестирования, при фиксированном количестве управляемых контроллером коммутаторов (256) и оконечных узлов (1000000). На каждой конфигурации проводилось 10 тестов продолжительностью 10 секунд каждый, после чего высчитывалось среднее значение исследуемого параметра. Тестирование проводилось при 4 активных ядрах ЦП.

6. Тестирование стабильности работы контроллера. Измерение производительности контроллера в режиме нагрузочного тестирования, при фиксированном количестве управляемых контроллером коммутаторов (32) и оконечных узлов (1000000), на временном интервале t = 20 часов (78000 секунд). Тестирование проводилось при 4 активных ядрах ЦП.

Измерение производительности контроллера, в зависимости от времени изменения заданной топологии сети на контроллере (10, 1, 0.1 секунд), при соответствующем параметре количества измерений. На каждой конфигурации проводилось 10 тестов продолжительностью 10 секунд каждый, после чего высчитывалось среднее значение исследуемого параметра. Тестирование проводилось при 4 активных ядрах ЦП.

Этап 2. Проведение первичных испытаний контроллера.

Этап 3. Выявления ошибок ПМИ и их корректировка.

Этап 4. Проведение главных испытаний и анализ результатов.

***5.3. Требования по испытаниям контроллера программно-конфигурируемой сети***

- Испытания должны быть проведены в той последовательности, которая указана в пункте 3.5.2.
- Должны быть проведены испытания определенных программно-аппаратных характеристик контроллера ПКС.
- Испытания должны показать, что определенные характеристики контроллера ПКС удовлетворяют заданным критериям оценок и метрикам тестирования.

***5.4. Перечень работ, проводимых после завершения испытаний***

По завершению испытаний, полученные результаты заносят в книгу испытаний, которая находится в лаборатории. Также оформляется протокол испытаний, который содержит соответствующие общие и специальные данные.

Общие данные:

1) описание материальной стороны испытуемого объекта – наименование, цвет, применение, предприятие-изготовитель и др.;

2) даты проведения испытаний и состав лиц, принимавших участие в испытаниях;

3) условия нахождение испытуемого объекта до проведения испытаний;

4) условия испытаний – относительная влажность воздуха в помещении, температура окружающей среды, атмосферное давление и др.;

5) измерительные приборы с указанием их класса точности и описанием лабораторного стенда для испытаний.

Специальные данные обуславливаются назначением и методом испытаний, особенностями испытуемого объекта.

Протокол подписывает руководитель и исполнители, проводившие испытания.

**6. Условия и порядок проведения испытаний**

***6.1. Условия проведения испытаний***

Испытания должны проводиться в нормальных климатических условиях по ГОСТ 22261-94. Условия проведения испытаний приведены ниже:

– температура окружающего воздуха, °С – 20 ± 5;

– относительная влажность, % – от 30 до 80;

– атмосферное давление, кПа – от 84 до 106;

– частота питающей электросети, Гц – 50 ± 0,5;

– напряжение питающей сети переменного тока, В – 220 ± 4,4.

### *6.2. Условия начала и завершения отдельных этапов испытаний*

Необходимым и достаточным условием завершения этапа испытаний и начала последующего этапа является успешное завершение проверок, проводимых на этом этапе. Перечень проверок должен включать в себя:

1) проверку комплектности программной документации (ПД);

2) проверку комплектности состава технических и программных средств.

### *6.3. Меры, обеспечивающие безопасность и безаварийность проведения испытаний*

При проведении испытаний необходимо обеспечить соблюдение требований безопасности, установленных ГОСТ 12.2.007.0–75 «Система стандартов безопасности труда. Изделия электротехнические. Общие требования безопасности», «Правилами техники безопасности при эксплуатации электроустановок потребителей» и «Правилами технической эксплуатации электроустановок потребителей».

### 6.4. *Материально-техническое обеспечение испытаний*

Для проведения испытаний необходимо обеспечить достаточную материально-техническую базу и выполнить все первичные действия, с помощью которых проводятся испытания.

Под материально-техническим обеспечением в данной работе понимается:

– лабораторный стенд, состоящий из разных аппаратно-программных комплексов (АПК) таких как:

- персональные компьютеры, используемые в качестве оконечных узлов (хостов);

- сервер, как головной управляющий элемент сети, на котором установлено программное обеспечения ПКС-контроллера;
- коммутатор поддерживающий протокол Openflow, основной обязанностью которого является коммутация хостов;
- коммутаторы агрегации, для подключения оконечных узлов;
- маршрутизатор, обеспечивающий доступ в сеть Internet.

– совокупность программных средств (ПС) для развертывания и тестирования контроллера ПКС. Устанавливаемые ПС:

- контроллер ПКС OpenDaylight Beryllium SR;
- программное обеспечение Open vSwitch и утилиту Cbench;
- программное обеспечение ProLan QuTester Plus;
- программное обеспечение OCCT Perestroika v 4.5.0;
- программное обеспечение Mininet;
- обустроенные рабочие места для проведения испытаний;

Под первичными действиями для проведения испытаний понимается:

– сборка лабораторного стенда для проведения испытаний;

– установка необходимого программного обеспечения на клиентское и серверное оборудование лабораторного стенда;

– обеспечение стабильного взаимодействия элементов стенда;

– общая организация процесса испытаний.

# Приложение Б.
# ЛИСТИНГ ПРИЛОЖЕНИЯ ТЕСТИРОВАНИЯ КОНТРОЛЛЕРА ПОВЕРХ MININET

```
__author__ = 'muhizi'
from mininet.node import CPULimitedHost
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.log import setLogLevel, info
from mininet.node import RemoteController
from mininet.cli import CLI
#from mininet.clean import Cleanup
import threading
from multiprocessing import Process
import os
from time import sleep
import random
"""
Instructions to run the topo:
   1. Go to directory where this fil is.
   2. run: sudo –E python Simple_Pkt_Topo.py.py

The topo has 4 switches and 4 hosts. They are connected in a star shape.
"""
class SimplePktSwitch(Topo):
   """Simple topology example."""

   def __init__(self, **opts):
      """Create custom topo."""

      # Initialize topology
      # It uses the constructor for the Topo class
      super(SimplePktSwitch, self).__init__(**opts)

      # Add hosts and switches
      h1 = self.addHost('h1')
      h2 = self.addHost('h2')
      h3 = self.addHost('h3')
      h4 = self.addHost('h4')

      h5 = self.addHost('h5')
      h6 = self.addHost('h6')
      h7 = self.addHost('h7')
      h8 = self.addHost('h8')
      h9 = self.addHost('h9')
```

```
h10= self.addHost('h10')
h11= self.addHost('h11')
h12= self.addHost('h12')
h13= self.addHost('h13')
h14= self.addHost('h14')
h15= self.addHost('h15')
h16= self.addHost('h16')
h17= self.addHost('h17')
h18= self.addHost('h18')
h19= self.addHost('h19')

# Adding switches
s1 = self.addSwitch('s1', dpid="0000000000000001")
s2 = self.addSwitch('s2', dpid="0000000000000002")
s3 = self.addSwitch('s3', dpid="0000000000000003")
s4 = self.addSwitch('s4', dpid="0000000000000004")

# Add links h's & s's
self.addLink(h1, s1)
self.addLink(h2, s2)
self.addLink(h3, s3)
self.addLink(h4, s4)
self.addLink(h5, s1)
self.addLink(h6, s1)
self.addLink(h7, s1)
self.addLink(h8, s1)
self.addLink(h9, s2)
self.addLink(h10,s2)
self.addLink(h11,s2)
self.addLink(h12,s2)
self.addLink(h13,s3)
self.addLink(h14,s3)
self.addLink(h15,s3)
self.addLink(h16,s3)
self.addLink(h16,s4)
self.addLink(h17,s4)
self.addLink(h18,s4)
self.addLink(h19,s4)

## Add links s's
self.addLink(s1, s2)
self.addLink(s1, s3)
self.addLink(s1, s4)
self.addLink(s2, s4)
self.addLink(s3, s4)
self.addLink(s2, s3)
```

```
def DynamicTopo(net, itter, delay):
    # This function for dynamic changing network topology
    i = 0
    #sleep(delay)
    while i < itter:
        print('through the 5 seconds topology will change')
        numS1 = random.randrange(1, 4, 1)
        #numS2 = random.randrange(1, 4, 1)
        numS2 = 2
        numS1 = 's'+str(numS1)
        numS2 = 's'+str(numS2)
        Mininet.configLinkStatus(net, numS1, numS2, 'down')
        print('look a new topo')
        sleep(delay)
        Mininet.configLinkStatus(net, numS1, numS2, 'up')
        i = i + 1
        print('Dynamic Iterration is: ' + ' ' + str(i) + '.....')

def IperfTest(net):
    # this function for pingig between hosts

    print('Iperf test between swithes')
    h1, h4 = net.get('h1', 'h4')
    net.iperf((h1, h4)) # between s1 & s4
    #net.iperf('h2', 'h3') # betwenn s2 & s3
    print('Stoped Iperf test')

def run():
    #Cleanup()

    c = RemoteController('c', '192.168.0.3', 6653)
    net = Mininet(topo=SimplePktSwitch(), host=CPULimitedHost, controller=None)
    net.addController(c)
    net.start()

    #IperfTest(net)
    Proc_1 = Process(target=DynamicTopo(net, 500000, 10000)) # (net, itteration, time_delay)
    #Proc_2 = Process(target=IperfTest(net))

    Proc_1.start()
    #Proc_2.start()

    Proc_1.join()
    #Proc_2.join()
```

```
    CLI(net)
    net.stop()

# if the script is run directly (sudo custom/optical.py):
if __name__ == '__main__':
    setLogLevel('info')
    run()
```

# Приложение В.
# ДОКУМЕНТЫ, ПОДТВЕРЖДАЮЩИЕ ВНЕДРЕНИЕ ОСНОВНЫХ РЕЗУЛЬТАТОВ ДИССЕРТАЦИОННОЙ РАБОТЫ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

Юридический адрес: набережная реки Мойки, д. 61, Санкт-Петербург, 191186

Почтовый адрес: пр. Большевиков, д. 22, корп. 1, Санкт-Петербург, 193232
Тел.(812) 3263156, Факс: (812) 3263159
E-mail: rector@sut.ru
ИНН 7808004760 КПП 784001001
ОГРН 1027809197635 ОКТМО 40909000

18.02.2019 № _____
на № _____ от _____

УТВЕРЖДАЮ:
Проректор по научной работе

К.В. Дукельский

Акт

о внедрении научных результатов,

полученных Мухизи Самуэлем в диссертационной работе

«РАЗРАБОТКА МОДЕЛЕЙ И МЕТОДОВ СЕГМЕНТАЦИИ РЕСУРСОВ В ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЯХ».

Комиссия в составе декана факультета Инфокоммуникационных сетей и систем Л.Б.Бузюкова, доцента кафедры сетей связи и передачи данных М.А.Маколкиной и заведующей лабораторией кафедры сетей связи и передачи данных О.И. Ворожейкина составила настоящий акт в том, что научные результаты, полученные в диссертации «Разработка моделей и методов сегментации ресурсов в программно-конфигурируемых сетях», использованы при чтении лекций, проведении практических занятий и лабораторных работ по курсам:

1. Интернет Вещей (Рабочая программа № 02.12.15/788, утверждена Первым проректором-проректором по учебной работе Г.М. Машковым 21.09.2015), разделы Программы:

- Сети M2M. Классификация сетей M2M по видам трафика. Модели для опосредованного и псевдодетерминированного трафика. Пуассоновский, самоподобный и антиперсистентный трафик. Влияние трафика M2M на качество обслуживания традиционных услуг связи (речь, видео, данные). Способы уменьшения влияния трафика M2M.
- Ad Hoc или самоорганизующиеся сети. Приложения самоорганизующихся сетей. Всепроникающие сенсорные сети как технологическая основа внедрения концепции Интернета Вещей.

При этом используются следующие новые научные результаты, полученные Мухизи С. в диссертационной работе:

- методика испытаний контроллера программно-конфигурируемых сетей в условиях высокой функциональной нагрузки.
- метод идентификации и приоритезации трафика приложений Интернета Вещей программно-конфигурируемых сетей.

2. Сети связи (Рабочая программа № 02.12.13/861, утверждена Первым проректором-проректором по учебной работе Г.М. Машковым 11.02.2016), разделы Программы:

- Управление информационными потоками в глобальных сетях, хранение информации, в т.ч. распределенное. Архитектура центров обработки данных. Распределенные облачные вычисления.

При этом используются следующие новые научные результаты, полученные Мухизи С. в диссертационной работе:

- метод сегментации ресурсов в программно-конфигурируемых сетях за счет распределения контроллеров ПКС и балансировки трафика в зависимости от приложений.

Кроме того, научные результаты, полученные Мухизи Самуэлем были использованы при подготовке вкладов СПбГУТ в Сектор Стандартизации Телекоммуникаций Международного Союза Электросвязи:

- Q.SDN-CT "Framework of SDN controller testing"
-Q.QMP-TCA "QoS management protocol for time constraint applications over SDN"

Декан факультета ИКСС — Л.Б. Бузюков

Доцент каф. ССиПД — М.А. Маколкина

Зав. лабораторией кафедры ССиПД — О.И. Ворожейкина